

ALGORITMICA GRAFURILOR

Conf. univ. dr. COSTEL BĂLCĂU

2020

Tematica

1	Aranjamente, combinări, permutări	6
1.1	Preliminarii	6
1.2	Produs cartezian	7
1.3	Submulțimi	8
1.4	Aranjamente cu repetiție	9
1.5	Aranjamente	9
1.6	Permutări	10
1.7	Combinări	11
1.8	Combinări cu repetiție	14
1.9	Permutări cu repetiție	15
2	Partiții	18
2.1	Compuneri ale unui număr natural	18
2.2	Partiții ale unui număr natural	20
2.3	Partiții ale unei mulțimi finite	22
3	Grafuri	25
3.1	Definiții generale	25
3.2	Reprezentarea (memorarea) grafurilor	30
3.3	Grade, secvențe grafice	33
3.4	Conexitate	40
3.5	Parcurgerea grafurilor	43
3.6	Algoritmul Roy-Warshall	48
4	Arbori și arborescențe	53
4.1	Numărul cicromatic	53
4.2	Teorema de caracterizare a arborilor	56
4.3	Teorema de caracterizare a arborescențelor	59
4.4	Numărarea arborilor parțiali	61
4.5	Numărarea arborescențelor parțiale	64
5	Arbori parțiali de cost minim	66
5.1	Expunerea problemei	66
5.2	Algoritmii lui Kruskal și Prim	68
6	Clase particulare de grafuri	74
6.1	Grafuri euleriene	74
6.2	Grafuri hamiltoniene	78
6.3	Colorări în grafuri	92

6.4	Grafuri bipartite	95
7	Distanțe și drumuri minime	104
7.1	Expunerea problemei	104
7.2	Algoritmul Dijkstra	106
7.3	Algoritmul Roy-Floyd	110
8	Fluxuri în rețele	114
8.1	Problema fluxului maxim	114
8.2	Algoritmul Ford-Fulkerson	116

Evaluare

- Activitate laborator: 30% (Programe obligatorii, programe suplimentare și probleme din Temele de laborator)
- Teme de casă: 20% (Programe suplimentare și probleme nerezolvate în timpul laboratorului, din Temele de laborator)
- Examen final: 50% (Probă scrisă: teorie, algoritmi -cu implementare- și probleme)

Bibliografie

- [1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Massachusetts, 2009.
- [2] Gh. Barbu, V. Păun, *Calculatoare personale și programare în C/C++*, Editura Didactică și Pedagogică, București, 2005.
- [3] Gh. Barbu, V. Păun, *Programarea în limbajul C/C++*, Editura Matrix Rom, București, 2011.
- [4] Gh. Barbu, I. Văduva, M. Boloșteanu, *Bazele informaticii*, Editura Tehnică, București, 1997.
- [5] C. Bălcău, *Combinatorică și teoria grafurilor*, Editura Universității din Pitești, Pitești, 2007.
- [6] O. Bâscă, L. Livovschi, *Algoritmi euristici*, Editura Universității din București, București, 2003.
- [7] E. Ciurea, *Algoritmi. Introducere în algoritmica grafurilor*, Editura Tehnică, București, 2001.
- [8] E. Ciurea, L. Ciupală, *Algoritmi. Introducere în algoritmica fluxurilor în rețele*, Editura Matrix Rom, București, 2006.
- [9] I. Codreanu, M. Lascu, *Probleme de combinatorică*, Editura Gil, Zalău, 2016.
- [10] T.H. Cormen, *Algorithms Unlocked*, MIT Press, Cambridge, 2013.
- [11] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, 2009.
- [12] C. Croitoru, *Tehnici de bază în optimizarea combinatorie*, Editura Universității "Al. I. Cuza", Iași, 1992.
- [13] S. Even, *Graph Algorithms*, Cambridge University Press, 2012.
- [14] D. Fanache, *Teoria algoritmică a grafurilor*, Editura Paralela 45, Pitești, 2016.
- [15] H. Georgescu, *Tehnici de programare*, Editura Universității din București, București, 2005.
- [16] C.A. Giumale, *Introducere în analiza algoritmilor. Teorie și aplicații*, Editura Polirom, Iași, 2004.
- [17] F.V. Jensen, T.D. Nielsen, *Bayesian Networks and Decision Graphs*, Springer, New York, 2007.
- [18] D. Jungnickel, *Graphs, Networks and Algorithms*, Springer, 2013.
- [19] M. Keller, W. Trotter, *Applied Combinatorics*, Open Textbook Library, 2017.
- [20] D.E. Knuth, *The Art Of Computer Programming. Vol. 4A: Combinatorial Algorithms*, Addison-Wesley, Massachusetts, 2011.
- [21] B. Korte, J. Vygen, *Combinatorial Optimization. Theory and Algorithms*, Springer, 2012.
- [22] L. Livovschi, H. Georgescu, *Sinteza și analiza algoritmilor*, Editura Științifică și Enciclopedică, București, 1986.
- [23] D. Logofătu, *Algoritmi fundamentali în Java: Aplicații*, Editura Polirom, Iași, 2007.
- [24] D. Lucanu, M. Craus, *Proiectarea algoritmilor*, Editura Polirom, Iași, 2008.

- [25] L. Panaitopol, A. Szilard, D. Șerbănescu, *Probleme de combinatorică pentru juniori*, Editura Gil, Zalău, 2019.
- [26] V. Pop, L. Popa, D. Popa, M. Olteanu, G. Mincu, M. Burlică, R. Strungariu, M. Ispas, *Teme și probleme pentru concursurile studențești de matematică. Volumul II. Concursuri internaționale*, Editura StudIS, Iași, 2013.
- [27] V. Pop, M. Teleucă, *Probleme de combinatorică elementară. Numărare, grafuri, jocuri*, Editura Matrix Rom, București, 2013.
- [28] D.R. Popescu, *Combinatorică și teoria grafurilor*, Societatea de Științe Matematice din România, București, 2005.
- [29] D.R. Popescu, R. Marinescu-Ghemeci, *Combinatorică și teoria grafurilor prin exerciții și probleme*, Editura Matrix Rom, București, 2014.
- [30] C.P. Popovici, H. Georgescu, L. State, *Bazele informaticii. Vol. I și II*, Editura Universității din București, București, 1990-1991.
- [31] S.S. Ray, *Graph Theory with Algorithms and its Applications*, Springer, New Delhi, 2013.
- [32] O.A. Schipor, S.G. Pentiuc, F. Gîză-Belciug, *Limbaajul C - Tehnici de programare eficientă*, Editura Matrix Rom, București, 2014.
- [33] R. Sedgewick, K. Wayne, *Algorithms*, Addison-Wesley, Massachusetts, 2011.
- [34] N.O. Stănășilă ș.a., *Teme și probleme pentru concursurile studențești de matematică. Volumul III. Concursuri naționale*, Editura StudIS, Iași, 2013.
- [35] R. Stephens, *Essential Algorithms: A Practical Approach to Computer Algorithms*, Wiley, Indianapolis, 2013.
- [36] T. Toadere, *Grafe. Teorie, algoritmi și aplicații*, Editura Albastră, Cluj-Napoca, 2002.
- [37] I. Tomescu, *Combinatorică și teoria grafurilor*, Tipografia Universității din București, București, 1978.
- [38] I. Tomescu, *Probleme de combinatorică și teoria grafurilor*, Editura Didactică și Pedagogică, București, 1981.
- [39] I. Tomescu, *Data structures*, Editura Universității din București, București, 2004.
- [40] ***, *Handbook of combinatorics*, edited by R.L. Graham, M. Grötschel and L. Lovász, Elsevier, Amsterdam, 1995.
- [41] ***, *Handbook of discrete and combinatorial mathematics*, edited by K.H. Rosen, J.G. Michaels, J.L. Gross, J.W. Grossman and D.R. Shier, CRC Press, Boca Raton, 2000.
- [42] ***, *Revista MATINF. Publicație bianuală de matematică și informatică pentru elevi și profesori*, editată de Departamentul Matematică-Informatică, Universitatea din Pitești, Editura Universității din Pitești.

Tema 1

Aranjamente, combinări, permutări

1.1 Preliminarii

În această lecție vom prezenta formule de numărare și algoritmi de generare (enumerare) pentru cele mai cunoscute familii de obiecte combinatoriale: produs cartezian, submulțimi, aranjamente (fără repetiție, cu repetiție sau ordonate), combinări (fără repetiție sau cu repetiție), permutări (fără repetiție sau cu repetiție). Reamintim în continuare câteva noțiuni uzuale.

Definiția 1.1.1. Fie A o mulțime finită. Numărul de elemente ale lui A , notat cu $\text{card}(A)$, se numește **cardinalul** mulțimii A .

Definiția 1.1.2. Fie A un **alfabet** (adică o mulțime finită) și $n \in \mathbb{N}^*$. O secvență de forma

$$a = a_1 a_2 \dots a_n, \text{ cu } a_1, a_2, \dots, a_n \in A,$$

se numește **cuvânt de lungime n** peste alfabetul A . Lungimea cuvântului a se notează cu $|a|$.

Observația 1.1.1. Evident, cuvântul $a_1 a_2 \dots a_n$ poate fi identificat cu vectorul (a_1, a_2, \dots, a_n) .

Definiția 1.1.3. Fie A o mulțime (alfabet) total ordonată și $x = (x_1, \dots, x_n)$, $y = (y_1, \dots, y_m)$ doi vectori (cuvinte) cu elemente (litere) din A . Spunem că x este **mai mic decât y în ordine lexicografică** și notăm $x \prec y$ dacă

$$(x_1, \dots, x_n) = (y_1, \dots, y_n) \text{ și } m > n$$

sau dacă există un indice i , $i \leq \min\{m, n\}$, astfel încât

$$(x_1, \dots, x_{i-1}) = (y_1, \dots, y_{i-1}) \text{ și } x_i < y_i.$$

Exemplul 1.1.1. Considerând alfabetul $A = \{a, b, c, \dots, z\}$ și ordinea alfabetică uzuală $a < b < c < \dots < z$, avem

$$\text{ion} \prec \text{ionela} \prec \text{ionescu}.$$

Definiția 1.1.4. Fie $x \in \mathbb{R}$ și $n \in \mathbb{N}$. Notăm

$$[x]_n = \begin{cases} 1, & \text{dacă } n = 0, \\ \underbrace{x(x-1) \dots (x-n+1)}_{n \text{ factori}}, & \text{dacă } n \geq 1, \end{cases} \quad [x]^n = \begin{cases} 1, & \text{dacă } n = 0, \\ \underbrace{x(x+1) \dots (x+n-1)}_{n \text{ factori}}, & \text{dacă } n \geq 1. \end{cases}$$

$[x]_n$ se numește **polinomul factorial descrescător** de gradul n , iar $[x]^n$ se numește **polinomul factorial crescător** de gradul n .

Exemplul 1.1.2. Avem $[1,5]_3 = 1,5 \cdot 0,5 \cdot (-0,5) = -0,375$, iar $[1,5]^3 = 1,5 \cdot 2,5 \cdot 3,5 = 13,125$.

1.2 Produs cartezian

Definiția 1.2.1. *Produsul cartezian al mulțimilor A_1, A_2, \dots, A_n ($n \in \mathbb{N}^*$) este mulțimea*

$$A_1 \times A_2 \times \dots \times A_n = \{(a_1, a_2, \dots, a_n) | a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n\}.$$

Exemplul 1.2.1. $\{a, b\} \times \{+, -\} \times \{c\} = \{(a, +, c), (a, -, c), (b, +, c), (b, -, c)\}.$

Propoziția 1.2.1 (de numărare a produsului cartezian). *Fie $n \in \mathbb{N}^*$ și A_1, A_2, \dots, A_n mulțimi finite. Atunci $\text{card}(A_1 \times A_2 \times \dots \times A_n) = \text{card}(A_1) \cdot \text{card}(A_2) \cdot \dots \cdot \text{card}(A_n).$*

Demonstrație. Se utilizează metoda inducției matematice după n . □

Algoritmul 1.2.1 (de generare a produsului cartezian). Fie mulțimile standard

$$A_1 = \{1, 2, \dots, m_1\}, A_2 = \{1, 2, \dots, m_2\}, \dots, A_n = \{1, 2, \dots, m_n\}.$$

Vom utiliza ”regula următorului”.

- Primul element al produsului cartezian $A_1 \times A_2 \times \dots \times A_n$, în ordine lexicografică, este

$$(c_1, c_2, \dots, c_n) = (1, 1, \dots, 1).$$

- Un element arbitrar (curent)

$$(c_1, \dots, c_{k-1}, c_k, c_{k+1}, \dots, c_n)$$

are un element următor dacă și numai dacă există un indice $k \in \{n, \dots, 1\}$ astfel încât $c_k < m_k$. În acest caz, luând cel mai mare indice k cu această proprietate, elementul următor, în ordine lexicografică, este

$$(c_1, \dots, c_{k-1}, c_k + 1, 1, \dots, 1).$$

În pseudocod, algoritmul poate fi descris sub forma

```

PRODUS_CARTEZIAN( $n, m$ ):
  for  $i = \overline{1, n}$  do  $c[i] \leftarrow 1$ ;
  AFISARE( $c, n$ );
  repeat
     $k \leftarrow n$ ;
    while ( $c[k] = m[k]$ ) and ( $k > 0$ ) do  $k \leftarrow k - 1$ ;
    if ( $k > 0$ ) then
       $c[k] \leftarrow \overline{c[k] + 1}$ ;
      for  $i = \overline{k + 1, n}$  do  $c[i] \leftarrow 1$ ;
      AFISARE( $c, n$ );
  while ( $k > 0$ );

```

unde funcția de afișare este

```

AFISARE( $c, n$ ):
  for  $i = \overline{1, n}$  do
     $\_ \text{afișează } c[i]$ ;

```


Observația 1.2.1. Pentru generarea produsului cartezian $A_1 \times A_2 \times \dots \times A_n$ al unor mulțimi finite arbitrare

$$A_1 = \{a_{11}, a_{12}, \dots, a_{1m_1}\}, A_2 = \{a_{21}, a_{22}, \dots, a_{2m_2}\}, \dots, A_n = \{a_{n1}, a_{n2}, \dots, a_{nm_n}\}$$

se poate folosi algoritmul anterior, bazat pe generarea indicilor, înlocuind afișarea indicilor c_i cu afișarea elementelor corespunzătoare a_{ic_i} din mulțimile A_i , adică înlocuind funcția **AFIȘARE**(c, n) cu funcția

```
AFIȘARE( $c, a, n$ ) :  
for  $i = \overline{1, n}$  do  
  afișează  $a[i, c[i]]$ ;
```

1.3 Submulțimi

Propoziția 1.3.1 (de numărare a submulțimilor). Fie A o mulțime finită și $\mathcal{P}(A)$ mulțimea tuturor submulțimilor (părților) lui A . Atunci $\text{card}(\mathcal{P}(A)) = 2^{\text{card}(A)}$.

Demonstrație. Fie $A = \{a_1, a_2, \dots, a_n\}$, $n \in \mathbb{N}^*$. Notăm $\{1, 2\}^n = \underbrace{\{1, 2\} \times \dots \times \{1, 2\}}_{\text{de } n \text{ ori}}$.

Definim funcțiile $\alpha : \mathcal{P}(A) \rightarrow \{1, 2\}^n$ și $\beta : \{1, 2\}^n \rightarrow \mathcal{P}(A)$ prin:

- $\forall B \in \mathcal{P}(A)$, $\alpha(B) = (c_1, c_2, \dots, c_n)$, unde $c_i = \begin{cases} 1, & \text{dacă } a_i \in B, \\ 2, & \text{dacă } a_i \notin B; \end{cases}$
- $\forall (c_1, c_2, \dots, c_n) \in \{1, 2\}^n$, $\beta(c_1, c_2, \dots, c_n) = \{a_i | c_i = 1, i \in \{1, \dots, n\}\}$.

Funcțiile α și β sunt inverse una celeilalte, deci sunt bijective și $\text{card}(\mathcal{P}(A)) = \text{card}(\{1, 2\}^n) = 2^n$. \square

Exemplul 1.3.1. Pentru $A = \{a, b, c\}$, corespondența submulțimi \leftrightarrow produs cartezian din demonstrația anterioară este redată în următorul tabel:

$(c_1, c_2, c_3) \in \{1, 2\}^3$	$B \in \mathcal{P}(A)$
(1,1,1)	$\{a, b, c\}$
(1,1,2)	$\{a, b\}$
(1,2,1)	$\{a, c\}$
(1,2,2)	$\{a\}$
(2,1,1)	$\{b, c\}$
(2,1,2)	$\{b\}$
(2,2,1)	$\{c\}$
(2,2,2)	\emptyset

Deci A are $2^3 = 8$ submulțimi.

Algoritmul 1.3.1 (de generare a submulțimilor). Fie mulțimea $A = \{a_1, a_2, \dots, a_n\}$. Conform demonstrației anterioare, putem genera produsul cartezian $\{1, 2\}^n$ cu Algoritmul 1.2.1 înlocuind afișarea elementelor (c_1, \dots, c_n) cu afișarea submulțimilor corespunzătoare

$$B = \{a_i | c_i = 1, i \in \{1, \dots, n\}\}.$$

Obținem următorul algoritm descris în pseudocod.

```

SUBMULTIMI( $a, n$ ) :
  for  $i = \overline{1, n}$  do  $c[i] \leftarrow 1$ ;
  AFISARE( $c, a, n$ );
  repeat
     $k \leftarrow n$ ;
    while ( $c[k] = 2$ ) and ( $k > 0$ ) do  $k \leftarrow k - 1$ ;
    if ( $k > 0$ ) then
       $c[k] \leftarrow 2$ ;
      for  $i = \overline{k+1, n}$  do  $c[i] \leftarrow 1$ ;
      AFISARE( $c, a, n$ );
  while ( $k > 0$ );

```

unde funcția de afișare este

```

AFISARE( $c, a, n$ ) :
  for  $i = \overline{1, n}$  do
    if  $c[i] = 1$  then afișează  $a[i]$ ;

```

1.4 Aranjamente cu repetiție

Propoziția 1.4.1 (de numărare a aranjamentelor cu repetiție). Fie $m, n \in \mathbb{N}$. Atunci numărul de cuvinte de lungime n peste un alfabet cu m litere este egal cu m^n .

Demonstrație. Fie $B = \{1, 2, \dots, m\}$ și $\mathcal{C} = \{(c_1, c_2, \dots, c_n) | c_i \in B \forall i\}$. Cum $\mathcal{C} = B^n$, conform Propoziției 1.2.1 avem $\text{card}(\mathcal{C}) = m^n$. \square

Definiția 1.4.1. Cuvintele numărate în propoziția anterioară se numesc **aranjamente cu repetiție** de m luate câte n . Prin abuz de limbaj, și numărul lor, adică m^n , se numește tot **aranjamente cu repetiție de m luate câte n** .

Exemplul 1.4.1. Pentru $m = 2$ și $n = 3$, aranjamente cu repetiție sunt, în ordine lexicografică:

$$(1, 1, 1), (1, 1, 2), (1, 2, 1), (1, 2, 2), (2, 1, 1), (2, 1, 2), (2, 2, 1), (2, 2, 2).$$

Deci avem $2^3 = 8$ aranjamente cu repetiție.

Algoritmul 1.4.1 (de generare a aranjamentelor cu repetiție, sub formă de cuvinte). Pentru mulțimea standard $B = \{1, 2, \dots, m\}$, conform demonstrației anterioare putem genera produsul cartezian B^n , cu Algoritmul 1.2.1, luând $m_1 = m_2 = \dots = m_n = m$.

Observația 1.4.1. Pentru o mulțime $A = \{a_1, a_2, \dots, a_n\}$ oarecare putem genera indicii (c_1, \dots, c_n) cu algoritmul anterior și afișa elementele corespunzătoare acestor indici $(a_{c_1}, \dots, a_{c_n})$.

1.5 Aranjamente

Propoziția 1.5.1 (de numărare a aranjamentelor). Fie $m, n \in \mathbb{N}$. Atunci numărul de cuvinte de lungime n cu litere distincte peste un alfabet cu m litere este egal cu $[m]_n$.

Demonstrație. Fie $B = \{1, 2, \dots, m\}$ și $\mathcal{C}_1 = \{(c_1, c_2, \dots, c_n) | c_i \in B \forall i, c_i \neq c_j \forall i \neq j\}$. Avem $\mathcal{C}_1 = \{(c_1, c_2, \dots, c_n) | c_1 \in \{1, \dots, m\}, c_2 \in \{1, \dots, m\} \setminus \{c_1\}, c_3 \in \{1, \dots, m\} \setminus \{c_1, c_2\}, \dots, c_n \in \{1, \dots, m\} \setminus \{c_1, \dots, c_{n-1}\}\}$, deci $\text{card}(\mathcal{C}_1) = m(m-1)(m-2) \dots (m-n+1) = [m]_n$. \square

Definiția 1.5.1. Cuvintele numărate în propoziția anterioară se numesc **aranjamente (fără repetiție)** de m luate câte n . De asemenea și numărul lor, adică $[m]_n$, se numește tot **aranjamente (fără repetiție) de m luate câte n** și se mai notează cu A_m^n .

Observația 1.5.1. Pentru $n > m$ avem $[m]_n = m \dots (m - m) \dots (m - n + 1) = 0$.

Exemplul 1.5.1. Pentru $m = 4$ și $n = 2$, aranjamentele sunt, în ordine lexicografică:

$$(1, 2), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4), (3, 1), (3, 2), (3, 4), (4, 1), (4, 2), (4, 3).$$

Deci avem $[4]_2 = 4 \cdot 3 = 12$ aranjamente.

Un algoritm pentru generarea aranjamentelor va fi prezentat în Secțiunea 1.7.

1.6 Permutări

Propoziția 1.6.1 (de numărare a permutărilor). Fie $n \in \mathbb{N}$. Atunci numărul de cuvinte ce conțin exact o dată fiecare literă a unui alfabet cu n litere este egal cu $n!$, unde

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n \text{ (} n \text{ factorial)}, 0! = 1.$$

Demonstrație. Luăm $m = n$ în Propoziția 1.5.1 și folosim egalitatea $[n]_n = n(n - 1) \dots 1 = n!$. \square

Definiția 1.6.1. Cuvintele numărate în propoziția anterioară se numesc **permutări (fără repetiție)** de ordinul n . De asemenea și numărul lor, adică $n!$, se numește tot **permutări (fără repetiție) de n** .

Exemplul 1.6.1. Pentru $n = 3$, permutările mulțimii standard $\{1, 2, 3\}$ sunt, în ordine lexicografică:

$$(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1).$$

Deci avem $3! = 1 \cdot 2 \cdot 3 = 6$ permutări.

Algoritmul 1.6.1 (de generare a permutărilor). Fie mulțimea standard $A = \{1, 2, \dots, n\}$. Vom utiliza din nou "regula următorului".

- Prima permutare, în ordine lexicografică, este

$$(p_1, p_2, \dots, p_n) = (1, 2, \dots, n).$$

- O permutare arbitrară (curentă)

$$(p_1, p_2, \dots, p_{k-1}, p_k, p_{k+1}, \dots, p_n)$$

are o permutare următoare dacă și numai dacă există un indice $k \in \{n - 1, \dots, 1\}$ astfel încât $p_k < p_{k+1}$. În acest caz, luând cel mai mare indice k cu această proprietate și cel mai mare indice j din $\{n, \dots, k + 1\}$ astfel încât $p_j > p_k$ (există, deoarece $p_{k+1} > p_k$), permutarea următoare, în ordine lexicografică, este

$$(p_1, p_2, \dots, p_{k-1}, p_j, p'_{k+1}, \dots, p'_j, \dots, p'_n),$$

unde vectorul $(p'_{k+1}, \dots, p'_j, \dots, p'_n)$ este obținut prin ordonarea crescătoare a elementelor rămase $(p_k, p_{k+1}, \dots, p_{j-1}, p_{j+1}, \dots, p_n)$. Cum aceste elemente formează vectorul ordonat descrescător

$$(p_{k+1}, \dots, p_{j-1}, p_k, p_{j+1}, \dots, p_n),$$

rezultă că vectorul $(p'_{k+1}, \dots, p'_j, \dots, p'_n)$ este răsturnatul acestuia și deci poate fi obținut din acesta, de exemplu, prin interschimbări între termenii situați la egală distanță față de mijloc.

De exemplu, pentru permutarea curentă

$$(2, 7, \underline{4}, 8, 6, \underline{5}, 3, 1)$$

avem $k = 3$ ($p_3 = 4 < p_4 = 8$), $j = 6$ ($p_j = 5 > p_k = 4$), deci permutarea următoare se obține interschimbând întâi $p_k = p_3 = 4$ cu $p_j = p_6 = 5$, apoi răsturnând subvectorul $(8, 6, 4, 3, 1)$ dintre pozițiile $k + 1$ și n (de exemplu prin interschimbările $8 \leftrightarrow 1, 6 \leftrightarrow 3$), adică această permutare următoare este

$$(2, 7, 5, 1, 3, 4, 6, 8).$$

În pseudocod, algoritmul poate fi descris sub forma

```

PERMUTĂRI( $n$ ):
for  $i = \overline{1, n}$  do  $p[i] \leftarrow i$ ;
AFISARE( $p, n$ );
repeat
     $k \leftarrow n - 1$ ;
    while ( $p[k] \geq p[k + 1]$ ) and ( $k > 0$ ) do  $k \leftarrow k - 1$ ;
    if ( $k > 0$ ) then
         $j \leftarrow n$ ;
        while ( $p[j] \leq p[k]$ ) do  $j \leftarrow j - 1$ ;
         $p[k] \leftrightarrow p[j]$ ;
        for  $i = \overline{1, \lfloor \frac{n-k}{2} \rfloor}$  do  $p[k + i] \leftrightarrow p[n - i + 1]$ ;
        AFISARE( $p, n$ );
while ( $k > 0$ );

```

unde \leftrightarrow reprezintă instrucțiunea de interschimbare, iar $\lfloor x \rfloor$ reprezintă partea întreagă (inferioară) a numărului $x \in \mathbb{R}$.

Observația 1.6.1. Pentru generarea permutărilor unei mulțimi arbitrare $A = \{a_1, a_2, \dots, a_n\}$ înlocuim afișarea indicilor (p_1, \dots, p_n) cu afișarea elementelor corespunzătoare $(a_{p_1}, \dots, a_{p_n})$.

1.7 Combinări

Propoziția 1.7.1 (de numărare a combinațiilor). Fie $m, n \in \mathbb{N}$. Atunci

$$\begin{aligned}
 & \text{numărul de cuvinte strict crescătoare de lungime } n \text{ peste un alfabet (ordonat) cu } m \text{ litere} \\
 &= \text{numărul de submulțimi cu } n \text{ elemente ale unei mulțimi cu } m \text{ elemente} \\
 &= \frac{[m]_n}{n!} \stackrel{\text{def}}{=} \binom{m}{n}.
 \end{aligned}$$

Demonstrație. Fie $B = \{1, 2, \dots, m\}$. Notăm mulțimile din enunț astfel:

$$\mathcal{C}_3 = \{(c_1, c_2, \dots, c_n) \mid c_i \in B \forall i, c_i < c_{i+1} \forall i\}, \mathcal{S}_3 = \{S \mid S \subseteq B, \text{card}(S) = n\}.$$

Între aceste mulțimi definim funcțiile $\mu : \mathcal{C}_3 \rightarrow \mathcal{S}_3$, $\nu : \mathcal{S}_3 \rightarrow \mathcal{C}_3$ prin:

- $\forall (c_1, c_2, \dots, c_n) \in \mathcal{C}_3$, $\mu(c_1, c_2, \dots, c_n) = \{c_1, c_2, \dots, c_n\}$;
- $\forall \{c_1, c_2, \dots, c_n\} \in \mathcal{S}_3$ cu $c_1 < c_2 < \dots < c_n$, $\nu(\{c_1, c_2, \dots, c_n\}) = (c_1, c_2, \dots, c_n)$.

Aceste funcții sunt inverse una celeilalte, deci sunt bijective și astfel $\text{card}(\mathcal{C}_3) = \text{card}(\mathcal{S}_3)$.

Permutând fiecare combinație $(c_1, c_2, \dots, c_n) \in \mathcal{C}_3$ în toate cele $n!$ moduri posibile, obținem fără repetare toate aranjamentele din mulțimea $\mathcal{C}_1 = \{(a_1, a_2, \dots, a_n) \mid a_i \in B \forall i, a_i \neq a_j \forall i \neq j\}$.

Deci $\text{card}(\mathcal{C}_3) \cdot n! = \text{card}(\mathcal{C}_1)$. Conform Propoziției 1.5.1, $\text{card}(\mathcal{C}_1) = [m]_n$, deci $\text{card}(\mathcal{C}_3) = \frac{[m]_n}{n!}$. \square

Definiția 1.7.1. Oricare obiecte din cele 2 tipuri numărate în propoziția anterioară se numesc **combinări (fără repetiție)** de m luate câte n . De asemenea și numărul lor, adică $\binom{m}{n}$, se numește tot **combinări (fără repetiție)** de m luate câte n și se mai notează cu C_m^n .

Observația 1.7.1. Pentru $n > m$ avem $\binom{m}{n} = 0$, deoarece $[m]_n = 0$.

Pentru $n \leq m$, deoarece $[m]_n = m(m-1)\dots(m-n+1) = \frac{m!}{(m-n)!}$ avem $\binom{m}{n} = \frac{m!}{n!(m-n)!}$.

Exemplul 1.7.1. Pentru $m = 5$ și $n = 3$, corespondențele din demonstrația anterioară sunt redată în următorul tabel:

$(c_1, c_2, c_3) \in \mathcal{C}_3$	$S \in \mathcal{S}_3$
(1,2,3)	{1, 2, 3}
(1,2,4)	{1, 2, 4}
(1,2,5)	{1, 2, 5}
(1,3,4)	{1, 3, 4}
(1,3,5)	{1, 3, 5}
(1,4,5)	{1, 4, 5}
(2,3,4)	{2, 3, 4}
(2,3,5)	{2, 3, 5}
(2,4,5)	{2, 4, 5}
(3,4,5)	{3, 4, 5}.

Deci avem $\binom{5}{3} = \frac{[5]_3}{3!} = \frac{5 \cdot 4 \cdot 3}{1 \cdot 2 \cdot 3} = 10$ combinări.

Algoritmul 1.7.1 (de generare a combinărilor). Fie mulțimea standard $B = \{1, 2, \dots, m\}$ și $n \leq m$. Vom utiliza din nou "regula următorului".

- Prima combinare (de m luate câte n), în ordine lexicografică, este

$$(c_1, c_2, \dots, c_n) = (1, 2, \dots, n).$$

- O combinare arbitrară (curentă)

$$(c_1, c_2, \dots, c_{k-1}, c_k, c_{k+1}, \dots, c_n)$$

are o combinare următoare dacă și numai dacă există un indice $k \in \{n, \dots, 1\}$ astfel încât $c_k < m - n + k$. În acest caz, luând cel mai mare indice k cu această proprietate, combinarea următoare, în ordine lexicografică, este

$$(c_1, c_2, \dots, c_{k-1}, c_k + 1, c_k + 2, \dots, c_k + 1 + n - k).$$

De exemplu, pentru $m = 8$, $n = 6$ și combinarea curentă $(1, \underline{2}, 5, 6, 7, 8)$ avem $k = 2$ ($c_2 = 2 < m - n + k = 8 - 6 + 2$), deci combinarea următoare este $(1, 3, 4, 5, 6, 7)$.

În pseudocod, algoritmul poate fi descris sub forma

```

COMBINĂRI( $m, n$ ) :
for  $i = \overline{1, n}$  do  $c[i] \leftarrow i$ ;
AFISARE( $c, n$ );
repeat
     $k \leftarrow n$ ;
    while ( $c[k] = m - n + k$ ) and ( $k > 0$ ) do  $k \leftarrow k - 1$ ;
    if ( $k > 0$ ) then
         $c[k] \leftarrow c[k] + 1$ ;
        for  $i = \overline{k+1, n}$  do  $c[i] \leftarrow c[i - 1] + 1$ ;
        AFISARE( $c, n$ );
while ( $k > 0$ );

```

Algoritmul 1.7.2 (de generare a aranjamentelor). Fie mulțimea standard $B = \{1, 2, \dots, m\}$ și $n \leq m$. Conform demonstrației anterioare, putem genera aranjamentele de m luate câte n prin generarea combinațiilor și permutarea fiecărei combinații. Folosind Algoritmii 1.7.1 și 1.6.1 obținem următoarea descriere în pseudocod.

```

ARANJAMENTE( $m, n$ ) :
for  $i = \overline{1, n}$  do  $c[i] \leftarrow i$ ;
PERMUTĂRI( $n$ );
repeat
     $k \leftarrow n$ ;
    while ( $c[k] = m - n + k$ ) and ( $k > 0$ ) do  $k \leftarrow k - 1$ ;
    if ( $k > 0$ ) then
         $c[k] \leftarrow c[k] + 1$ ;
        for  $i = \overline{k+1, n}$  do  $c[i] \leftarrow c[i - 1] + 1$ ;
        PERMUTĂRI( $n$ );
while ( $k > 0$ );

```

unde **PERMUTĂRI**(n) este funcția din Algoritmul 1.6.1 înlocuind funcția de afișare cu

```

AFISARE( $c, p, n$ ) :
for  $i = \overline{1, n}$  do
    afișează  $c[p[i]]$ ;

```

Exemplul 1.7.2. Pentru $m = 4$ și $n = 3$, corespondența *aranjamente* \leftrightarrow *permutările combinațiilor* din demonstrația anterioară și din algoritmul anterior este redată în următorul tabel:

Combinaire	Permutare	Aranjament
(1,2,3)	(1,2,3)	(1,2,3)
	(1,3,2)	(1,3,2)
	(2,1,3)	(2,1,3)
	(2,3,1)	(2,3,1)
	(3,1,2)	(3,1,2)
	(3,2,1)	(3,2,1)
(1,2,4)	(1,2,3)	(1,2,4)
	(1,3,2)	(1,4,2)
	(2,1,3)	(2,1,4)
	(2,3,1)	(2,4,1)
	(3,1,2)	(4,1,2)
	(3,2,1)	(4,2,1)
(1,3,4)	(1,2,3)	(1,3,4)
	(1,3,2)	(1,4,3)
	(2,1,3)	(3,1,4)
	(2,3,1)	(3,4,1)
	(3,1,2)	(4,1,3)
	(3,2,1)	(4,3,1)
(2,3,4)	(1,2,3)	(2,3,4)
	(1,3,2)	(2,4,3)
	(2,1,3)	(3,2,4)
	(2,3,1)	(3,4,2)
	(3,1,2)	(4,2,3)
	(3,2,1)	(4,3,2)

Deci avem $\binom{4}{3} \cdot 3! = [4]_3 = 4 \cdot 3 \cdot 2 = 24$ aranjamente.

Observația 1.7.2. Analog Observației 1.6.1, algoritmi anteriori pot fi adaptați pentru generarea combinărilor și aranjamentelor pentru mulțimi oarecare.

1.8 Combinări cu repetiție

Propoziția 1.8.1 (de numărare a combinărilor cu repetiție). Fie $m, n \in \mathbb{N}$. Atunci numărul de cuvinte crescătoare de lungime n peste un alfabet (ordonat) cu m litere este egal cu $\frac{[m]^n}{n!} \stackrel{\text{def}}{=} \binom{m}{n}$.

Demonstrație. Fie $B = \{1, 2, \dots, m\}$, $A = \{1, 2, \dots, m + n - 1\}$,

$$\mathcal{C}_4 = \{(c_1, c_2, \dots, c_n) | c_i \in B \forall i, c_i \leq c_{i+1} \forall i\}, \mathcal{C}_3 = \{(d_1, d_2, \dots, d_n) | d_i \in A \forall i, d_i < d_{i+1} \forall i\}.$$

Definim corespondențele $\rho : \mathcal{C}_4 \rightarrow \mathcal{C}_3$, $\sigma : \mathcal{C}_3 \rightarrow \mathcal{C}_4$ astfel:

- $\rho(c_1, c_2, c_3, \dots, c_n) = (c_1, c_2 + 1, c_3 + 2, \dots, c_n + n - 1)$;
- $\sigma(d_1, d_2, d_3, \dots, d_n) = (d_1, d_2 - 1, d_3 - 2, \dots, d_n - n + 1)$.

Acestea sunt funcții inverse, deci utilizând Propoziția 1.7.1 avem

$$\text{card}(\mathcal{C}_4) = \text{card}(\mathcal{C}_3) = \binom{m+n-1}{n} = \frac{[m+n-1]_n}{n!} = \frac{[m]^n}{n!} = \binom{m}{n}. \quad \square$$

Definiția 1.8.1. Cuvintele numărate în propoziția anterioară se numesc **combinări cu repetiție** de m luate câte n . De asemenea și numărul lor, adică $\binom{m}{n}$, se numește tot **combinări cu repetiție de m luate câte n** .

Exemplul 1.8.1. Pentru $m = 3$ și $n = 4$, combinările cu repetiție sunt, în ordine lexicografică:

$$(1, 1, 1, 1), (1, 1, 1, 2), (1, 1, 1, 3), (1, 1, 2, 2), (1, 1, 2, 3), (1, 1, 3, 3), (1, 2, 2, 2), (1, 2, 2, 3), \\ (1, 2, 3, 3), (1, 3, 3, 3), (2, 2, 2, 2), (2, 2, 2, 3), (2, 2, 3, 3), (2, 3, 3, 3), (3, 3, 3, 3).$$

$$\text{Deci avem } \binom{3}{4} = \frac{[3]^4}{4!} = \frac{3 \cdot 4 \cdot 5 \cdot 6}{1 \cdot 2 \cdot 3 \cdot 4} = 15 \text{ combinări cu repetiție.}$$

Algoritmul 1.8.1 (de generare a combinărilor cu repetiție). Fie mulțimea standard $B = \{1, 2, \dots, m\}$. Vom utiliza din nou "regula următorului".

- Prima combinare cu repetiție (de m luate câte n), în ordine lexicografică, este

$$(c_1, c_2, \dots, c_n) = (1, 1, \dots, 1).$$

- O combinare cu repetiție arbitrară (curentă)

$$(c_1, c_2, \dots, c_{k-1}, c_k, c_{k+1}, \dots, c_n)$$

are o combinare cu repetiție următoare dacă și numai dacă există un indice $k \in \{n, \dots, 1\}$ astfel încât $c_k < m$. În acest caz, luând cel mai mare indice k cu această proprietate, combinarea cu repetiție următoare, în ordine lexicografică, este

$$(c_1, c_2, \dots, c_{k-1}, c_k + 1, c_k + 1, \dots, c_k + 1).$$

De exemplu, pentru $m = 8$, $n = 6$ și combinarea cu repetiție curentă $(2, 4, \underline{5}, 8, 8, 8)$ avem $k = 3$ ($c_3 = 5 < m = 8$) deci combinarea cu repetiție următoare este $(2, 4, 6, 6, 6, 6)$.

În pseudocod, algoritmul poate fi descris sub forma

```

COMBINĂRI_CU_REPETIȚIE( $m, n$ ) :
for  $i = \overline{1, n}$  do  $c[i] \leftarrow 1$ ;
AFISARE( $c, n$ );
repeat
     $k \leftarrow n$ ;
    while ( $c[k] = m$ ) and ( $k > 0$ ) do  $k \leftarrow k - 1$ ;
    if ( $k > 0$ ) then
         $c[k] \leftarrow c[k] + 1$ ;
        for  $i = \overline{k + 1, n}$  do  $c[i] \leftarrow c[k]$ ;
        AFISARE( $c, n$ );
while ( $k > 0$ );

```

Observația 1.8.1. Analog Observației 1.6.1, algoritmul anterior poate fi adaptat pentru generarea combinărilor cu repetiție pentru mulțimi oarecare.

1.9 Permutări cu repetiție

Propoziția 1.9.1 (de numărare a permutărilor cu repetiție). Fie $m, t_1, t_2, \dots, t_m \in \mathbb{N}$ și $n = t_1 + t_2 + \dots + t_m$. Atunci numărul de cuvinte de lungime n ce pot fi formate peste un alfabet cu m litere astfel încât litera numărul i să apară de exact t_i ori pentru orice $i \in \{1, \dots, m\}$ este egal cu

$$\frac{n!}{t_1! t_2! \dots t_m!} \stackrel{\text{def}}{=} \binom{n}{t_1, t_2, \dots, t_m}.$$

Demonstrație. Fie $A = \{1, 2, \dots, m\}$ și

$$\mathcal{C}_5 = \{(x_1, x_2, \dots, x_n) | x_i \in A \forall i, \text{card}(\{i | x_i = j\}) = t_j \forall j\}.$$

Numărăm cuvintele din \mathcal{C}_5 astfel:

- alegem cei t_1 indici (din totalul de n) ai literelor egale cu 1, rezultă $\binom{n}{t_1}$ moduri posibile;
- pentru fiecare alegere de mai sus, alegem cei t_2 indici (din restul de $n - t_1$) ai literelor egale cu 2, rezultă $\binom{n - t_1}{t_2}$ moduri posibile, deci obținem $\binom{n}{t_1} \binom{n - t_1}{t_2}$ moduri posibile de alegere a indicilor literelor 1 și 2;
- ...
- pentru fiecare alegere de mai sus, alegem cei t_m indici (din restul de $n - t_1 - \dots - t_{m-1}$) ai literelor egale cu m ; rezultă $\binom{n - t_1 - \dots - t_{m-1}}{t_m}$ moduri posibile, deci obținem

$$\binom{n}{t_1} \binom{n - t_1}{t_2} \dots \binom{n - t_1 - \dots - t_{m-1}}{t_m}$$

moduri posibile de alegere a tuturor indicilor literelor $1, 2, \dots, m$.

Astfel

$$\begin{aligned} \text{card}(\mathcal{C}_5) &= \binom{n}{t_1} \binom{n-t_1}{t_2} \cdots \binom{n-t_1-\dots-t_{m-1}}{t_m} \\ &= \frac{n!}{t_1!(n-t_1)!} \cdot \frac{(n-t_1)!}{t_2!(n-t_1-t_2)!} \cdots \frac{(n-t_1-\dots-t_{m-1})!}{t_m!(n-t_1-\dots-t_m)!} \\ &= \frac{n!}{t_1!t_2!\dots t_m!} \end{aligned}$$

(deoarece $(n-t_1-\dots-t_m)! = 0! = 1$).

□

Definiția 1.9.1. Cuvintele numărate în propoziția anterioară se numesc **permutări cu repetiție** (**anagrame**) de n luate câte t_1, t_2, \dots, t_m . De asemenea și numărul lor, adică $\binom{n}{t_1, t_2, \dots, t_m}$, se numește tot **permutări cu repetiție de n luate câte t_1, t_2, \dots, t_m** .

Observația 1.9.1. Luând $t_1 = t_2 = \dots = t_m = 1$ obținem $n = m$ și $\binom{n}{1, 1, \dots, 1} = n!$, deci permutările (fără repetiție) sunt un caz particular al permutărilor cu repetiție. Pe de altă parte, luând $m = 2$ obținem $\binom{n}{t_1, t_2} = \frac{n!}{t_1!t_2!}$, deci și combinările (fără repetiție) sunt un caz particular al permutărilor cu repetiție.

Exemplul 1.9.1. Pentru $m = 3$ și $t_1 = 2, t_2 = t_3 = 1$, deci $n = 4$, permutările cu repetiție sunt, în ordine lexicografică:

$$\begin{aligned} &(1, 1, 2, 3), (1, 1, 3, 2), (1, 2, 1, 3), (1, 2, 3, 1), (1, 3, 1, 2), (1, 3, 2, 1), \\ &(2, 1, 1, 3), (2, 1, 3, 1), (2, 3, 1, 1), (3, 1, 1, 2), (3, 1, 2, 1), (3, 2, 1, 1). \end{aligned}$$

Deci avem $\binom{4}{2, 1, 1} = \frac{4!}{2!1!1!} = 12$ permutări cu repetiție.

Algoritmul 1.9.1 (de generare a permutărilor cu repetiție). Fie mulțimea standard $B = \{1, 2, \dots, m\}$ și $t_1, t_2, \dots, t_m \in \mathbb{N}$, $n = t_1 + t_2 + \dots + t_m$. Vom utiliza din nou "regula următorului".

- Prima permutare cu repetiție (de n luate câte t_1, t_2, \dots, t_m), în ordine lexicografică, este

$$(p_1, p_2, \dots, p_n) = (\underbrace{1, 1, \dots, 1}_{\text{de } t_1 \text{ ori}}, \underbrace{2, 2, \dots, 2}_{\text{de } t_2 \text{ ori}}, \dots, \underbrace{m, m, \dots, m}_{\text{de } t_m \text{ ori}}).$$

- Existența și forma permutării cu repetiție următoare în ordine lexicografică pentru o permutare cu repetiție curentă arbitrară se determină exact ca la permutările fără repetiție (Algoritmul 1.6.1).

De exemplu, pentru permutarea cu repetiție curentă

$$(4, 4, 1, \underline{3}, 6, 5, \underline{\underline{5}}, 3, 2, 1)$$

avem $k = 4$ ($p_4 = 3 < p_5 = 6$), $j = 7$ ($p_j = 5 > p_k = 3$), deci permutarea cu repetiție următoare se obține interschimbând întâi $p_k = p_4 = 3$ cu $p_j = p_7 = 5$, apoi răsturnând subvectorul $(6, 5, 3, 3, 2, 1)$ dintre pozițiile $k + 1$ și n (de exemplu prin interschimbările $6 \leftrightarrow 1, 5 \leftrightarrow 2, 3 \leftrightarrow 3$), adică această permutare cu repetiție următoare este

$$(4, 4, 1, 5, 1, 2, 3, 3, 5, 6).$$

În pseudocod, algoritmul poate fi descris sub forma

```

PERMUTĂRI_CU_REPETIȚIE( $n, t, m$ ) :
 $n \leftarrow 0$ ;
for  $i = \overline{1, m}$  do  $n \leftarrow n + t[i]$ ;
 $i \leftarrow 0$ ;
for  $j = \overline{1, m}$  do
    for  $k = \overline{1, t[j]}$  do
         $i \leftarrow i + 1$ ;
         $p[i] \leftarrow j$ ;
AFISARE( $p, n$ );
repeat
     $k \leftarrow n - 1$ ;
    while ( $p[k] \geq p[k + 1]$ ) and ( $k > 0$ ) do  $k \leftarrow k - 1$ ;
    if ( $k > 0$ ) then
         $j \leftarrow n$ ;
        while ( $p[j] \leq p[k]$ ) do  $j \leftarrow j - 1$ ;
         $p[k] \leftrightarrow p[j]$ ;
        for  $i = \overline{1, \lceil \frac{n-k}{2} \rceil}$  do  $p[k + i] \leftrightarrow p[n - i + 1]$ ;
        AFISARE( $p, n$ );
while ( $k > 0$ );

```

Observația 1.9.2. Analog Observației 1.6.1, algoritmul anterior poate fi ușor adaptat pentru generarea permutărilor cu repetiție pentru mulțimi arbitrare.

Tema 2

Partiții

2.1 Compuneri ale unui număr natural

Definiția 2.1.1. Fie $n, m \in \mathbb{N}$. O **compunere** a lui n este o scriere de forma

$$n = n_1 + n_2 + \cdots + n_m,$$

unde $n_1, n_2, \dots, n_m \in \mathbb{N}$ și contează ordinea dintre termenii n_1, n_2, \dots, n_m .

Propoziția 2.1.1 (de numărare a compunerilor unui număr natural). Fie $n, m \in \mathbb{N}$. Atunci:

- a) numărul de compuneri ale lui n cu m termeni este egal cu $\binom{m}{n}$;
- b) numărul de compuneri ale lui n cu m termeni nenuli este egal cu $\binom{n-1}{m-1}$.

Demonstrație. a) Fie mulțimile

$$\mathcal{N} = \{(n_1, n_2, \dots, n_m) | n_i \in \mathbb{N} \forall i, n_1 + n_2 + \cdots + n_m = n\},$$

$$\mathcal{C}_4 = \{(c_1, c_2, \dots, c_n) | c_i \in \{1, 2, \dots, m\} \forall i, c_i \leq c_{i+1} \forall i\}.$$

Definim corespondențele $\alpha : \mathcal{N} \rightarrow \mathcal{C}_4$, $\beta : \mathcal{C}_4 \rightarrow \mathcal{N}$ prin:

- $\alpha(n_1, \dots, n_m) = (c_1, \dots, c_n)$, unde
$$\begin{cases} c_1 = \cdots = c_{n_1} = 1 \text{ (} n_1 \text{ litere)}, \\ c_{n_1+1} = \cdots = c_{n_1+n_2} = 2 \text{ (} n_2 \text{ litere)}, \\ \dots \\ c_{n_1+\dots+n_{m-1}+1} = \cdots = c_{n_1+\dots+n_{m-1}+n_m} = m \text{ (} n_m \text{ litere)}; \end{cases}$$
- $\beta(c_1, \dots, c_n) = (n_1, \dots, n_m)$, unde $n_i = \text{numărul de litere } i \text{ ale cuvântului } (c_1, \dots, c_n), \forall i$.

Acestea sunt funcții inverse, deci conform Propoziției 1.8.1 avem $\text{card}(\mathcal{N}) = \text{card}(\mathcal{C}_4) = \binom{m}{n}$.

- b) Fie mulțimile $\mathcal{N}_1 = \{(n_1, n_2, \dots, n_m) | n_i \in \mathbb{N}^* \forall i, n_1 + n_2 + \cdots + n_m = n\}$,

$$\mathcal{N}_2 = \{(t_1, t_2, \dots, t_m) | t_i \in \mathbb{N} \forall i, t_1 + t_2 + \cdots + t_m = n - m\}.$$

Definim corespondențele $\varphi : \mathcal{N}_1 \rightarrow \mathcal{N}_2$, $\psi : \mathcal{N}_2 \rightarrow \mathcal{N}_1$ prin:

- $\forall (n_1, \dots, n_m) \in \mathcal{N}_1, \varphi(n_1, \dots, n_m) = (n_1 - 1, \dots, n_m - 1)$;

- $\forall (t_1, \dots, t_m) \in \mathcal{N}_2, \psi(t_1, \dots, t_m) = (t_1 + 1, \dots, t_m + 1).$

Acestea sunt funcții inverse, deci conform punctului a) avem $\text{card}(\mathcal{N}_1) = \text{card}(\mathcal{N}_2) = \left(\binom{m}{n-m} \right) = \frac{[m]^{n-m}}{(n-m)!} = \frac{m(m+1) \dots (n-1)}{(n-m)!} = \frac{(n-1)!}{(m-1)!(n-m)!} = \binom{n-1}{m-1}.$ \square

Exemplul 2.1.1. Pentru $n = 4$ și $m = 3$ compunerile sunt

$$\begin{aligned} 4 &= 0 + 0 + 4 = 0 + 1 + 3 = 0 + 2 + 2 = 0 + 3 + 1 = 0 + 4 + 0 \\ &= 1 + 0 + 3 = 1 + 1 + 2 = 1 + 2 + 1 = 1 + 3 + 0 = 2 + 0 + 2 \\ &= 2 + 1 + 1 = 2 + 2 + 0 = 3 + 0 + 1 = 3 + 1 + 0 = 4 + 0 + 0. \end{aligned}$$

Deci avem $\left(\binom{3}{4} \right) = \frac{3 \cdot 4 \cdot 5 \cdot 6}{1 \cdot 2 \cdot 3 \cdot 4} = 15$ compuneri, dintre care $\binom{4-1}{3-1} = \binom{3}{2} = 3$ compuneri cu termenii nenuli.

Algoritmul 2.1.1 (de generare a compunerilor unui număr natural). Fie $n, m \in \mathbb{N}^*$. Vom utiliza din nou "regula următorului".

- Prima compunere a lui n cu m termeni, în ordine lexicografică, este

$$n = 0 + 0 + \dots + 0 + n.$$

- O compunere curentă arbitrară

$$n = t_1 + t_2 + \dots + t_{k-1} + t_k + t_{k+1} + \dots + t_m$$

are o compunere următoare dacă și numai dacă există un indice $k \in \{m, \dots, 2\}$ astfel încât $t_k > 0$. În acest caz, luând cel mai mare indice k cu această proprietate, compunerea următoare, în ordine lexicografică, este

$$n = t_1 + t_2 + \dots + t_{k-2} + (t_{k-1} + 1) + 0 + \dots + 0 + (t_k - 1).$$

De exemplu, următoarea compunere după

$$10 = 2 + 0 + 5 + \underline{3} + 0 + 0$$

este $10 = 2 + 0 + 6 + 0 + 0 + 2$ (deoarece $t_k = t_4 = 3$).

În pseudocod, algoritmul poate fi descris sub forma

COMPUNERI(n, m):

for $i = \overline{1, m-1}$ **do** $t[i] \leftarrow 0$;

$t[m] \leftarrow n$;

AFISARE(t, m);

repeat

$k \leftarrow m$;

while ($t[k] = 0$) *and* ($k > 1$) **do** $k \leftarrow k - 1$;

if ($k > 1$) **then**

$t[k-1] \leftarrow t[k-1] + 1$;

$t[m] \leftarrow t[m] - 1$;

if ($k < m$) **then** $t[k] \leftarrow 0$;

AFISARE(t, m);

while ($k > 1$);

unde funcția de afișare este

```
AFISARE( $t, m$ ):
for  $i = \overline{1, m}$  do
  | afișează  $t[i]$ ;
```

Algoritmul 2.1.2 (de generare a compunerilor cu termeni nenuli). Analog algoritmului anterior se obține următorul algoritm pentru generarea compunerilor lui n cu m termeni nenuli, $m \leq n$.

```
COMPUNERI_TERMENI_NENULI( $n, m$ ):
for  $i = \overline{1, m-1}$  do  $t[i] \leftarrow 1$ ;
 $t[m] \leftarrow n - m + 1$ ;
AFISARE( $t, m$ );
repeat
  |  $k \leftarrow m$ ;
  | while ( $t[k] = 1$ ) and ( $k > 1$ ) do  $k \leftarrow k - 1$ ;
  | if ( $k > 1$ ) then
  |   |  $t[k-1] \leftarrow t[k-1] + 1$ ;
  |   |  $t[m] \leftarrow t[k] - 1$ ;
  |   | if ( $k < m$ ) then  $t[k] \leftarrow 1$ ;
  |   | AFISARE( $t, m$ );
while ( $k > 1$ );
```

2.2 Partiții ale unui număr natural

Definiția 2.2.1. O *partiție (descompunere)* a numărului $n \in \mathbb{N}^*$ este o scriere de forma

$$n = n_1 + n_2 + \cdots + n_k,$$

unde $n_1, n_2, \dots, n_k \in \mathbb{N}^*$ ($k \in \mathbb{N}^*$) și nu contează ordinea dintre termenii n_1, n_2, \dots, n_k .

Observația 2.2.1. Deoarece într-o partiție ca mai sus nu contează ordinea dintre termeni, putem presupune că aceștia sunt scriși în ordine crescătoare.

Definiția 2.2.2. Fie $n, k \in \mathbb{N}^*$. Notăm cu $P(n, k)$ numărul de partiții ale lui n cu k termeni, iar cu $P(n)$ numărul tuturor partițiilor lui n .

Exemplul 2.2.1. Numărul $n = 6$ are partițiile

$$\begin{aligned} 6 &= 6 = 1 + 5 = 2 + 4 = 3 + 3 = 1 + 1 + 4 = 1 + 2 + 3 = 2 + 2 + 2 \\ &= 1 + 1 + 1 + 3 = 1 + 1 + 2 + 2 = 1 + 1 + 1 + 1 + 2 = 1 + 1 + 1 + 1 + 1 + 1, \end{aligned}$$

deci $P(6, 1) = 1$, $P(6, 2) = 3$, $P(6, 3) = 3$, $P(6, 4) = 2$, $P(6, 5) = 1$, $P(6, 6) = 1$ și $P(6) = 11$.

Observația 2.2.2. Avem $P(n) = P(n, 1) + P(n, 2) + \cdots + P(n, n)$, $\forall n \in \mathbb{N}^*$.

Propoziția 2.2.1 (relația de recurență a numerelor $P(n, k)$). Pentru orice $n \in \mathbb{N}^*$ și orice $k \in \{1, \dots, n-1\}$ avem

$$P(n, k) = P(n-k, 1) + P(n-k, 2) + \cdots + P(n-k, k).$$

Demonstrație. Pentru orice $n \in \mathbb{N}^*$ și orice $k \in \{1, \dots, n-1\}$ notăm

$$\mathcal{P}(n, k) = \{(n_1, \dots, n_k) | n_i \in \mathbb{N}^* \forall i, n_1 \leq \dots \leq n_k, n_1 + \dots + n_k = n\}.$$

Definim corespondențele $\alpha : \mathcal{P}(n, k) \rightarrow \bigcup_{i=1}^k \mathcal{P}(n-k, i)$ și $\beta : \bigcup_{i=1}^k \mathcal{P}(n-k, i) \rightarrow \mathcal{P}(n, k)$ prin:

- $\forall (n_1, \dots, n_k) \in \mathcal{P}(n, k), \alpha(n_1, \dots, n_k) = (n_j - 1, \dots, n_k - 1)$, unde $j = \min\{i | n_i \geq 2, i \in \{1, \dots, k\}\}$ (există j , deoarece $k \leq n-1$);
- $\forall i \in \{1, \dots, k\}, \forall (n_1, \dots, n_i) \in \mathcal{P}(n-k, i), \beta(n_1, \dots, n_i) = (\underbrace{1, \dots, 1}_{\text{de } k-i \text{ ori}}, n_1 + 1, \dots, n_i + 1)$.

Interpretarea acestor funcții este următoarea: aplicarea funcției α unei partiții a lui n cu k termeni constă în micșorarea cu 1 a fiecărui termen și eliminarea termenilor care astfel devin egali cu zero, obținându-se o partiție a lui $n-k$ cu cel mult k termeni. Reciproc, aplicarea funcției β unei partiții a lui $n-k$ cu cel mult k termeni constă în mărirea cu 1 a fiecărui termen și adăugarea de termeni egali cu 1 pentru a obține k termeni, astfel obținându-se o partiție a lui n cu k termeni.

Funcțiile α și β sunt bine definite și inverse una celeilalte, deci $\text{card}(\mathcal{P}(n, k)) = \text{card}\left(\bigcup_{i=1}^k \mathcal{P}(n-k, i)\right)$.

Cum mulțimile $\mathcal{P}(n-k, 1), \dots, \mathcal{P}(n-k, k)$ sunt evident disjuncte două câte două, rezultă că $P(n, k) = \sum_{i=1}^k P(n-k, i)$. □

Observația 2.2.3. Relația de recurență din Propoziția 2.2.1, împreună cu condițiile inițiale evidente $P(n, 1) = P(n, n) = 1, P(n, k) = 0 \forall k > n$ permit calculul tuturor numerelor $P(n, k)$, deci și al numerelor $P(n)$, conform Corolarului 2.2.2. De exemplu, tabelul numerelor $P(n, k)$ și $P(n)$ pentru $n \leq 7$ (și $k \leq 7$) este:

$P(n, k)$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$P(n)$
$n = 1$	1	0	0	0	0	0	0	1
$n = 2$	1	1	0	0	0	0	0	2
$n = 3$	1	1	1	0	0	0	0	3
$n = 4$	1	2	1	1	0	0	0	5
$n = 5$	1	2	2	1	1	0	0	7
$n = 6$	1	3	3	2	1	1	0	11
$n = 7$	1	3	4	3	2	1	1	15

Numerele $P(n, k)$ nenule, aflate doar pe diagonala principală și sub această diagonală (adică $1 \leq k \leq n$) formează **triunghiul numerelor** $P(n, k)$.

Algoritmul 2.2.1 (de generare a partițiilor lui n cu k termeni). Fie $n, k \in \mathbb{N}^*, k \leq n$. Pentru generarea în ordine lexicografică a celor $P(n, k)$ partiții ale lui n cu k termeni

$$(t_1, t_2, \dots, t_k), n = t_1 + t_2 + \dots + t_k, t_i \in \mathbb{N}^* \forall i, t_1 \leq t_2 \leq \dots \leq t_k$$

folosim din nou ”regula următorului”.

- Prima partiție este $(\underbrace{1, \dots, 1}_{\text{de } k-1 \text{ ori}}, n-k+1)$.

- O partiție curentă arbitrară

$$(t_1, t_2, \dots, t_{j-1}, t_j, t_{j+1}, \dots, t_k)$$

are o partiție următoare dacă și numai dacă există un indice $j \in \{k-1, \dots, 1\}$ astfel încât $t_j \leq t_k - 2$. În acest caz, luând cel mai mare indice j cu această proprietate, următoarea partiție este

$$(t_1, t_2, \dots, t_{j-1}, t_j + 1, t_j + 1, \dots, t_j + 1, n - r),$$

unde $r = t_1 + t_2 + \dots + t_{j-1} + (t_j + 1) + (t_j + 1) + \dots + (t_j + 1)$.

De exemplu, următoarea partiție după

$$22 = 1 + 1 + \underline{2} + 4 + 4 + 5 + 5$$

este

$$22 = 1 + 1 + 3 + 3 + 3 + 3 + 8$$

(deoarece $t_j = t_3 = 2 \leq t_k - 2 = 5 - 2$).

Descrierea în pseudocod a algoritmului are forma

GENERARE_P(n, k):

for $i = \overline{1, k-1}$ **do** $t[i] \leftarrow 1$;

$t[k] \leftarrow n - k + 1$;

AFISARE(t, k);

repeat

$j \leftarrow k - 1$;

while ($t[j] > t[k] - 2$) *and* ($j > 0$) **do** $j \leftarrow j - 1$;

if ($j > 0$) **then**

$t[j] \leftarrow t[j] + 1$;

for $i = \overline{j+1, k-1}$ **do** $t[i] \leftarrow t[j]$;

$r \leftarrow 0$;

for $i = \overline{1, k-1}$ **do** $r \leftarrow r + t[i]$;

$t[k] \leftarrow n - r$;

AFISARE(t, k);

while ($j > 0$);

unde funcția de afișare este aceeași ca în Algoritmul 2.1.1.

2.3 Partiții ale unei mulțimi finite

Definiția 2.3.1. O *partiție a unei mulțimi nevide* A este o scriere de forma

$$A = A_1 \cup A_2 \cup \dots \cup A_k,$$

unde submulțimile (părțile, clasele) A_1, A_2, \dots, A_k ($k \in \mathbb{N}^*$) sunt nevide și disjuncte două câte două, și nu contează ordinea dintre aceste submulțimi. Considerăm că singura partiție a mulțimii vide este partiția cu zero submulțimi.

Definiția 2.3.2. Fie $n, k \in \mathbb{N}$.

a) **Numărul lui Stirling de speța a doua**, notat cu $S(n, k)$, reprezintă numărul de partiții ale unei mulțimi cu n elemente în k părți.

b) **Numărul lui Bell**, notat cu B_n , reprezintă numărul tuturor partițiilor unei mulțimi cu n elemente.

Exemplul 2.3.1. Mulțimea $A = \{a, b, c\}$ are partițiile

$$A = \{a, b, c\} = \{a, b\} \cup \{c\} = \{a, c\} \cup \{b\} = \{a\} \cup \{b, c\} = \{a\} \cup \{b\} \cup \{c\},$$

deci $S(3, 1) = 1$, $S(3, 2) = 3$, $S(3, 3) = 1$ și $B_3 = 5$.

Observația 2.3.1. Evident, avem $B_0 = 1$ și $B_n = S(n, 1) + S(n, 2) + \dots + S(n, n)$, $\forall n \in \mathbb{N}^*$.

Propoziția 2.3.1 (relația de recurență a numerelor $S(n, k)$).

$$S(n, k) = S(n - 1, k - 1) + kS(n - 1, k), \forall n, k \in \mathbb{N}^*.$$

Demonstrație. Notăm

$$\mathcal{S}(n, k) = \{\{A_1, \dots, A_k\} \mid A_i \neq \emptyset \forall i, A_i \cap A_j = \emptyset \forall i \neq j, A = A_1 \cup \dots \cup A_k\},$$

pentru orice $n, k \in \mathbb{N}^*$. Definim corespondențele

$$\begin{aligned} \alpha : \mathcal{S}(n, k) &\rightarrow \mathcal{S}(n - 1, k - 1) \cup \{1, \dots, k\} \times \mathcal{S}(n - 1, k), \\ \beta : \mathcal{S}(n - 1, k - 1) \cup \{1, \dots, k\} \times \mathcal{S}(n - 1, k) &\rightarrow \mathcal{S}(n, k), \end{aligned}$$

prin:

- $\forall \{A_1, \dots, A_k\} \in \mathcal{S}(n, k)$, $\alpha(\{A_1, \dots, A_k\}) =$

$$= \begin{cases} \{A_1, \dots, A_k\} \setminus \{A_i\}, & \text{dacă } n \in A_i \text{ și } A_i = \{n\}, \\ (i, \{A_1, \dots, A_i \setminus \{n\}, \dots, A_k\}), & \text{dacă } n \in A_i \text{ și } A_i \neq \{n\}; \end{cases}$$
- $\forall \{A_1, \dots, A_{k-1}\} \in \mathcal{S}(n - 1, k - 1)$, $\beta(\{A_1, \dots, A_{k-1}\}) = \{A_1, \dots, A_{k-1}, \{n\}\}$,
- $\forall (i, \{A_1, \dots, A_k\}) \in \{1, \dots, k\} \times \mathcal{S}(n - 1, k)$, $\beta(i, \{A_1, \dots, A_k\}) = \{A_1, \dots, A_i \cup \{n\}, \dots, A_k\}$.

Interpretarea acestor definiții este următoarea: aplicarea funcției α unei partiții a mulțimii $\{1, \dots, n\}$ în k părți constă în eliminarea elementului n , astfel obținându-se o partiție a mulțimii $\{1, \dots, n - 1\}$ în $k - 1$ sau în k părți, după cum n este sau nu singur într-o parte. Reciproc, aplicarea funcției β constă în adăugarea elementului n , fie singur într-o parte, fie la oricare din cele k părți.

Funcțiile α și β sunt bine definite și inverse una celeilalte, deci

$$\text{card}(\mathcal{S}(n, k)) = \text{card}(\mathcal{S}(n - 1, k - 1) \cup \{1, \dots, k\} \times \mathcal{S}(n - 1, k)),$$

adică $S(n, k) = S(n - 1, k - 1) + kS(n - 1, k)$. □

Observația 2.3.2. Relația de recurență de mai sus împreună cu condițiile inițiale evidente $S(0, 0) = 1$, $S(0, k) = 0 \forall k \geq 1$, $S(n, 0) = 0 \forall n \geq 1$ permit calculul tuturor numerelor $S(n, k)$, deci și al numerelor B_n , conform Corolarului 2.3.1. De exemplu, tabelul numerelor $S(n, k)$ și B_n pentru $n \leq 6$ (și $k \leq 6$) este:

$S(n, k)$	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	B_n
$n = 0$	1	0	0	0	0	0	0	1
$n = 1$	0	1	0	0	0	0	0	1
$n = 2$	0	1	1	0	0	0	0	2
$n = 3$	0	1	3	1	0	0	0	5
$n = 4$	0	1	7	6	1	0	0	15
$n = 5$	0	1	15	25	10	1	0	52
$n = 6$	0	1	31	90	65	15	1	203

Tema 3

Grafuri

Grafurile sunt modele matematice cu o gamă largă de aplicații. Această aplicabilitate a condus la dezvoltarea accelerată a teoriei grafurilor, atât din punct de vedere al rezultatelor teoretice cât și al algoritmicii, grafurile impunându-se drept modele de bază în informatică, în special în teoria structurilor de date și a analizei algoritmilor.

3.1 Definiții generale

Definiția 3.1.1. *Un graf neorientat (graf, pseudograf, graf general) este o pereche $G = (V, E)$ unde:*

- *V este o mulțime finită și nevidă, elementele sale numindu-se **nodurile** (**vârfurile**, **punctele**) grafului G ;*
- *E este o colecție (mulțime multiplă, multiset) finită de perechi neordonate, posibil egale, de noduri, elementele sale numindu-se **muchii** (**legăturile directe**, **liniile**) grafului G .*

Observația 3.1.1. Într-o pereche neordonată, notată $[x, y]$, nu contează ordinea dintre elemente, adică $[x, y] = [y, x]$.

Definiția 3.1.2. *Un graf orientat (digraf, pseudodigraf) este o pereche $G = (V, E)$ unde:*

- *V este o mulțime finită și nevidă, elementele sale numindu-se **vârfurile** (**nodurile**, **punctele**) grafului G ;*
- *E este o colecție (mulțime multiplă, multiset) finită de perechi ordonate de vârfuri, elementele sale numindu-se **arcele** (**legăturile directe ale**) grafului G .*

Observația 3.1.2. Într-o pereche ordonată, notată (x, y) , contează ordinea dintre elemente, adică $(x, y) \neq (y, x)$ pentru $x \neq y$.

Definiția 3.1.3. *Numărul de noduri ale unui graf (neorientat sau orientat) se numește **ordinul grafului**, iar numărul de muchii sau arce se numește **dimensiunea grafului**.*

Definiția 3.1.4. a) *Dacă $e = [x, y]$ este o muchie a unui graf neorientat, atunci nodurile x și y se numesc **extremitățile** muchiei e și spunem că muchia e este **incidentă** cu nodurile x și y .*

- b) Dacă $e = (x, y)$ este un arc al unui graf orientat, atunci nodul x se numește **extremitatea inițială** iar nodul y se numește **extremitatea finală** a arcului e și spunem că arcul e este **incident** cu x **spre exterior** și cu y **spre interior**.
- c) O **bucă** a unui graf (neorientat sau orientat) este o muchie sau un arc având extremitățile egale (adică o muchie de forma $[x, x]$, respectiv un arc de forma (x, x)).
- d) Două noduri x și y ale unui graf (neorientat sau orientat) se numesc **adiacente (vecine)** dacă există o muchie sau un arc incident cu x și cu y (adică o muchie de forma $[x, y]$, respectiv un arc de forma (x, y) sau de forma (y, x)).

Definiția 3.1.5. Dacă în colecția (mulțimea multiplă) de muchii sau arce a unui graf (neorientat sau orientat) există două sau mai multe elemente egale (și aflate pe poziții diferite), atunci acestea se numesc **muchii** sau **arce multiple**.

Definiția 3.1.6. Un graf (neorientat sau orientat) fără bucle se numește **multigraf (neorientat, respectiv orientat)**.

Definiția 3.1.7. Un graf (neorientat sau orientat) se numește **simplu (sau strict)** dacă nu conține nici bucle și nici muchii sau arce multiple.

Observația 3.1.3. a) Un **graf neorientat simplu** este o pereche $G = (V, E)$, unde V este o mulțime finită și nevidă (de elemente numite noduri) iar $E \subseteq \mathcal{P}_2(V)$ este o mulțime finită (de elemente numite muchii), unde

$$\mathcal{P}_2(V) = \{\{x, y\} | x, y \in V, x \neq y\}$$

(mulțimea tuturor submulțimilor cu două elemente ale lui V).

b) Un **graf orientat simplu** este o pereche $G = (V, E)$, unde V este o mulțime finită și nevidă (de elemente numite noduri) iar $E \subseteq V \times V \setminus \{(x, x) | x \in V\}$ este o mulțime finită (de elemente numite arce).

Definiția 3.1.8. Fie $G = (V, E)$ un graf (orientat sau neorientat). O **reprezentare grafică** a lui G se obține reprezentând nodurile sale prin puncte distincte (în plan sau pe o altă suprafață dată), iar muchiile $[x, y]$ sau arcele (x, y) prin segmente (de curbă continuă) distincte neorientate, respectiv orientate, de la punctul ce reprezintă nodul x până la punctul ce reprezintă nodul y .

Observația 3.1.4. În reprezentările grafice, nodurile unui graf sunt reprezentate adesea încadrate în cercurițe (sau pătrățele).

Exemplul 3.1.1. Graful neorientat $G = (V, E)$, cu $V = \{1, 2, 3, 4, 5, 6\}$ și $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$, unde $e_1 = [1, 2]$, $e_2 = [1, 4]$, $e_3 = [2, 2]$, $e_4 = [2, 5]$, $e_5 = [3, 6]$, $e_6 = [3, 6]$, $e_7 = [4, 5]$, $e_8 = [4, 5]$, $e_9 = [4, 5]$ (E este o mulțime multiplă!) are reprezentarea grafică din Figurile 3.1.1 (cu nodurile reprezentate prin puncte) și 3.1.2 (cu nodurile reprezentate în cercurițe).

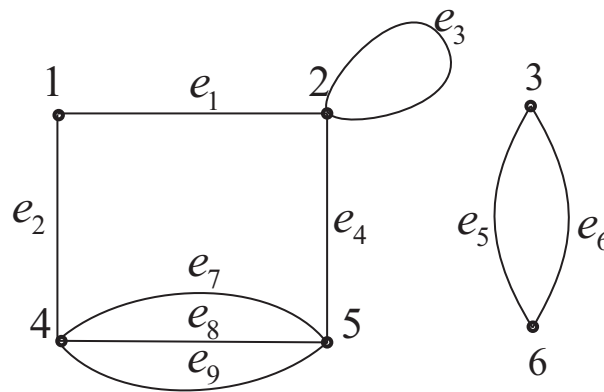


Figura 3.1.1:

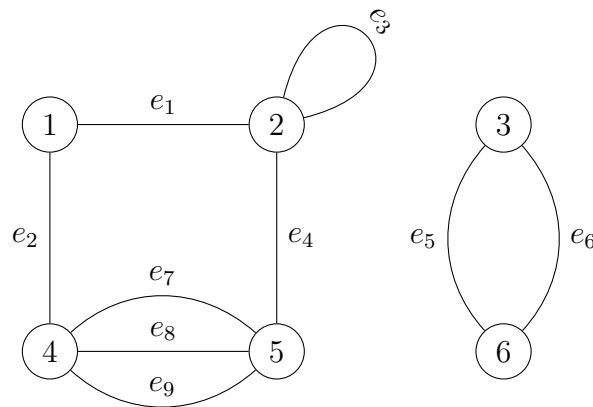


Figura 3.1.2:

Acest graf are ordinul 6 și dimensiunea 9. El conține bucla e_3 , muchiile multiple e_5 , e_6 și muchiile multiple e_7 , e_8 , e_9 , deci nu este nici multigraf, nici simplu. Nodurile 1 și 2 sunt adiacente, iar nodurile 1 și 5 nu sunt adiacente.

Exemplul 3.1.2. Graful orientat $G = (V, E)$, cu $V = \{1, 2, 3, 4, 5\}$ și

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

este un graf simplu având reprezentarea grafică din Figurile 3.1.3 cu nodurile reprezentate prin puncte) și 3.1.4 (cu nodurile reprezentate în cerculețe).

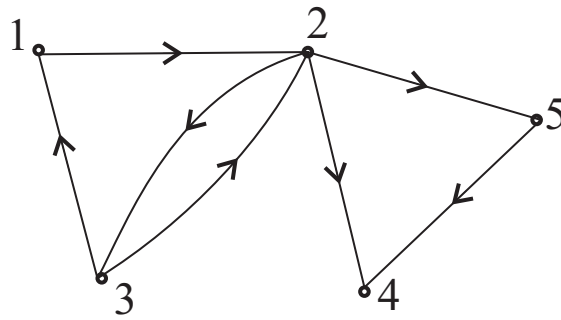


Figura 3.1.3:

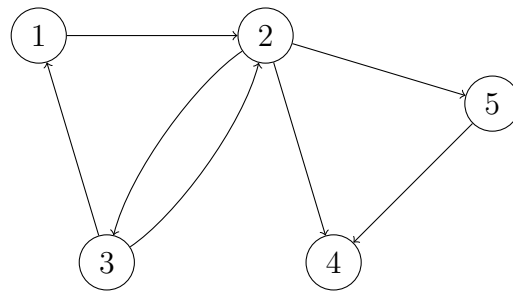


Figura 3.1.4:

Arcul $(3, 1)$ are extremitatea inițială 3 și extremitatea finală 1.

Definiția 3.1.9. Fie $G_1 = (V_1, E_1)$ și $G_2 = (V_2, E_2)$ două grafuri (ambele orientate sau ambele neorientate).

- a) Spunem că G_1 este un **subgraf** al lui G_2 și notăm $G_1 \subseteq G_2$ dacă $V_1 \subseteq V_2$ și $E_1 \subseteq E_2$.
- b) Spunem că G_1 este un **graf parțial** al lui G_2 dacă $V_1 = V_2$ și $E_1 \subseteq E_2$.

Definiția 3.1.10. Fie $G = (V, E)$ un graf (neorientat sau orientat) și $U \subseteq V$ o submulțime nevidă de noduri. **Subgraful indus (generat)** de U în G este subgraful $G[U] = (U, F)$, unde F este colecția tuturor muchiilor sau arcelor din E ce au ambele extremități în U .

Definiția 3.1.11. Fie $G = (V, E)$ un graf (orientat sau neorientat) și $F \subseteq E$ o colecție de muchii sau de arce.

- a) **Subgraful indus (generat)** de F în G este subgraful $G[F] = (U, F)$, unde U este mulțimea tuturor nodurilor din V ce sunt extremități pentru cel puțin o muchie sau un arc din F .
- b) **Graful parțial indus (generat)** de F în G este graful parțial (V, F) .

Exemplul 3.1.3. Pentru graful neorientat din Exemplul 3.1.1, subgraful generat de submulțimea de noduri $\{1, 3, 4, 5\}$ are reprezentarea din Figura 3.1.5.

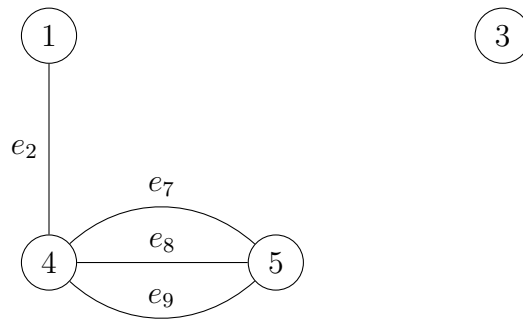


Figura 3.1.5:

Pentru graful orientat din Exemplul 3.1.2, subgraful generat de submulțimea de arce $\{(2, 3), (5, 4)\}$ are reprezentarea din Figura 3.1.6, iar graful parțial generat de aceeași submulțime de arce are reprezentarea din Figura 3.1.7.

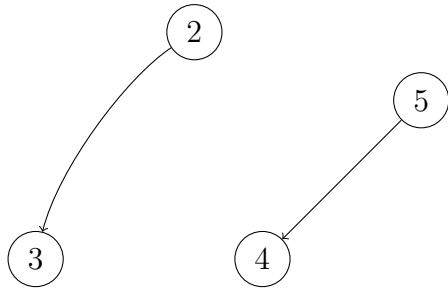


Figura 3.1.6:

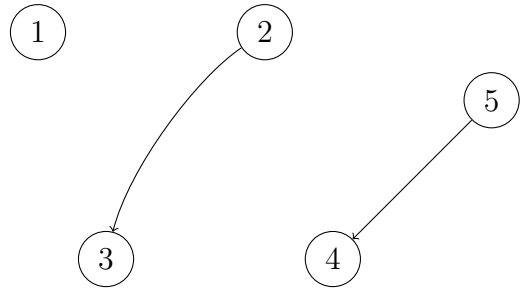


Figura 3.1.7:

Definiția 3.1.12. Fie $G = (V, E)$ un graf (neorientat sau orientat) și $U \subseteq V$ o submulțime de noduri a.î. $U \neq V$. **Subgraful obținut din G prin eliminarea nodurilor mulțimii U** este subgraful $G \setminus U = (V \setminus U, F)$, unde F este colecția tuturor muchiilor sau arcelor din E ce nu sunt incidente cu niciun nod din U .

Definiția 3.1.13. Fie $G = (V, E)$ un graf (neorientat sau orientat) și $F \subseteq E$ o colecție de muchii sau arce. **Graful parțial obținut din G prin eliminarea muchiilor sau arcelor din F** este $G \setminus F = (V, E \setminus F)$ (adică subgraful ce conține toate nodurile lui G și muchiile sau arcele lui G care nu aparțin lui F).

Exemplul 3.1.4. Pentru graful orientat din Exemplul 3.1.2, subgraful obținut prin eliminarea nodului 2 are reprezentarea din Figura 3.1.8.

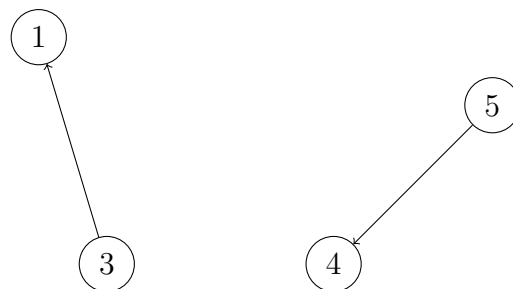


Figura 3.1.8:

Pentru graful neorientat din Exemplit 3.1.1, graful parțial obținut prin eliminarea muchiilor e_2, e_4, e_6, e_8 are reprezentarea din Figura 3.1.9.

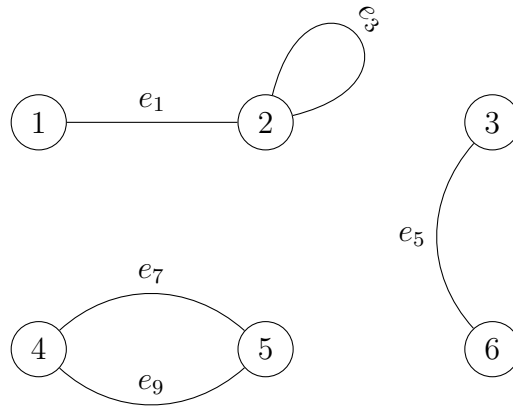


Figura 3.1.9:

Observația 3.1.5. Fie $G = (V, E)$ un graf (neorientat sau orientat) și $U \subseteq V$.

- a) Dacă $U \neq \emptyset$, atunci $G[U] = G \setminus (V \setminus U)$.
- b) Dacă $U \neq V$, atunci $G \setminus U = G[V \setminus U]$.

Observația 3.1.6. Fie $G = (V, E)$ un graf (neorientat sau orientat) și $F \subseteq E$.

- a) Graful parțial indus de F în G este $G \setminus (E \setminus F)$.
- b) Graful parțial $G \setminus F$ este chiar graful parțial indus de $E \setminus F$ în G .

3.2 Reprezentarea (memorarea) grafurilor

În continuare descriem câteva forme de reprezentare (memorare) a grafurilor în informatică. Dintre aceste forme, cea mai utilizată este matricea de adiacență.

Definiția 3.2.1. Fie $G = (V, E)$ un graf (neorientat sau orientat) unde $V = \{v_1, \dots, v_n\}$ și $E = \{e_1, \dots, e_m\}$. **Matricea de adiacență** asociată grafului G este matricea $A = (a_{ij})_{i,j=\overline{1,n}}$ definită prin

$a_{ij} =$ numărul de muchii sau de arce $e_k \in E$ de la nodul v_i la nodul v_j (adică muchii de forma $e_k = [v_i, v_j]$, respectiv arce de forma $e_k = (v_i, v_j)$), $\forall i, j \in \{1, \dots, n\}$.

Observația 3.2.1. a) Dacă graful neorientat $G = (V, E)$ este simplu, atunci

$$a_{ij} = \begin{cases} 1, & \text{dacă } v_i \text{ și } v_j \text{ sunt adiacente (adică } [v_i, v_j] \in E), \\ 0, & \text{în caz contrar.} \end{cases}$$

b) Dacă graful orientat $G = (V, E)$ este simplu, atunci

$$a_{ij} = \begin{cases} 1, & \text{dacă } (v_i, v_j) \in E, \\ 0, & \text{în caz contrar.} \end{cases}$$

Exemplul 3.2.1. Matricea de adiacență asociată grafului neorientat din Exemplul 3.1.1 este

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \\ 1 & 0 & 0 & 0 & 3 & 0 \\ 0 & 1 & 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{pmatrix},$$

iar matricea de adiacență asociată grafului orientat din Exemplul 3.1.2 este

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

Observația 3.2.2. Evident, orice graf neorientat are matricea de adiacență simetrică ($a_{ij} = a_{ji} \forall i, j$).

Propoziția 3.2.1. Fie $V = \{v_1, \dots, v_n\}$, $n \in \mathbb{N}^*$.

- a) Numărul de grafuri neorientate simple având mulțimea de noduri V este egal cu $2^{\frac{n(n-1)}{2}}$.
- b) Numărul de grafuri orientate simple având mulțimea de noduri V este egal cu 2^{n^2-n} .

Demonstrație. a) Orice graf neorientat simplu având mulțimea de noduri V este bine determinat de matricea sa de adiacență, care este o matrice binară (cu elemente 0 și 1), simetrică și cu elementele de pe diagonala principală egale cu zero. Cum o astfel de matrice este bine determinată de cele

$$n - 1 + \dots + 2 + 1 = \frac{n(n-1)}{2}$$

elemente 0 sau 1 de deasupra diagonalei principale, utilizând Propoziția 1.4.1 obținem că numărul de astfel de matrice este egal cu $2^{\frac{n(n-1)}{2}}$.

Punctul b) se demonstrează analog, folosind acum faptul că matricea de adiacență a unui graf orientat simplu este o matrice binară cu elementele de pe diagonala principală egale cu zero, deci este bine determinată de cele $n^2 - n$ elemente 0 sau 1 nesituate pe această diagonală. \square

Definiția 3.2.2. Fie $G = (V, E)$ un graf, unde $V = \{v_1, \dots, v_n\}$ și $E = \{e_1, \dots, e_m\}$, $E \neq \emptyset$.

- a) Dacă G este neorientat, atunci **matricea de incidență** asociată grafului G este matricea $B = (b_{ij})_{\substack{i = \overline{1, n} \\ j = \overline{1, m}}}$ definită prin

$$b_{ij} = \begin{cases} 0, & \text{dacă } e_j \text{ nu este incidentă cu } v_i, \\ 1, & \text{dacă } e_j \text{ nu este buclă și este incidentă cu } v_i, \\ 2, & \text{dacă } e_j \text{ este o buclă incidentă cu } v_i, \end{cases}$$

$$\forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, m\}.$$

b) Dacă G este orientat și fără bucle, atunci **matricea de incidență** asociată grafului G este matricea $B = (b_{ij})$ $i = \overline{1, n}$ $j = \overline{1, m}$ definită prin

$$b_{ij} = \begin{cases} 0, & \text{dacă } e_j \text{ nu este incidentă cu } v_i, \\ 1, & \text{dacă } e_j \text{ este incidentă cu } v_i \text{ spre exterior,} \\ -1, & \text{dacă } e_j \text{ este incidentă cu } v_i \text{ spre interior,} \end{cases}$$

$$\forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, m\}.$$

Exemplul 3.2.2. Matricea de incidență a grafului neorientat din Exemplul 3.1.1 este

$$B = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix},$$

iar matricea de incidență a grafului orientat din Exemplul 3.1.2 este

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

Observația 3.2.3. Fie $G = (V, E)$ un graf (neorientat sau orientat), unde $V = \{v_1, \dots, v_n\}$ și $E = \{e_1, \dots, e_m\}$, $E \neq \emptyset$. O altă reprezentare a grafului G este matricea T ce reține în fiecare coloană extremitățile unei muchii sau arc, adică $T = (t_{kj})$ $k = \overline{1, 2}$ $j = \overline{1, m}$ definită prin $t_{1j} = x$, $t_{2j} = y$, unde

$e_j = [x, y]$ (pentru muchii) sau $e_j = (x, y)$ (pentru arce), $\forall j \in \{1, \dots, m\}$. Pentru graful din Exemplul 3.1.1 matricea T este

$$T = \begin{pmatrix} 1 & 1 & 2 & 2 & 3 & 3 & 4 & 4 & 4 \\ 2 & 4 & 2 & 5 & 6 & 6 & 5 & 5 & 5 \end{pmatrix},$$

iar pentru graful din Exemplul 3.1.2 matricea T este

$$T = \begin{pmatrix} 1 & 2 & 2 & 2 & 3 & 3 & 5 \\ 2 & 3 & 4 & 5 & 1 & 2 & 4 \end{pmatrix}.$$

Observația 3.2.4. O altă formă frecvent utilizată în reprezentarea grafurilor simple este dată de **listele de adiacență** (memorate static sau dinamic). Lista de adiacență a unui nod x este formată din toate nodurile y pentru care există muchie sau arc de la x la y (y se numește **succesor direct** al lui x). Pentru graful orientat din Exemplul 3.1.2, listele de adiacență sunt

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

unde $L(i)$ reprezintă lista de adiacență a nodului i . Pentru grafurile orientate simple, deseori se memorează și listele de **predecesori direcți**. Lista de predecesori direcți a unui nod x este formată

din toate nodurile y pentru care există arc de la y la x (y se numește **predecesor direct** al lui x). Pentru graful din Exemplul 3.1.2, aceste liste sunt

$$\bar{L}(1) = \{3\}, \bar{L}(2) = \{1, 3\}, \bar{L}(3) = \{2\}, \bar{L}(4) = \{2, 5\}, \bar{L}(5) = \{2\}.$$

Evident, pentru grafurile neorientate noțiunile de *succesor direct* și *predecesor direct* coincid, fiind și sinonime cu noțiunile de *vecin* sau *adiacent*.

Observația 3.2.5. Alegerea uneia sau alteia dintre formele de reprezentare a grafurilor descrise mai sus depinde de eficiența și de ușurința implementării operațiilor necesare de prelucrare a acestor forme, deci de tipul problemei modelate prin grafuri și de algoritmul de rezolvare utilizat.

3.3 Grade, secvențe grafice

Definiția 3.3.1. Fie $G = (V, E)$ un graf și $x \in V$ un nod arbitrar fixat.

a) Dacă G este neorientat, atunci **gradul** lui x , notat $d_G(x) = d(x)$, este definit prin

$$d(x) = a(x) + 2 \cdot b(x),$$

unde $a(x)$ reprezintă numărul de muchii $e \in E$ ce nu sunt bucle și sunt incidente cu x , iar $b(x)$ este numărul de bucle $e \in E$ ce sunt incidente cu x .

b) Dacă G este orientat, atunci:

- **gradul de ieșire (semigradul exterior)** al lui x , notat $d_G^+(x) = d^+(x)$, reprezintă numărul de arce $e \in E$ incidente cu x spre exterior;
- **gradul de intrare (semigradul interior)** al lui x , notat $d_G^-(x) = d^-(x)$, reprezintă numărul de arce $e \in E$ incidente cu x spre interior;
- **gradul (total al)** lui x , notat $d_G(x) = d(x)$, este

$$d(x) = d^+(x) + d^-(x).$$

Exemplul 3.3.1. Pentru graful neorientat din Exemplul 3.1.1, gradele nodurilor sunt: $d(1) = 2$, $d(2) = 4$, $d(3) = 2$, $d(4) = 4$, $d(5) = 4$, $d(6) = 2$.

Pentru graful orientat din Exemplul 3.1.2, gradele nodurilor sunt:

$$\begin{aligned} d^+(1) &= 1, & d^-(1) &= 1, & d(1) &= 2, \\ d^+(2) &= 3, & d^-(2) &= 2, & d(2) &= 5, \\ d^+(3) &= 2, & d^-(3) &= 1, & d(3) &= 3, \\ d^+(4) &= 0, & d^-(4) &= 2, & d(4) &= 2, \\ d^+(5) &= 1, & d^-(5) &= 1, & d(5) &= 2. \end{aligned}$$

Definiția 3.3.2. Fie $G = (V, E)$ un graf. Un nod $x \in V$ cu $d_G(x) = 0$ se numește **nod izolat**, iar un nod $y \in V$ cu $d_G(y) = 1$ se numește **nod terminal**.

Propoziția 3.3.1. Fie $G = (V, E)$ un graf, $V = \{v_1, \dots, v_n\}$, și fie $A = (a_{ij})_{i,j=\overline{1,n}}$ matricea de adiacență a grafului G .

a) Dacă G este neorientat, atunci $d(v_i) = a_{ii} + \sum_{j=1}^n a_{ij}$, $\forall i \in \{1, \dots, n\}$.

b) Dacă G este orientat, atunci $d^+(v_i) = \sum_{j=1}^n a_{ij}$, $d^-(v_i) = \sum_{j=1}^n a_{ji}$, $\forall i \in \{1, \dots, n\}$.

Demonstrație. Egalitățile din enunț sunt evidente, conform definițiilor gradelor și matricei de adiacență. \square

Observația 3.3.1. Dacă graful G este simplu, atunci $a_{ii} = 0 \forall i$ și egalitatea de la punctul a) al propoziției anterioare devine $d(v_i) = \sum_{j=1}^n a_{ij}$.

Propoziția 3.3.2 (Euler). Fie $G = (V, E)$ un graf având dimensiunea m . Atunci $\sum_{x \in V} d(x) = 2m$.

În plus, dacă G este orientat atunci $\sum_{x \in V} d^+(x) = \sum_{x \in V} d^-(x) = m$.

Demonstrație. a) Considerăm că G este neorientat. Atunci fiecare muchie $e = [v_i, v_j] \in E$ (din cele m muchii) contribuie cu 1 la $d(v_i)$ și cu 1 la $d(v_j)$ (cu 2 la $d(v_i)$ dacă $v_i = v_j$, adică e = buclă), deci cu 2 la suma $\sum_{x \in V} d(x)$. Rezultă că această sumă este egală cu $2m$.

b) Considerăm că G este orientat. Deoarece fiecare arc $e = (v_i, v_j) \in E$ contribuie cu 1 la $d^+(v_i)$, deci la $\sum_{x \in V} d^+(x)$, și tot cu 1 la $d^-(v_j)$, deci la $\sum_{x \in V} d^-(x)$, rezultă că aceste sume sunt egale cu numărul total de arce, adică cu m . Cum $d(x) = d^+(x) + d^-(x) \forall x \in V$, obținem și $\sum_{x \in V} d(x) = \sum_{x \in V} d^+(x) + \sum_{x \in V} d^-(x) = 2m$. \square

Propoziția 3.3.3. Fie $d_1, d_2, \dots, d_n \in \mathbb{N}$, unde $n \in \mathbb{N}^*$. Atunci numerele d_1, d_2, \dots, d_n sunt gradele nodurilor unui graf neorientat de ordinul n dacă și numai dacă suma $\sum_{i=1}^n d_i$ este un număr par.

Demonstrație. " \Rightarrow " Conform Propoziției 3.3.2 avem $\sum_{i=1}^n d_i = 2m$, unde m este dimensiunea grafului având gradele nodurilor d_1, d_2, \dots, d_n , deci suma considerată este un număr par.

" \Leftarrow " Demonstrăm că dacă $\sum_{i=1}^n d_i = 2m$ cu $m \in \mathbb{N}$, atunci există un graf neorientat $G = (V, E)$ cu $V = \{v_1, \dots, v_n\}$ astfel încât $d_G(v_i) = d_i$, $\forall i \in \{1, \dots, n\}$, prin inducție după m .

Pentru $m = 0$ rezultă că $d_i = 0$, $\forall i \in \{1, \dots, n\}$, deci luând $E = \emptyset$ avem $d_G(v_i) = 0 = d_i$, $\forall i \in \{1, \dots, n\}$.

Presupunem afirmația adevărată pentru $m - 1$ și o demonstrăm pentru m , unde $m \in \mathbb{N}^*$. Fie $\sum_{i=1}^n d_i = 2m$. Avem două cazuri.

Cazul 1) Există $j, k \in \{1, \dots, n\}$, $j \neq k$, astfel încât $d_j \geq 1$ și $d_k \geq 1$. Fie

$$d'_i = \begin{cases} d_i - 1, & \text{dacă } i \in \{j, k\}, \\ d_i, & \text{dacă } i \in \{1, \dots, n\} \setminus \{j, k\}. \end{cases}$$

Atunci $\sum_{i=1}^n d'_i = \sum_{i=1}^n d_i - 2 = 2(m - 1)$, deci, conform ipotezei de inducție, există un graf neorientat $G' = (V, E')$ cu $V = \{v_1, \dots, v_n\}$ astfel încât $d_{G'}(v_i) = d'_i$, $\forall i \in \{1, \dots, n\}$. Fie graful $G = (V, E' \cup \{e\})$, unde $e = [v_j, v_k]$, $e \notin E'$. Evident, avem $d_G(v_j) = d_{G'}(v_j) + 1 = d'_j + 1 = d_j$, $d_G(v_k) = d_{G'}(v_k) + 1 = d'_k + 1 = d_k$ și $d_G(v_i) = d_{G'}(v_i) = d'_i = d_i$ pentru orice $i \in \{1, \dots, n\} \setminus \{j, k\}$. Astfel $d_G(v_i) = d_i$, $\forall i \in \{1, \dots, n\}$.

Cazul 2) Există $j \in \{1, \dots, n\}$ astfel încât $d_i = 0$ pentru orice $i \in \{1, \dots, n\} \setminus \{j\}$. Atunci $d_j = \sum_{i=1}^n d_i = 2m$. Fie graful $G = (V, E)$, cu $V = \{v_1, \dots, v_n\}$ și $E = \{e_1, \dots, e_m\}$, unde $e_k = [v_j, v_j]$, $\forall k \in \{1, \dots, m\}$, iar buclele e_1, \dots, e_m sunt distincte două câte două. Evident, avem $d_G(v_j) = 2m = d_j$ și $d_G(v_i) = 0 = d_i$ pentru orice $i \in \{1, \dots, n\} \setminus \{j\}$. Astfel, din nou, $d_G(v_i) = d_i$, $\forall i \in \{1, \dots, n\}$.

Demonstrația prin inducție este încheiată. \square

Observația 3.3.2. Demonstrația propoziției anterioare este constructivă, ea indicând un algoritm de generare a unui graf neorientat cu gradele nodurilor date.

Exemplul 3.3.2. Numerele 1, 1, 1, 2, 4, 4, 5, 5 nu pot fi gradele nodurilor unui graf neorientat, deoarece suma $\sum_{i=1}^8 d_i = 23$ este un număr impar.

Exemplul 3.3.3. Numerele 1, 1, 1, 1, 2, 2, 3, 13 pot fi gradele nodurilor unui graf neorientat, deoarece suma $\sum_{i=1}^8 d_i = 24$ este un număr par. Mai mult, conform demonstrației propoziției anterioare (partea " \Leftarrow "), un graf având gradele nodurilor numerele date se construiește adăugând succesiv câte o muchie între două noduri de grade nenule, grade care se micșorează cu 1, până când fie toate gradele devin egale cu zero, fie rămâne un grad nenul d'_i , par, care este anulat și el prin $d'_i/2$ bucle. Acest procedeu este evidențiat în tabelul următor:

Pas	Gradele rămase	Muchia adăugată
1	1, 1, 1, 1, 2, 2, 3, 13	$[v_1, v_2]$
2	0, 0, 1, 1, 2, 2, 3, 13	$[v_3, v_4]$
3	0, 0, 0, 0, 2, 2, 3, 13	$[v_5, v_6]$
4	0, 0, 0, 0, 1, 1, 3, 13	$[v_5, v_6]$
5	0, 0, 0, 0, 0, 0, 3, 13	$[v_7, v_8]$
6	0, 0, 0, 0, 0, 0, 2, 12	$[v_7, v_8]$
7	0, 0, 0, 0, 0, 0, 1, 11	$[v_7, v_8]$
8	0, 0, 0, 0, 0, 0, 0, 10	$[v_8, v_8]$ de 5 ori
—	0, 0, 0, 0, 0, 0, 0, 0	—

Graful obținut are matricea de adiacență

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 5 \end{pmatrix}.$$

Propoziția 3.3.4. Fie $d_1, d_2, \dots, d_n \in \mathbb{N}$ astfel încât $d_1 \leq d_2 \leq \dots \leq d_n$, unde $n \in \mathbb{N}$, $n \geq 2$. Atunci numerele d_1, d_2, \dots, d_n sunt gradele nodurilor unui multigraf neorientat de ordinul n dacă și numai dacă suma $\sum_{i=1}^n d_i$ este un număr par și $d_n \leq \sum_{i=1}^{n-1} d_i$.

Demonstrație. " \Rightarrow " Conform Propoziției 3.3.2 avem $\sum_{i=1}^n d_i = 2m$, unde m este dimensiunea multigrafului având gradele nodurilor d_1, d_2, \dots, d_n , deci suma considerată este un număr par. De asemenea,

avem $d_n = \sum_{i=1}^{n-1} a_i$, unde, pentru orice $i \in \{1, \dots, n-1\}$, a_i reprezintă numărul de muchii de forma $[v_n, v_i]$. Cum $a_i \leq d_i$ pentru orice $i \in \{1, \dots, n-1\}$, rezultă că $d_n \leq \sum_{i=1}^{n-1} d_i$.

” \Leftarrow ” Arătăm că dacă $\sum_{i=1}^n d_i = 2m$ cu $m \in \mathbb{N}$, $d_1 \leq d_2 \leq \dots \leq d_n$ și $d_n \leq \sum_{i=1}^{n-1} d_i$, atunci există un multigraf neorientat $G = (V, E)$ cu $V = \{v_1, \dots, v_n\}$ astfel încât $d_G(v_i) = d_i$, $\forall i \in \{1, \dots, n\}$.

Pentru $n = 2$ rezultă că $d_1 = d_2$, deci considerând multigraful $G = (V, E)$, cu $V = \{v_1, v_2\}$ și $E = \{e_1, \dots, e_m\}$, unde $m = d_1 = d_2$ și $e_k = [v_1, v_2]$, $\forall k \in \{1, \dots, m\}$, iar muchiile multiple e_1, \dots, e_m sunt distincte două câte două, obținem că $d_G(v_1) = m = d_1$ și $d_G(v_2) = m = d_2$.

Pentru $n \geq 3$ demonstrăm afirmația prin inducție după m .

Pentru $m = 0$ rezultă că $d_i = 0$, $\forall i \in \{1, \dots, n\}$, deci luând $E = \emptyset$ avem $d_G(v_i) = 0 = d_i$, $\forall i \in \{1, \dots, n\}$.

Presupunem afirmația adevărată pentru $m-1$ și o demonstrăm pentru m , unde $m \in \mathbb{N}^*$. Fie $\sum_{i=1}^n d_i = 2m$, $d_1 \leq d_2 \leq \dots \leq d_n$ și $d_n \leq \sum_{i=1}^{n-1} d_i$. Deoarece $\sum_{i=1}^n d_i = 2m \geq 2$, rezultă că $d_n \geq 1$. Cum $d_n \leq \sum_{i=1}^{n-1} d_i$, rezultă că și $d_{n-1} \geq 1$. Fie

$$d'_i = \begin{cases} d_i, & \text{dacă } 1 \leq i \leq n-2, \\ d_i - 1, & \text{dacă } i \in \{n-1, n\}. \end{cases}$$

Atunci $\sum_{i=1}^n d'_i = \sum_{i=1}^n d_i - 2 = 2(m-1)$. Avem două cazuri.

Cazul 1) $d_{n-2} < d_n$. Atunci $d'_1 \leq d'_2 \leq \dots \leq d'_n$ și $d'_n = d_n - 1 \leq \sum_{i=1}^{n-1} d_i - 1 = \sum_{i=1}^{n-1} d'_i$, deci, conform ipotezei de inducție, există un multigraf neorientat $G' = (V, E')$ cu $V = \{v_1, \dots, v_n\}$ astfel încât $d_{G'}(v_i) = d'_i$, $\forall i \in \{1, \dots, n\}$.

Cazul 2) $d_{n-2} = d_n$, deci și $d_{n-1} = d_n$. Atunci $d'_1 \leq d'_2 \leq \dots \leq d'_{n-2} = d_n$ și $d'_{n-1} = d'_n = d_n - 1$, deci

$$\max_{i=1, n} d'_i = d'_{n-2}.$$

Avem $\sum_{i=1}^n d'_i - 2d'_{n-2} = \sum_{i=1}^{n-3} d_i + d_n + (d_n - 1) + (d_n - 1) - 2d_n = \sum_{i=1}^{n-3} d_i + d_n - 2 \geq -1$, deoarece $d_n \geq 1$.

Pe de altă parte, $\sum_{i=1}^n d'_i - 2d'_{n-2} = 2(m-1) - 2d'_{n-2}$, deci $\sum_{i=1}^n d'_i - 2d'_{n-2}$ este un număr par. Astfel rezultă că $\sum_{i=1}^n d'_i - 2d'_{n-2} \geq 0$. Obținem că

$$\max_{i=1, n} d'_i = d'_{n-2} \leq \sum_{i=1}^{n-3} d'_i + d'_{n-1} + d'_n,$$

deci, conform ipotezei de inducție (aplicată pentru numerele d'_1, d'_2, \dots, d'_n rearanjate în ordine crescătoare), rezultă din nou că există un multigraf neorientat $G' = (V, E')$ cu $V = \{v_1, \dots, v_n\}$ astfel încât $d_{G'}(v_i) = d'_i$, $\forall i \in \{1, \dots, n\}$.

În ambele cazuri, fie multigraful $G = (V, E' \cup \{e\})$, unde $e = [v_{n-1}, v_n]$, $e \notin E'$. Evident, avem $d_G(v_{n-1}) = d_{G'}(v_{n-1}) + 1 = d'_{n-1} + 1 = d_{n-1}$, $d_G(v_n) = d_{G'}(v_n) + 1 = d'_n + 1 = d_n$ și $d_G(v_i) = d_{G'}(v_i) = d'_i = d_i$ pentru orice $i \in \{1, \dots, n-2\}$. Astfel $d_G(v_i) = d_i$, $\forall i \in \{1, \dots, n\}$.

Demonstrația prin inducție este încheiată. \square

Observația 3.3.3. Demonstrația propoziției anterioare este constructivă, ea indicând un algoritm de generare a unui multigraf neorientat cu gradele nodurilor date.

Exemplul 3.3.4. Numerele 1, 1, 1, 1, 2, 2, 3, 13 din exemplul anterior nu pot fi gradele nodurilor unui multigraf neorientat, deoarece suma $\sum_{i=1}^8 d_i = 24$ este un număr par, dar $d_8 = 13 > 11 = \sum_{i=1}^7 d_i$.

Exemplul 3.3.5. Numerele 1, 1, 1, 2, 4, 5, 5, 5 pot fi gradele nodurilor unui multigraf neorientat, deoarece suma $\sum_{i=1}^8 d_i = 24$ este un număr par și $d_8 = 5 \leq 19 = \sum_{i=1}^7 d_i$. Mai mult, conform demonstrației propoziției anterioare (partea " \Leftarrow "), un multigraf având gradele nodurilor numerele date se construiește adăugând succesiv câte o muchie între două noduri de grade nenule maxime, grade care se micșorează cu 1, până când gradele devin toate egale cu zero. Acest procedeu este evidențiat în tabelul următor:

Pas	Gradele rămase	Muchia adăugată
1	1, 1, 1, 2, 4, 5, 5, 5	$[v_7, v_8]$
2	1, 1, 1, 2, 4, 5, 4, 4	$[v_6, v_8]$
3	1, 1, 1, 2, 4, 4, 4, 3	$[v_6, v_7]$
4	1, 1, 1, 2, 4, 3, 3, 3	$[v_5, v_8]$
5	1, 1, 1, 2, 3, 3, 3, 2	$[v_6, v_7]$
6	1, 1, 1, 2, 3, 2, 2, 2	$[v_5, v_8]$
7	1, 1, 1, 2, 2, 2, 2, 1	$[v_6, v_7]$
8	1, 1, 1, 2, 2, 1, 1, 1	$[v_4, v_5]$
9	1, 1, 1, 1, 1, 1, 1, 1	$[v_7, v_8]$
10	1, 1, 1, 1, 1, 1, 0, 0	$[v_5, v_6]$
11	1, 1, 1, 1, 0, 0, 0, 0	$[v_3, v_4]$
12	1, 1, 0, 0, 0, 0, 0, 0	$[v_1, v_2]$
–	0, 0, 0, 0, 0, 0, 0, 0	–

Multigraful obținut are matricea de adiacență

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 1 & 2 & 0 \end{pmatrix}.$$

Definiția 3.3.3. O secvență (d_1, d_2, \dots, d_n) de numere naturale ($n \in \mathbb{N}^*$) se numește **secvență grafică** (**secvență de grade**, **șir grafic**, **șir de grade**) dacă există un graf neorientat simplu $G = (V, E)$ cu $V = \{v_1, v_2, \dots, v_n\}$ astfel încât $d_G(v_i) = d_i$, $\forall i \in \{1, 2, \dots, n\}$.

Teorema 3.3.1 (Havel-Hakimi). Fie $d_1, d_2, \dots, d_n \in \mathbb{N}$ astfel încât $d_1 \leq d_2 \leq \dots \leq d_n$ și $d_n \geq 1$, unde $n \in \mathbb{N}$, $n \geq 2$. Atunci (d_1, d_2, \dots, d_n) este o secvență grafică dacă și numai dacă $d_n \leq n - 1$ și $(d_1, d_2, \dots, d_{n-d_n-1}, d_{n-d_n} - 1, d_{n-d_n+1} - 1, \dots, d_{n-1} - 1)$ este secvență grafică.

Demonstrație. " \Rightarrow " Fie $G = (V, E)$ un graf neorientat simplu cu $V = \{v_1, \dots, v_n\}$ astfel încât $d_G(v_i) = d_i$, $\forall i \in \{1, \dots, n\}$. Deoarece graful G este simplu, adică nu conține nici bucle, nici muchii multiple, rezultă că gradul $d_n = d_G(v_n)$ este egal cu numărul de noduri adiacente cu v_n , deci $d_n \leq n - 1$. Fie M mulțimea nodurilor adiacente cu v_n , deci $\text{card}(M) = d_n$. Avem două cazuri.

Cazul 1) $M = \{v_{n-d_n}, v_{n-d_n+1}, \dots, v_{n-1}\}$. Atunci graful $G' = (V', E')$ definit prin $V' = V \setminus \{v_n\} = \{v_1, \dots, v_{n-1}\}$, $E' = E \setminus \{[v_n, x] \mid x \in M\}$ este un graf neorientat simplu având gradele nodurilor

$$\begin{aligned} d_{G'}(v_i) &= d_G(v_i) = d_i, \text{ pentru } 1 \leq i \leq n - d_n - 1, \\ d_{G'}(v_i) &= d_G(v_i) - 1 = d_i - 1, \text{ pentru } n - d_n \leq i \leq n - 1, \end{aligned}$$

deci $(d_1, d_2, \dots, d_{n-d_n-1}, d_{n-d_n} - 1, d_{n-d_n+1} - 1, \dots, d_{n-1} - 1)$ este secvență grafică.

Cazul 2) $M \neq \{v_{n-d_n}, v_{n-d_n+1}, \dots, v_{n-1}\}$. Fie

$$A = \{v_{n-d_n}, v_{n-d_n+1}, \dots, v_{n-1}\} \setminus M, \quad B = M \setminus \{v_{n-d_n}, v_{n-d_n+1}, \dots, v_{n-1}\}.$$

Cum $\text{card}(\{v_{n-d_n}, v_{n-d_n+1}, \dots, v_{n-1}\}) = \text{card}(M) = d_n$, rezultă că avem $\text{card}(A) = \text{card}(B) \geq 1$. Fie $r = \text{card}(A) = \text{card}(B)$, $A = \{v_{j_1}, v_{j_2}, \dots, v_{j_r}\}$ și $B = \{v_{k_1}, v_{k_2}, \dots, v_{k_r}\}$, unde $\{j_1, j_2, \dots, j_r\} \subseteq \{n-d_n, n-d_n+1, \dots, n-1\}$ și $\{k_1, k_2, \dots, k_r\} \subseteq \{1, 2, \dots, n-d_n-1\}$. Pentru orice $i \in \{1, 2, \dots, r\}$, cum $d_G(v_{j_i}) = d_{j_i} \geq d_{k_i} = d_G(v_{k_i})$ și v_n nu este adiacent cu v_{j_i} , dar este adiacent cu v_{k_i} , rezultă că există un nod $v_{t_i} \in V$ care este adiacent cu v_{j_i} , dar nu este adiacent cu v_{k_i} . Fie graful $G_1 = (V, E_1)$ definit prin

$$E_1 = \left(E \setminus \bigcup_{i=1}^r \{[v_n, v_{k_i}], [v_{t_i}, v_{j_i}]\} \right) \cup \bigcup_{i=1}^r \{[v_n, v_{j_i}], [v_{t_i}, v_{k_i}]\}.$$

Atunci G_1 este un graf neorientat simplu având aceleași grade ale nodurilor ca și G , adică $d_{G_1}(v_i) = d_G(v_i) = d_i$, pentru orice $i \in \{1, 2, \dots, n\}$. În plus, în graful G_1 mulțimea nodurilor adiacente cu v_n este chiar mulțimea $\{v_{n-d_n}, v_{n-d_n+1}, \dots, v_{n-1}\}$. Conform demonstrației de la Cazul 1 (înlocuind G cu G_1) rezultă din nou că $(d_1, d_2, \dots, d_{n-d_n-1}, d_{n-d_n} - 1, d_{n-d_n+1} - 1, \dots, d_{n-1} - 1)$ este secvență grafică.

" \Leftarrow " Fie $d_n \leq n - 1$ și $G' = (V', E')$ un graf neorientat simplu cu $V' = \{v_1, \dots, v_{n-1}\}$ astfel încât

$$\begin{aligned} d_{G'}(v_i) &= d_i, \text{ pentru } 1 \leq i \leq n - d_n - 1, \\ d_{G'}(v_i) &= d_i - 1, \text{ pentru } n - d_n \leq i \leq n - 1. \end{aligned}$$

Atunci graful $G = (V, E)$ definit prin $V = V' \cup \{v_n\} = \{v_1, \dots, v_n\}$ (cu $v_n \notin V'$) și

$$E = E' \cup \{[v_n, v_i] \mid n - d_n \leq i \leq n - 1\}$$

este un graf neorientat simplu având gradele nodurilor

$$\begin{aligned} d_G(v_i) &= d_{G'}(v_i) = d_i, \text{ pentru } 1 \leq i \leq n - d_n - 1, \\ d_G(v_i) &= d_{G'}(v_i) + 1 = d_i - 1 + 1 = d_i, \text{ pentru } n - d_n \leq i \leq n - 1, \\ d_G(v_n) &= d_n, \end{aligned}$$

deci (d_1, d_2, \dots, d_n) este secvență grafică. □

Observația 3.3.4. Demonstrația teoremei anterioare este constructivă, ea indicând un algoritm recursiv de generare a unui graf neorientat simplu cu gradele nodurilor date.

Exemplul 3.3.6. Verificăm dacă $(1, 1, 1, 2, 4, 5, 5, 5)$ este secvență grafică. Conform teoremei anterioare avem echivalentele:

$$\begin{aligned} (1, 1, 1, 2, 4, 5, 5, 5) &\text{ este secvență grafică} \\ \Leftrightarrow 5 \leq 7 \text{ și } (1, 1, 0, 1, 3, 4, 4) &\text{ este secvență grafică} \\ \Leftrightarrow 4 \leq 6 \text{ și } (1, 0, 0, 0, 2, 3) &\text{ este secvență grafică} \\ \Leftrightarrow 3 \leq 5 \text{ și } (0, 0, 0, -1, 1) &\text{ este secvență grafică.} \end{aligned}$$

Cum ultima secvență nu este o secvență grafică (deoarece gradele nu pot fi negative), rezultă că nici secvența inițială nu este secvență grafică.

Exemplul 3.3.7. Verificăm dacă secvența $(1, 1, 2, 3, 4, 5, 5, 5)$ din exemplul anterior este secvență grafică. Conform teoremei anterioare avem echivalentele:

$$\begin{aligned}
 S_1 &= (1, 1, 2, 3, 4, 5, 5, 5) \text{ este secvență grafică} \\
 &\Leftrightarrow 5 \leq 7 \text{ și } S_2 = (1, 1, 1, 2, 3, 4, 4) \text{ este secvență grafică} \\
 &\Leftrightarrow 4 \leq 6 \text{ și } S_3 = (1, 1, 0, 1, 2, 3) \text{ este secvență grafică} \\
 &\Leftrightarrow 3 \leq 5 \text{ și } S_4 = (1, 0, 0, 0, 1) \text{ este secvență grafică} \\
 &\Leftrightarrow 1 \leq 4 \text{ și } S_5 = (0, 0, 0, 0) \text{ este secvență grafică.}
 \end{aligned}$$

Cum ultima secvență, S_5 , este o secvență grafică, fiind secvența gradelor grafului simplu de ordin 4 și dimensiune 0, rezultă că și secvența inițială, S_1 , este secvență grafică. Mai mult, conform demonstrației teoremei amintite (partea " \Leftarrow "), avem succesiv:

$$\begin{aligned}
 S_5 &= (0, 0, 0, 0) \text{ este secvența gradelor grafului } G_5 = (\{v_1, \dots, v_4\}, E_5) \text{ cu } E_5 = \emptyset; \\
 S_4 &= (1, 0, 0, 0, 1) \text{ este secvența gradelor grafului } G_4 = (\{v_1, \dots, v_5\}, E_4) \text{ cu} \\
 &\quad E_4 = E_5 \cup \{[v_5, v_1]\}; \\
 S_3 &= (1, 1, 0, 1, 2, 3) \text{ este secvența gradelor grafului } G_3 = (\{v_1, \dots, v_6\}, E_3) \text{ cu} \\
 &\quad E_3 = E_4 \cup \{[v_6, v_2], [v_6, v_4], [v_6, v_5]\}; \\
 S_2 &= (1, 1, 1, 2, 3, 4, 4) \text{ este secvența gradelor grafului } G_2 = (\{v_1, \dots, v_7\}, E_2) \text{ cu} \\
 &\quad E_2 = E_3 \cup \{[v_7, v_3], [v_7, v_4], [v_7, v_5], [v_7, v_6]\}; \\
 S_1 &= (1, 1, 2, 3, 4, 5, 5, 5) \text{ este secvența gradelor grafului } G_1 = (\{v_1, \dots, v_8\}, E_1) \text{ cu} \\
 &\quad E_1 = E_2 \cup \{[v_8, v_3], [v_8, v_4], [v_8, v_5], [v_8, v_6], [v_8, v_7]\}.
 \end{aligned}$$

Graful G_1 este reprezentat în Figura 3.3.1.

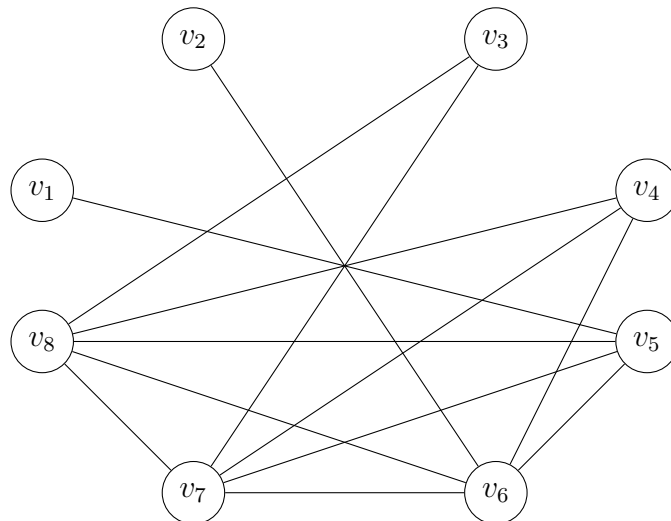


Figura 3.3.1:

3.4 Conexitate

Definiția 3.4.1. Fie $G = (V, E)$ un graf.

a) Un **lanț** în graful G este o succesiune alternantă

$$[v_1, e_1, v_2, e_2, \dots, v_k, e_k, v_{k+1}],$$

unde $v_1, v_2, \dots, v_{k+1} \in V$ (noduri), $e_1, e_2, \dots, e_k \in E$ (muchii sau arce), $k \in \mathbb{N}$, cu proprietatea că pentru orice $i \in \{1, \dots, k\}$ e_i este o muchie sau un arc având extremitățile v_i și v_{i+1} (adică $e_i = [v_i, v_{i+1}]$ pentru un graf neorientat, respectiv $e_i = (v_i, v_{i+1})$ sau $e_i = (v_{i+1}, v_i)$ pentru un graf orientat). Nodurile v_1 și v_{k+1} se numesc **extremitățile** lanțului, iar nodurile v_2, \dots, v_k se numesc **noduri intermediare**. Numărul k de muchii sau arce (nu neapărat distincte) ale lanțului se numește **lungimea** lanțului.

b) Un lanț se numește **închis** dacă are extremitățile egale, respectiv **deschis** în caz contrar.

c) Un lanț se numește **simplic** dacă muchiile sau arcele sale sunt diferite două câte două (considerând diferite și muchiile sau arcele multiple aflate pe poziții diferite în E).

d) Un lanț **deschis** se numește **elementar** dacă nodurile sale sunt diferite două câte două. Un lanț **închis** se numește **elementar** dacă este simplic și, cu excepția extremităților, nodurile sale sunt diferite două câte două.

e) Un **ciclu** este un lanț închis și simplic de lungime nenulă (adică cu cel puțin o muchie sau un arc).

f) Un **drum** este un lanț $[v_1, e_1, v_2, e_2, \dots, v_k, e_k, v_{k+1}]$ cu proprietatea că pentru orice $i \in \{1, \dots, k\}$ muchia sau arcul e_i are extremitatea inițială v_i și extremitatea finală v_{i+1} (adică $e_i = (v_i, v_{i+1})$ pentru graf orientat).

Un astfel de drum se notează și cu

$$(v_1, e_1, v_2, e_2, \dots, v_k, e_k, v_{k+1}),$$

nodurile v_1 și v_{k+1} numindu-se **extremitatea inițială**, respectiv **extremitatea finală** a drumului.

g) Un **circuit** este un drum închis și simplic de lungime nenulă (adică un drum ce este și ciclu).

Observația 3.4.1. Pentru grafurile neorientate noțiunile de lanț și de drum coincid; deci și noțiunile de ciclu și de circuit coincid.

Observația 3.4.2. Orice lanț elementar este și simplic.

Observația 3.4.3. Pentru grafurile neorientate sau orientate simple orice lanț $[v_1, e_1, v_2, e_2, \dots, v_k, e_k, v_{k+1}]$ sau drum $(v_1, e_1, v_2, e_2, \dots, v_k, e_k, v_{k+1})$ este bine determinat de nodurile sale (deoarece muchia sau arcul e_i nu poate fi decât singura muchie sau singurul arc de la v_i la v_{i+1} , $\forall i$), deci poate fi notat, pe scurt, doar prin nodurile succesive

$$[v_1, v_2, \dots, v_k, v_{k+1}], \text{ respectiv } (v_1, v_2, \dots, v_k, v_{k+1}).$$

Exemplul 3.4.1. Pentru graful neorientat din Exemplul 3.1.1,

$$[1, e_2, 4, e_9, 5, e_4, 2, e_1, 1, e_2, 4]$$

este un lanț (drum) deschis, nesimplu (conține de două ori e_2), neelementar (conține de două ori 1), de lungime 5.

În același graf, $[1, e_1, 2, e_3, 2, e_4, 5, e_7, 4, e_9, 5, e_8, 4, e_2, 1]$ este un ciclu (circuit) simplu și neelementar (conține de două ori 2), de lungime 7.

Pentru graful orientat simplu din Exemplul 3.1.2, $[3, 1, 2, 4, 5]$ este un lanț ce nu este drum (deoarece $(4, 5)$ nu este arc, ci $(5, 4)$), iar $(3, 1, 2, 5, 4)$ este un drum (și lanț) deschis elementar (și simplu) de lungime 4. În același graf, $(1, 2, 3, 1)$ este un circuit (și ciclu) elementar, iar $[2, 5, 4, 2]$ este un ciclu elementar ce nu este circuit (deoarece $(4, 2)$ nu este arc, ci $(2, 4)$).

Definiția 3.4.2. a) Un graf (neorientat sau orientat) se numește **conex** dacă pentru orice două noduri distincte x, y există cel puțin un lanț de la x la y .

b) Un graf orientat se numește **tare-conex** dacă pentru orice două noduri distincte x, y există cel puțin un drum de la x la y (deci, schimbând ordinea lui x și y , există cel puțin un drum și de la y la x).

Observația 3.4.4. Orice graf tare-conex este și conex (deoarece orice drum este și lanț). Orice graf cu un singur nod verifică definiția anterioară, deci este și conex și tare-conex.

Observația 3.4.5. Deoarece pentru orice nod x există lanțul $[x]$ și drumul (x) de lungimi zero, rezultă că în definiția anterioară putem renunța la condițiile ca nodurile x și y să fie distincte.

De asemenea, deoarece prin eliminarea tuturor porțiunilor dintre noduri egale dintr-un lanț sau drum se obține un lanț sau un drum elementar având aceleași extremități, rezultă că în definiția anterioară putem înlocui termenii de "lanț" și "drum" cu "lanț elementar", respectiv "drum elementar".

Exemplul 3.4.2. Graful neorientat din Exemplul 3.1.1 nu este conex, deoarece nu există lanțuri între nodurile 1 și 3. Graful orientat din Exemplul 3.1.2 este conex, dar nu este tare-conex, deoarece nu există drum de la nodul 4 la nodul 1 (deși există drum de la nodul 1 la nodul 4!).

Definiția 3.4.3. a) O **componentă conexă** a unui graf (orientat sau neorientat) $G = (V, E)$ este un subgraf $G[U]$ generat de o submulțime $U \subseteq V$ de noduri cu proprietatea că $G[U]$ este conex și maximal cu această proprietate (în raport cu incluziunea), adică pentru orice nod $x \in V \setminus U$ subgraful $G[U \cup \{x\}]$ nu mai este conex.

b) O **componentă tare-conexă** a unui graf orientat $G = (V, E)$ este un subgraf $G[U]$ generat de o mulțime $U \subseteq V$ de noduri cu proprietatea că $G[U]$ este tare-conex și maximal cu această proprietate (în raport cu incluziunea), adică pentru orice noduri $x_1, x_2, \dots, x_p \in V \setminus U$ subgraful $G[U \cup \{x_1, x_2, \dots, x_p\}]$ nu mai este tare-conex.

Observația 3.4.6. Un graf este conex dacă și numai dacă are o singură componentă conexă. Un graf orientat este tare-conex dacă și numai dacă are o singură componentă tare-conexă. Numărul de componente tare-conexe ale unui graf orientat este mai mare sau egal decât numărul de componente conexes ale acelui graf, deoarece orice componentă tare-conexă este inclusă într-o componentă conexă (conform definiției anterioare și Observației 3.4.4).

Observația 3.4.7. Orice nod izolat x generează o componentă conexă și o componentă tare-conexă, ambele având forma $G[\{x\}] = (\{x\}, \emptyset)$.

- Propoziția 3.4.1.** a) Fie $G[V_1], \dots, G[V_k]$ componentele conexe (diferite) ale unui graf $G = (V, E)$. Atunci $\{V_1, \dots, V_k\}$ este o partiție a mulțimii V (adică $V_i \neq \emptyset \forall i$, $V_i \cap V_j = \emptyset \forall i \neq j$, $V_1 \cup \dots \cup V_k = V$).
- b) Fie $G[V'_1], \dots, G[V'_r]$ componentele tare-conexe (diferite) ale unui graf orientat $G = (V, E)$. Atunci $\{V'_1, \dots, V'_r\}$ este de asemenea o partiție a mulțimii V .

Exemplul 3.4.3. Componentele conexe ale grafului neorientat din Exemplul 3.1.1 sunt generate de submulțimile de noduri $V_1 = \{1, 2, 4, 5\}$ și $V_2 = \{3, 6\}$, deci acel graf are 2 componente conexe. Componentele tare-conexe ale grafului orientat din Exemplul 3.1.2 sunt generate de submulțimile de noduri $V_1 = \{1, 2, 3\}$, $V_2 = \{4\}$ și $V_3 = \{5\}$, deci acel graf are 3 componente tare-conexe.

Observația 3.4.8. Submulțimile de muchii sau arce ale componentelor conexe ale unui graf formează de asemenea o partiție a mulțimii de muchii sau arce a grafului (deoarece pentru orice muchie $e = [x, y]$ sau arc $e = (x, y)$ nodurile x și y se află într-o aceeași componentă conexă și această componentă va conține și pe e). Afirmatia nu mai este valabilă pentru componentele tare-conexe. De exemplu, pentru graful din Exemplul 3.1.2 arcul $(5, 4)$ nu aparține niciunei componente tare-conexe.

Algoritmi pentru testarea conexității și tare-conexității unui graf, precum și pentru determinarea componentelor conexe și tare-conexe ale unui graf, vor fi prezentați în secțiunile următoare.

Definiția 3.4.4. a) Un **arbore** este un graf conex și fără cicluri.

b) O **pădure** este un graf fără cicluri.

c) Un **arbore parțial** al unui graf $G = (V, E)$ este un graf parțial al lui G ce este arbore (adică un arbore $T = (V, F)$ cu $F \subseteq E$).

Observația 3.4.9. Arborii și pădurile sunt grafuri simple (deoarece orice buclă este un ciclu și orice două muchii sau arce multiple formează un ciclu).

Observația 3.4.10. Componentele conexe ale unei păduri sunt arbori.

Exemplul 3.4.4. Graful neorientat din Exemplul 3.1.1 nu este pădure (deoarece are cicluri), deci nici arbore. Graful său parțial reprezentat în Figura 3.4.1 este o pădure (având două componente conexe arbori).

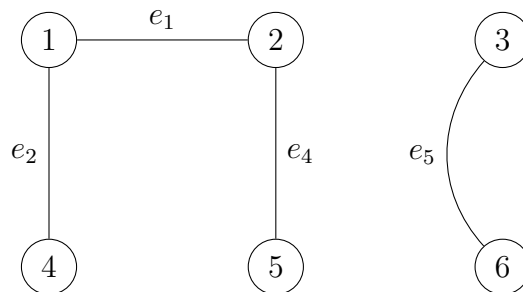


Figura 3.4.1:

Graful orientat din Exemplul 3.1.2 nu este arbore (deoarece are cicluri). Doi arbori parțiali ai săi sunt reprezentați în Figurile 3.4.2 și 3.4.3.

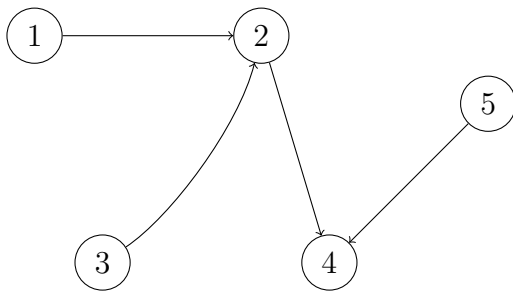


Figura 3.4.2:

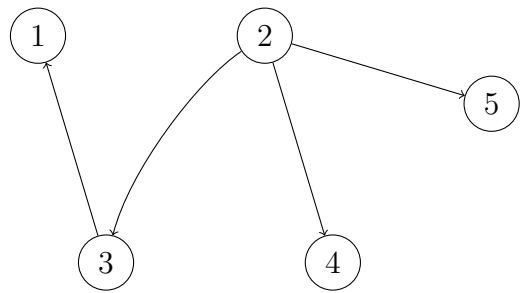


Figura 3.4.3:

Proprietățile teoretice și algoritmice de bază ale arborilor vor fi evidențiate în capitolul următor.

3.5 Parcurgerea grafurilor

Prin parcurgerea unui graf se înțelege o metodă sistematică de vizitare succesivă a nodurilor sale (în vederea prelucrării informațiilor atașate în structura de date modelată prin graful dat).

Definiția 3.5.1. Fie $G = (V, E)$ un graf și $x \in V$ un nod arbitrar fixat. **Parcurgerea în adâncime (DF, "depth first")** a grafului G pornind din nodul x , numit și **radăcină** a acestei parcurgeri, constă în:

- se vizitează nodul x , acesta devine nod curent;
- dacă nodul curent v_i are succesori direcți (adică noduri v_j pentru care există muchie sau arc de la v_i la v_j) nevizitați, atunci se vizitează primul astfel de nod v_j ; nodul v_j devine nod curent și se continuă procedeul de parcurgere pornind din acest nod;
- dacă nodul curent v_j nu mai are succesori direcți nevizitați, atunci se revine la nodul predecesor direct v_i (cel din care a fost vizitat); nodul v_i redevine nod curent și se continuă procedeul de parcurgere pornind din acest nod;
- dacă nodul curent nu mai are nici succesori direcți nevizitați, nici predecesor direct (deci este chiar radăcina x), atunci parcurgerea se încheie.

Observația 3.5.1. Pentru parcurgerea DF, considerând câte o muchie sau un arc de la fiecare nod curent v_i la primul său succesori direct nevizitat v_j (care va deveni următorul nod curent) se obține un arbore, numit **arbore DF**.

Exemplul 3.5.1. Pentru graful din Exemplul 3.1.2, parcurgerea în adâncime pornind din nodul 2 este

$$DF(2) : 2, 3, 1, 4, 5$$

(considerând că ordinea dintre succesorii direcți ai fiecărui nod este ordinea crescătoare). Arborele DF corespunzător acestei parcurgeri este reprezentat în Figura 3.5.1.

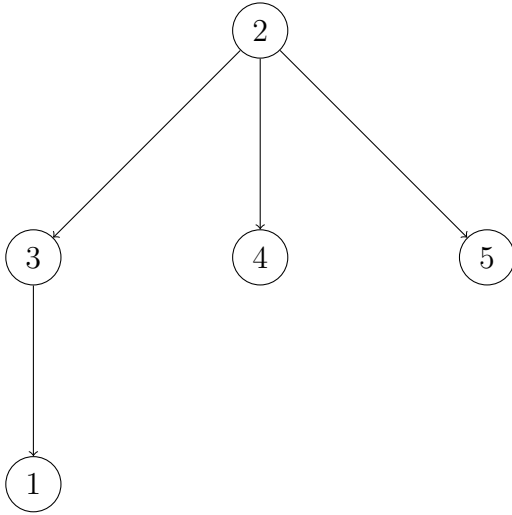


Figura 3.5.1:

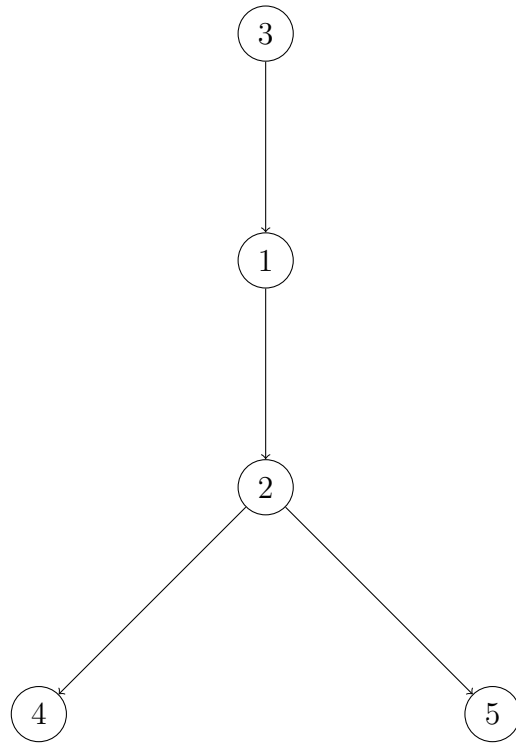


Figura 3.5.2:

Pentru același graf, parcurgerea DF pornind din nodul 3 este

$$DF(3) : 3, 1, 2, 4, 5,$$

iar arborele DF corespunzător este reprezentat în Figura 3.5.2.

Prezentăm în continuare doi algoritmi, unul recursiv și altul nerecursiv, pentru implementarea parcurgerii în adâncime.

Algoritmul 3.5.1 (parcurgerea DF, recursiv). Fie graful $G = (V, E)$ având mulțimea de noduri $V = \{1, \dots, n\}$ și matricea de adiacență $A = (a_{ij})_{i,j=1,n}$. Fie $x \in V$ un nod arbitrar fixat. Pentru implementarea parcurgerii $DF(x)$ vom utiliza un vector VIZ având semnificația

$$VIZ[i] = \begin{cases} 1, & \text{dacă nodul } i \text{ a fost vizitat,} \\ 0, & \text{în caz contrar,} \end{cases} \quad \forall i \in \{1, \dots, n\}.$$

Pentru memorarea arborelui $DF(x)$ vom utiliza un vector $TATA$ având semnificația

$$TATA[i] = \begin{cases} 0, & \text{dacă } i = x \text{ (rădăcina),} \\ \text{predecesorul direct al lui } i, & \text{dacă } i \neq x, \end{cases}$$

pentru orice nod i din parcurgerea DF .

Descrierea în pseudocod a algoritmului recursiv de parcurgere în adâncime pornind din nodul x are forma

```

DF_RECURSIV( $x$ ):
VIZITEAZĂ( $x$ );                                // se vizitează  $x$ , de exemplu se afișează  $x$ 
 $VIZ[x] \leftarrow 1$ ;                             //  $x$  a fost vizitat
for  $y = \overline{1, n}$  do
    if ( $a[x, y] \geq 1$ ) and ( $VIZ[y] = 0$ ) then
        //  $y$  este primul succesor direct nevizitat al lui  $x$ 
         $TATA[y] \leftarrow x$ ;
        DF_RECURSIV( $y$ );                          // se continuă parcurgerea DF
                                                    // din nodul  $y$ 

```

Algoritmul 3.5.2 (parcurgerea DF, nerecursiv). Fie din nou $G = (V, E)$ un graf având mulțimea de noduri $V = \{1, \dots, n\}$ și matricea de adiacență $A = (a_{ij})_{i,j=\overline{1,n}}$. Fie $x \in V$ un nod arbitrar fixat. Pentru implementarea nerecursivă a parcurgerii $DF(x)$ vom utiliza vectorii VIZ și $TATA$ cu aceleași semnificații ca în algoritmul anterior, un vector URM cu semnificația

$URM[i] =$ următorul succesor direct al nodului i ,

și o structură de tip **stivă** S , memorată ca un vector, ce conține nodurile vizitate și în curs de prelucrare, adică de vizitare a tuturor succesorilor.

Descrierea în pseudocod a algoritmului are forma

```

DF( $x$ ):
VIZITEAZĂ( $x$ );                                // se vizitează  $x$ , de exemplu se afișează  $x$ 
 $VIZ[x] \leftarrow 1$ ;                             //  $x$  a fost vizitat
 $TATA[x] \leftarrow 0$ ;
 $varf \leftarrow 1$ ;  $S[varf] \leftarrow x$ ;           //  $x$  se introduce în vârful stivei
while ( $varf > 0$ ) do                           // stiva este nevidă
     $i \leftarrow S[varf]$ ;                          //  $i$  este nodul din vârful stivei
     $j \leftarrow URM[i] + 1$ ;                        //  $j$  va fi următorul succesor direct
                                                    // nevizitat al lui  $i$ , dacă există
    while ( $a[i, j] = 0$ ) and ( $j \leq n$ ) do  $j \leftarrow j + 1$ ;
    if ( $j > n$ ) then
        // nodul  $i$  nu mai are succesorii direcți nevizitați
         $varf \leftarrow varf - 1$ ;                  // s-a încheiat prelucrarea lui  $i$ 
                                                    // și îl eliminăm din stivă
    else
         $URM[i] \leftarrow j$ ;                      //  $j$  este următorul succesor direct
                                                    // al lui  $i$ 
        if ( $VIZ[j] = 0$ ) then                      //  $j$  nu a fost vizitat
            VIZITEAZĂ( $j$ );                          // se vizitează  $j$ 
             $VIZ[j] \leftarrow 1$ ;                    //  $j$  a fost vizitat
             $TATA[j] \leftarrow i$ ;
             $varf \leftarrow varf + 1$ ;
             $S[varf] \leftarrow j$ ;                    // se introduce  $j$  în vârful stivei

// stiva este vidă, nu mai există noduri neprelucrate,
// parcurgerea este încheiată.

```

Observația 3.5.2. Pentru un graf cu n noduri și m muchii sau arce, implementarea anterioară a parcurgerii DF are complexitatea $O(n^2)$, deoarece oricare din cele n noduri este vizitat (deci introdus

și extras din stivă) cel mult câte o dată, iar căutarea succesorilor direcți j nevizitați ai fiecărui nod i extras din stivă se efectuează în cel mult n pași, prin parcurgerea liniei i din matricea de adiacență.

Reamintim notația utilizată, pentru orice funcție $f : \mathbb{N} \rightarrow \mathbb{N}$,

$$O(f) = \{g | g : \mathbb{N} \rightarrow \mathbb{N}, \exists r > 0, \exists n_0 \in \mathbb{N} \text{ a.î. } g(n) \leq rf(n) \forall n \geq n_0\}.$$

Dacă graful este memorat prin intermediul listelor de adiacență, atunci cautarea succesorilor direcți j nevizitați ai fiecărui nod i extras din stivă se efectuează, prin parcurgerea lor succesivă, în exact $d(i)$ (pentru graf neorientat) sau $d^+(i)$ (pentru graf orientat) pași. Cum, conform Propoziției 3.3.2, $\sum_{x \in V} d(x) = 2m$ și $\sum_{x \in V} d^+(x) = m$, obținem că în acest caz parcurgerea DF are complexitatea $O(n+m)$.

Observația 3.5.3. Dacă $G = (V, E)$ este un graf neorientat, atunci pentru orice nod $x \in V$ componenta conexă a nodului x este (indusă de) chiar mulțimea nodurilor vizitate prin parcurgerea $DF(x)$.

Pe baza acestei observații obținem următorul algoritm.

Algoritmul 3.5.3 (determinarea componentelor conexe). Fie din nou $G = (V, E)$ un graf neorientat, $V = \{1, \dots, n\}$ și fie $A = (a_{ij})_{i,j=\overline{1,n}}$ matricea de adiacență a grafului G . Pentru determinarea componentelor conexe ale grafului G vom utiliza un vector CC cu semnificația

$$CC[i] = \text{numărul componentei conexe în care se află nodul } i,$$

$\forall i \in \{1, \dots, n\}$ și o variabilă nrc ce reprezintă numărul de componente conexe. Evident, graful G este conex dacă și numai dacă valoarea finală a variabilei nrc este egală cu 1.

Descrierea în pseudocod a algoritmului are forma

COMPONENTE_CONEXE :

```

nrc ← 0;
for i =  $\overline{1, n}$  do CC[i] ← 0;
for i =  $\overline{1, n}$  do
    if (CC[i] = 0) then
        nrc ← nrc + 1;
        DF(i);
        // nodul i nu a fost vizitat
        // nodurile din parcurgerea DF(i)
        // vor forma o nouă componentă conexă

```

unde funcția **DF**(i) este cea din Algoritmul 3.5.1 sau cea din Algoritmul 3.5.2, adăugând instrucțiunea $CC[i] \leftarrow nrc$ în funcția **VIZITEAZĂ**(i).

Observația 3.5.4. Algoritmul anterior poate fi utilizat și pentru determinarea componentelor conexe ale unui graf orientat, înlocuind condiția " $a_{xy} \geq 1$ " din funcția **DF_RECURSIV**(x) cu " $a_{xy} \geq 1$ sau $a_{yx} \geq 1$ ", respectiv condiția " $a_{ij} = 0$ " din funcția **DF**(x) cu " $a_{ij} = 0$ și $a_{ji} = 0$ " (deoarece în determinarea componentelor conexe nu se ține cont de orientarea arcelor).

Observația 3.5.5. Dacă $G = (V, E)$ este un graf orientat, atunci pentru orice nod $x \in V$ componenta tare-conexă a nodului x este (indusă de) mulțimea nodurilor y vizitate prin parcurgerea $DF(x)$ a grafului G cu proprietatea că y este vizitat și în parcurgerea $DF(x)$ a grafului

$$\overline{G} = (V, \overline{E}), \text{ unde } \overline{E} = \{(j, i) | (i, j) \in E\},$$

numit **transpusul (simetricul)** lui G . Evident, orice drum (y, v_1, \dots, v_k, x) în graful G corespunde drumului (x, v_k, \dots, v_1, y) în graful \overline{G} , deci x și y sunt în aceeași componentă tare-conexă a lui G dacă și numai dacă există un drum de la x la y în G și un drum de la x la y în \overline{G} , adică y este vizitat prin parcurgerea $DF(x)$ atât în G cât și în \overline{G} .

Matricea de adiacență a grafului \overline{G} este transpusa matricei de adiacență a grafului G , deci pentru parcurgerea $DF(x)$ a grafului \overline{G} putem utiliza tot matricea de adiacență A a grafului dat G , înlocuind condiția " $a_{ij} = 0$ " cu " $a_{ji} = 0$ ", iar condiția " $a_{xy} \geq 1$ " cu " $a_{yx} \geq 1$ " în funcțiile $\mathbf{DF}(x)$, respectiv $\mathbf{DF_RECURSIV}(x)$. Astfel, algoritmul DF poate fi utilizat și pentru determinarea componentelor tare-conexe, deci și pentru verificarea tare-conexității grafului.

Definiția 3.5.2. Fie $G = (V, E)$ un graf și $x \in V$ un nod arbitrar fixat. **Parcurgerea în lățime** (**BF**, "**breadth first**", **parcurgerea pe nivele**) a grafului G pornind din nodul x , numit și **rădăcină** a acestei parcurgeri, constă în:

- se vizitează nodul x , considerat nod de nivelul zero;
- se vizitează apoi succesorii direcți nevizitați ai acestuia (diferiți de x), considerați noduri de nivelul 1;
- se vizitează apoi, pe rând, succesorii direcți nevizitați ai acestora, considerați noduri de nivelul 2;
ș.a.m.d.;
- parcurgerea se încheie când niciun nod de pe un nivel nu mai are succesori direcți nevizitați.

Observația 3.5.6. Analog parcurgerii DF, considerând câte o muchie sau un arc de la fiecare nod curent v al parcurgerii BF la fiecare din nodurile nevizitate (de pe următorul nivel) pentru care v este predecesorul direct, se obține un arbore, numit **arbore BF**.

Exemplul 3.5.2. Pentru graful din Exemplul 3.1.2, parcurgerea în lățime pornind din nodul 2 este

$$BF(2) : 2, 3, 4, 5, 1$$

(considerând din nou ordinea dintre succesorii direcți ai fiecărui nod ca fiind ordinea crescătoare). Arborele BF corespunzător acestei parcurgeri este reprezentat în Figura 3.5.3.

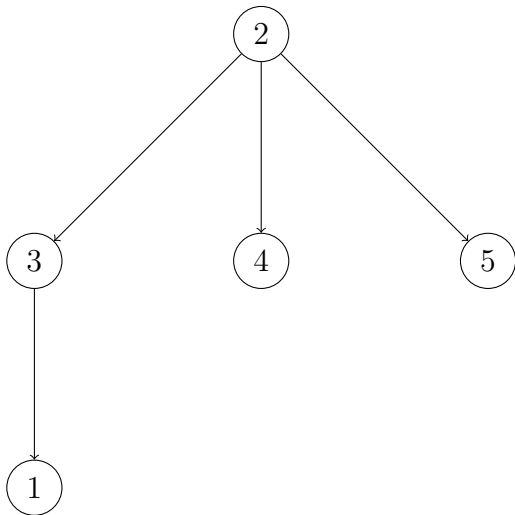


Figura 3.5.3:

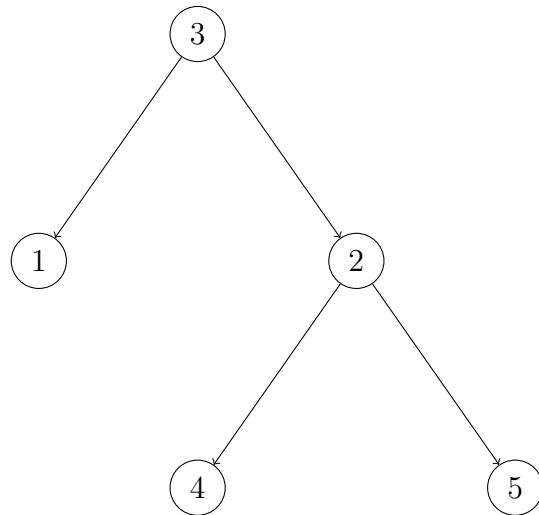


Figura 3.5.4:

Pentru același graf, parcurgerea BF pornind din nodul 3 este

$$BF(3) : 3, 1, 2, 4, 5,$$

iar arborele BF corespunzător este reprezentat în Figura 3.5.4.

Algoritmul 3.5.4 (parcurgerea BF). Fie graful $G = (V, E)$ având mulțimea de noduri $V = \{1, \dots, n\}$ și matricea de adiacență $A = (a_{ij})_{i,j=1,n}$. Fie $x \in V$ un nod arbitrar fixat. Pentru implementarea parcurgerii $BF(x)$ vom utiliza vectorii VIZ , $TATA$, URM și S având aceleași semnificații ca la parcurgerea $DF(x)$ (Algoritmul 3.5.2), cu deosebirea că vectorul S al nodurilor vizitate și în curs de prelucrare este organizat și utilizat acum ca o structură de tip **coadă**.

Aceasta fiind singura modificare față de Algoritmul 3.5.2, obținem următoarea descriere în pseudocod a algoritmului de parcurgere în lățime

```

BF( $x$ ) :
  VIZITEAZĂ( $x$ );
   $VIZ[x] \leftarrow 1$ ;
   $TATA[x] \leftarrow 0$ ;
   $coada \leftarrow 1$ ;                                // nodurile se adaugă la  $S$  pe poziția " $coada$ "
   $varf \leftarrow 1$ ;                                // și se elimină de pe poziția " $varf$ "
   $S[coada] \leftarrow x$ ;
  while ( $varf \leq coada$ ) do                        // coada este nevidă
  |    $i \leftarrow S[varf]$ ;
  |    $j \leftarrow URM[i] + 1$ ;
  |   while ( $a[i, j] = 0$ ) and ( $j \leq n$ ) do  $j \leftarrow j + 1$ ;
  |   if ( $j > n$ ) then
  |   |    $varf \leftarrow varf + 1$ ;
  |   else
  |   |    $URM[i] \leftarrow j$ ;
  |   |   if ( $VIZ[j] = 0$ ) then
  |   |   |   VIZITEAZĂ( $j$ );
  |   |   |    $VIZ[j] \leftarrow 1$ ;
  |   |   |    $TATA[j] \leftarrow i$ ;
  |   |   |    $coada \leftarrow coada + 1$ ;
  |   |   |    $S[coada] \leftarrow j$ ;
  |   |   end if;
  |   end if;
  |    $varf \leftarrow varf + 1$ ;
  end while

```

Observația 3.5.7. Analog parcurgerii DF, implementarea anterioară a parcurgerii BF are complexitatea $O(n^2)$, iar dacă graful este memorat prin intermediul listelor de adiacență, atunci complexitatea este $O(n + m)$.

Observația 3.5.8. Observațiile 3.5.3, 3.5.4 și 3.5.5 și algoritmii corespunzători rămân valabile dacă înlocuim parcurgerea DF cu parcurgerea BF.

3.6 Algoritmul Roy-Warshall

Definiția 3.6.1. Fie $G = (V, E)$ un graf (neorientat sau orientat), unde $V = \{v_1, \dots, v_n\}$. **Matricea drumurilor** asociată grafului G este matricea $D = (d_{ij})_{i,j=1,n}$ definită prin

$$d_{ij} = \begin{cases} 1, & \text{dacă } \exists \mu = (v_i, \dots, v_j) \text{ drum cu } l(\mu) > 0, \\ 0, & \text{în caz contrar,} \end{cases}$$

unde $l(\mu)$ reprezintă lungimea drumului μ .

Exemplul 3.6.1. Matricea drumurilor asociată grafului neorientat din Exemplul 3.1.1 este

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

iar matricea drumurilor asociată grafului orientat din Exemplul 3.1.2 este

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

Observația 3.6.1. Evident, orice graf neorientat are matricea drumurilor simetrică.

Observația 3.6.2. Conform Observației 3.4.5, pentru $i \neq j$ putem înlocui termenul de "drum" cu "drum elementar" în definiția matricei drumurilor.

Următorul algoritm determină matricea drumurilor unui graf pornind de la matricea de adiacență.

Algoritmul 3.6.1 (Roy-Warshall). Fie $G = (V, E)$ un graf simplu cu $V = \{v_1, \dots, v_n\}$ și fie $A = (a_{ij})_{i,j=\overline{1,n}}$ matricea sa de adiacență (având toate elementele 0 sau 1). Se calculează matricele

$$D^{(k)} = (d_{ij}^{(k)})_{i,j=\overline{1,n}}, \quad k \in \{0, 1, \dots, n\},$$

definite prin

$$D^{(0)} = A, \tag{3.6.1}$$

$$d_{ij}^{(k)} = d_{ij}^{(k-1)} \vee d_{ik}^{(k-1)} d_{kj}^{(k-1)}, \quad \forall k, i, j \in \{1, \dots, n\}, \tag{3.6.2}$$

unde $0 \vee 0 = 0$, $0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1$ (operația de disjuncție, "sau").

Teorema 3.6.1 (de corectitudine a Algoritmului Roy-Warshall). În contextul Algoritmului Roy-Warshall, ultima matrice calculată este chiar matricea drumurilor asociată grafului G , adică

$$D^{(n)} = D.$$

Demonstrație. Vom demonstra prin inducție după $k \in \{0, 1, \dots, n\}$ că pentru orice $i, j \in \{1, \dots, n\}$ avem

$$d_{ij}^{(k)} = \begin{cases} 1, & \text{dacă } \exists \mu = (v_i, \dots, v_j) \text{ drum cu } l(\mu) > 0 \text{ și } I(\mu) \subseteq \{v_1, \dots, v_k\}, \\ 0, & \text{în caz contrar,} \end{cases} \tag{3.6.3}$$

unde $l(\mu)$ reprezintă lungimea drumului μ , $I(\mu)$ reprezintă mulțimea nodurilor intermediare ale drumului μ , iar $\{v_1, \dots, v_k\}$ reprezintă mulțimea $\{v_i \mid 1 \leq i \leq k\}$, deci pentru $k = 0$ această mulțime este \emptyset .

Pentru $k = 0$ avem

$$d_{ij}^{(0)} = a_{ij} \text{ (conform (3.6.1)) și } a_{ij} = \begin{cases} 1, & \text{dacă } (v_i, v_j) \in E, \\ 0, & \text{în caz contrar} \end{cases}$$

(din definiția matricei de adiacență), iar $(v_i, v_j) \in E$ dacă și numai dacă $\exists \mu = (v_i, \dots, v_j)$ drum cu $l(\mu) > 0$ și $I(\mu) = \emptyset$, deci obținem egalitatea (3.6.3).

Presupunem acum egalitatea (3.6.3) adevărată pentru $k-1$ ($1 \leq k \leq n$) și o demonstrăm pentru k . Folosind (3.6.2), definiția operațiilor \vee și \cdot și ipoteza de inducție avem echivalențele:

$$\begin{aligned} d_{ij}^{(k)} = 1 &\Leftrightarrow d_{ij}^{(k-1)} = 1 \text{ sau } d_{ik}^{(k-1)} = d_{kj}^{(k-1)} = 1 \Leftrightarrow \exists \mu = (v_i, \dots, v_j) \text{ drum cu} \\ l(\mu) > 0, I(\mu) &\subseteq \{v_1, \dots, v_{k-1}\} \text{ sau } \exists \mu_1 = (v_i, \dots, v_k), \mu_2 = (v_k, \dots, v_j) \\ \text{drumuri cu } l(\mu_1), l(\mu_2) &> 0, I(\mu_1), I(\mu_2) \subseteq \{v_1, \dots, v_{k-1}\} \\ &\Leftrightarrow \exists \mu = (v_i, \dots, v_j) \text{ cu } l(\mu) > 0, I(\mu) \subseteq \{v_1, \dots, v_k\}, v_k \notin I(\mu) \text{ sau} \\ \exists \mu' = (v_i, \dots, v_k, \dots, v_j) &\text{ cu } l(\mu') > 0, I(\mu') \subseteq \{v_1, \dots, v_k\}, v_k \in I(\mu') \end{aligned}$$

(μ' se obține parcurgând întâi μ_1 apoi μ_2 și, reciproc, μ_1, μ_2 sunt porțiunile din μ' dintre v_i și prima apariție a lui v_k în $I(\mu')$, respectiv dintre ultima apariție a lui v_k în $I(\mu')$ și v_j)

$$\Leftrightarrow \exists \mu = (v_i, \dots, v_j) \text{ drum cu } l(\mu) > 0 \text{ și } I(\mu) \subseteq \{v_1, \dots, v_k\}.$$

Demonstrația prin inducție a egalității (3.6.3) este astfel încheiată.

Pentru $k = n$ condiția $I(\mu) \subseteq \{v_1, \dots, v_n\}$ poate fi eliminată, fiind întotdeauna adevărată, deci din (3.6.3) și Definiția 3.6.1 obținem că

$$d_{ij}^{(n)} = d_{ij}, \quad \forall i, j \in \{1, \dots, n\},$$

adică egalitatea din enunț. □

Observația 3.6.3. Pentru n fixat, Algoritmul Roy-Warshall necesită $2n^3$ operații (câte o operație \cdot și \vee pentru fiecare $(k, i, j) \in \{1, \dots, n\} \times \{1, \dots, n\} \times \{1, \dots, n\}$), deci acest algoritm are complexitatea $\Theta(n^3)$. Reamintim notația utilizată, pentru orice funcție $f: \mathbb{N} \rightarrow \mathbb{N}$,

$$\Theta(f) = \{g|g: \mathbb{N} \rightarrow \mathbb{N}, \exists r_1, r_2 > 0, \exists n_0 \in \mathbb{N} \text{ a.î. } r_1 f(n) \leq g(n) \leq r_2 f(n) \forall n \geq n_0\}.$$

Observația 3.6.4. Algoritmul Roy-Warshall rămâne evident valabil și pentru grafuri nesimple, cu modificarea

$$d_{ij}^{(0)} = \begin{cases} 1, & \text{dacă } a_{ij} > 0, \\ 0, & \text{dacă } a_{ij} = 0, \end{cases} \quad \forall i, j \in \{1, \dots, n\}.$$

Exemplul 3.6.2. Pentru graful din Exemplul 3.1.2, prin aplicarea Algoritmului Roy-Warshall obținem matricele:

$$D^{(0)} = A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad (\text{matricea de adiacență});$$

$$D^{(1)} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}; \quad D^{(2)} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

(deoarece, de exemplu, $d_{13}^{(2)} = d_{13}^{(1)} \vee d_{12}^{(1)} d_{23}^{(1)} = 0 \vee 1 \cdot 1 = 0 \vee 1 = 1$);

$$D^{(3)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}; \quad D^{(4)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix};$$

$$D^{(5)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} = D \text{ (matricea drumurilor).}$$

Observația 3.6.5. Conform ecuațiilor (3.6.3), Algoritmul Roy-Warshall este un algoritm specific **metodei programării dinamice**.

Conform ecuațiilor (3.6.2) și definiției operației \vee avem $d_{ik}^{(k)} = d_{ik}^{(k-1)}$ și $d_{kj}^{(k)} = d_{kj}^{(k-1)}$, $\forall k, i, j \in \{1, \dots, n\}$, deci pentru implementarea algoritmului putem utiliza o singură matrice D . Astfel obținem următoarea descriere în pseudocod a algoritmului

ROY_WARSHALL:

```

for  $i = \overline{1, n}$  do                                     // inițializări
    for  $j = \overline{1, n}$  do
         $d[i, j] \leftarrow a[i, j]$ ;
    for  $k = \overline{1, n}$  do
        for  $i = \overline{1, n}$  do
            for  $j = \overline{1, n}$  do
                 $d[i, j] \leftarrow d[i, j] \vee d[i, k]d[k, j]$ ;

```

Pentru grafuri oarecare (simple sau nesimple) inițializarea are forma

```

for  $i = \overline{1, n}$  do                                     // inițializări
    for  $j = \overline{1, n}$  do
        if  $(a[i, j] > 0)$  then
             $d[i, j] \leftarrow 1$ ;
        else
             $d[i, j] \leftarrow 0$ ;

```

Observația 3.6.6. Dacă $G = (V, E)$ este un graf neorientat și $V = \{1, \dots, n\}$, atunci pentru orice nod $x \in V$ componenta conexă a nodului x este (indusă de) mulțimea $\{x\} \cup \{y | y \in V \setminus \{x\}, d_{xy} = 1\}$.

Astfel Algoritmul Roy-Warshall poate fi utilizat pentru determinarea componentelor conexe și pentru verificarea conexității grafului.

Algoritmul 3.6.2 (determinarea componentelor tare-conexe). Dacă $G = (V, E)$ este un graf orientat și $V = \{1, \dots, n\}$, atunci pentru orice nod $x \in V$ componenta tare-conexă a nodului x este (indusă de) mulțimea

$$\{x\} \cup \{y | y \in V \setminus \{x\}, d_{xy} = d_{yx} = 1\},$$

deci Algoritmul Roy-Warshall poate fi utilizat pentru determinarea componentelor tare-conexe și pentru verificarea tare-conexității grafului. Descrierea în pseudocod a algoritmului are forma

COMPONENTE_TARE_CONEXE:

```

nrc ← 0;                                     // numărul de componente tare-conexe
for i =  $\overline{1, n}$  do
    [ CTC[i] ← 0;                             // vectorul componentelor tare-conexe
      // CTC[i] = numărul componentei tare-conexe în care se află nodul i
    ]

```

ROY_WARSHALL;

```

for i =  $\overline{1, n}$  do
    [ if (CTC[i] = 0) then
      [ nrc ← nrc + 1;
        CTC[i] ← nrc;
        for j =  $\overline{i + 1, n}$  do
            [ if (CTC[j] = 0) and (d[i, j] = 1) and (d[j, i] = 1) then
              [ CTC[j] ← nrc;
            ]
          ]
        ]
    ]

```

Tema 4

Arbori și arborescențe

4.1 Numărul cicromatic

Propoziția 4.1.1 (Dirac). Fie $G = (V, E)$ un graf neorientat simplu și

$$\delta(G) = \min_{x \in V} d(x)$$

gradul minim al nodurilor din G .

a) G conține un lanț deschis elementar μ astfel încât

$$l(\mu) \geq \delta(G),$$

unde $l(\mu)$ reprezintă lungimea lanțului μ .

b) Dacă $\delta(G) \geq 2$, atunci G conține un ciclu elementar C astfel încât

$$l(C) \geq \delta(G) + 1.$$

Definiția 4.1.1. Un graf se numește **par** dacă toate nodurile sale au gradele pare.

Propoziția 4.1.2 (Veblen). Fie $G = (V, E)$ un graf cu $E \neq \emptyset$. Atunci graful G este par dacă și numai dacă există C_1, \dots, C_k cicluri elementare muchie-disjuncte (două câte două) astfel încât

$$E = E(C_1) \cup \dots \cup E(C_k) \quad (k \in \mathbb{N}^*), \quad (4.1.1)$$

unde $E(C_i)$ reprezintă mulțimea muchiilor ciclului C_i , $\forall i \in \{1, \dots, k\}$.

Propoziția 4.1.3. Fie $G = (V, E)$ un graf. Atunci $(\mathcal{P}(E), \Delta, \cdot)$ este un spațiu vectorial peste corpul finit cu două elemente $(\mathbb{Z}_2, +, \cdot)$, unde

$$\mathcal{P}(E) = \{F \mid F \subseteq E\}, \quad A \Delta B = (A \setminus B) \cup (B \setminus A) \text{ (diferența simetrică),}$$

$$\widehat{0} \cdot A = \emptyset, \quad \widehat{1} \cdot A = A, \quad \forall A \in \mathcal{P}(E).$$

Definiția 4.1.2. Spațiul vectorial $(\mathcal{P}(E), \Delta, \cdot)$ din propoziția anterioară se numește **spațiul muchiilor** grafului G .

Propoziția 4.1.4. Fie $G = (V, E)$ un graf și

$$\mathcal{C}(E) = \{F \subseteq E \mid (V, F) = \text{graf par}\}.$$

Atunci $\mathcal{C}(E)$ este un subspațiu vectorial al spațiului muchiilor grafului G .

Propoziția 4.1.5. Fie $G = (V, E)$ un graf și

$$\mathcal{C}_0(E) = \{F \subseteq E \mid \exists C = \text{ciclu elementar în } G \text{ a.î. } F = E(C)\},$$

unde $E(C)$ este mulțimea muchiilor ciclului C . Atunci subspațiul vectorial al spațiului muchiilor lui G generat de $\mathcal{C}_0(E)$ este subspațiul $\mathcal{C}(E)$ din propoziția anterioară.

Demonstrație. Evident, $\mathcal{C}_0(E) \subseteq \mathcal{C}(E)$. Conform Propoziției 4.1.2, pentru orice $F \in \mathcal{C}(E)$, $F \neq \emptyset$, există $F_1, \dots, F_k \in \mathcal{C}_0(E)$ disjuncte două câte două a.î. $F = F_1 \cup \dots \cup F_k$, deci $F = F_1 \Delta \dots \Delta F_k = \hat{1} \cdot F_1 \Delta \dots \Delta \hat{1} \cdot F_k$. \square

Definiția 4.1.3. Subspațiul $\mathcal{C}(E)$ din Propoziția 4.1.4 se numește **spațiul ciclurilor** grafului G . Dimensiunea acestui subspațiu se numește **numărul ciclomantic** al grafului G și se notează cu $\gamma(G)$.

Propoziția 4.1.6 (Listing). Fie $G = (V, E)$ o pădure cu n noduri, m muchii și k componente conexe. Atunci

$$m - n + k = 0.$$

Demonstrație. Demonstrăm egalitatea din enunț prin inducție după m .

Pentru $m = 0$ rezultă că toate nodurile sunt izolate, deci fiecare nod formează o componentă conexă, adică $k = n$ și egalitatea $m - n + k = 0$ este verificată.

Presupunem adevărată egalitatea din enunț pentru orice pădure cu $m - 1$ muchii ($m \geq 1$) și o demonstrăm pentru pădurea G cu m muchii. Fie

$$G' = (V, E'), \text{ unde } E' = E \setminus \{e\}, \text{ cu } e \in E, e = [x, y].$$

Evident G' este o pădure cu n noduri, $m - 1$ muchii și $k + 1$ componente conexe (deoarece componentele conexe din G ce nu conțin x și y rămân componente conexe și în G' , iar componenta conexă din G ce conține x și y se partiționează în două componente conexe în G' , una conținând pe x și cealaltă pe y). Aplicând ipoteza de inducție pentru graful G' avem

$$(m - 1) - n + (k + 1) = 0, \text{ deci } m - n + k = 0.$$

\square

Corolarul 4.1.1. Orice arbore cu n noduri are $m = n - 1$ muchii.

Demonstrație. Se aplică propoziția anterioară, cu $k = 1$. \square

Propoziția 4.1.7. Un graf este conex dacă și numai dacă are arbori parțiali.

Demonstrație. Dacă graful $G = (V, E)$ are un arbore parțial T , atunci pentru orice $x, y \in V$, $x \neq y$, există un lanț $[x, \dots, y]$ în T (fiind conex), deci și în G , și astfel G este conex.

Reciproc, demonstrăm că orice graf conex $G = (V, E)$ are arbori parțiali prin inducție după numărul p de cicluri elementare ale lui G .

Pentru $p = 0$ rezultă că însuși G este un arbore, deci arbore parțial în G .

Presupunem adevărată afirmația pentru orice graf conex cu cel mult $p - 1$ cicluri elementare ($p \geq 1$) și o demonstrăm pentru graful conex $G = (V, E)$ cu p cicluri elementare. Fie $e = [x, y]$ o muchie a lui G aflată pe un ciclu elementar C și fie $G' = (V, E \setminus \{e\})$. Graful G' rămâne conex (între x și y există lanțul $[x, v_1, \dots, v_k, y]$, unde $[x, v_1, \dots, v_k, y, x]$ este ciclul C) și are cel mult $p - 1$ cicluri elementare. Conform ipotezei de inducție, există un arbore parțial T în G' , deci și în G . \square

Propoziția 4.1.8. Fie $G = (V, E)$ un graf, $T = (V, F)$ un arbore parțial al lui G și $e \in E \setminus F$. Atunci graful parțial

$$T + e = (V, F \cup \{e\})$$

are un unic ciclu elementar și acest ciclu conține muchia e .

Demonstrație. Fie $e = [x, y]$. T fiind arbore parțial, există un lanț elementar $\mu = [x, v_1, \dots, v_k, y]$ în T , deci $\mu + e = [x, v_1, \dots, v_k, y, x]$ este un ciclu elementar în $T + e$.

Unicitatea acestui ciclu rezultă din unicitatea lanțului elementar μ (dacă ar exista două lanțuri elementare distincte $\mu_1 = [x, \dots, y]$ și $\mu_2 = [x, \dots, y]$ în T , porțiunile lor muchie-disjuncte ar produce un ciclu, contradicție cu $T = \text{arbore}$). \square

Teorema 4.1.1 (Teorema numărului ciclomatic). Orice graf $G = (V, E)$ cu n noduri, m muchii și k componente conexe are numărul ciclomatic

$$\gamma(G) = m - n + k.$$

Demonstrație. Demonstrăm egalitatea din enunț în două etape.

Etapa 1) Presupunem că graful G este conex, deci $k = 1$. Fie $T = (V, F)$ un arbore parțial al lui G (există, conform Propoziției 4.1.7). Conform Corolarului 4.1.1, $\text{card}(F) = n - 1$. Fie

$$E \setminus F = \{e_1, \dots, e_{m-n+1}\}.$$

Pentru orice $i \in \{1, \dots, m - n + 1\}$, fie C_i mulțimea muchiilor ciclului elementar unic din $T + e_i$ (conform Propoziției 4.1.8). Atunci

$$\mathcal{B} = \{C_1, \dots, C_{m-n+1}\}$$

este o bază a spațiului ciclurilor $\mathcal{C}(E)$ (numită **bază de cicluri** a grafului G). Rezultă că

$$\gamma(G) = \text{card}(\mathcal{B}) = m - n + 1 = m - n + k.$$

Etapa 2) Fie acum G un graf oarecare și G_1, \dots, G_k componentele sale conexe. Fie n_i și m_i numerele de noduri, respectiv muchii ale componentei G_i . Evident, $n = n_1 + \dots + n_k$ și $m = m_1 + \dots + m_k$.

Deoarece subspațiile ciclurilor componentelor G_1, \dots, G_k sunt liniar independente, avem

$$\gamma(G) = \gamma(G_1) + \dots + \gamma(G_k).$$

Dar, conform etapei 1), $\gamma(G_i) = m_i - n_i + 1 \ \forall i \in \{1, \dots, k\}$, și astfel prin adunare rezultă că $\gamma(G) = m - n + k$. \square

Observația 4.1.1. Demonstrația teoremei anterioare este constructivă, indicând următorul **algoritm de determinare a unei baze de cicluri** pentru graful G :

- se determină componentele conexe G_1, \dots, G_k ;
- pentru fiecare componentă G_i , se determină un arbore parțial T (de exemplu arborele DF sau BF) și ciclurile elementare $C_1, \dots, C_{m_i-n_i+1}$ din grafurile $T + e_i$, pentru fiecare muchie e_i din G_i ce nu aparține lui T ;
- ciclurile astfel determinate formează împreună o bază de cicluri pentru graful G .

Exemplul 4.1.1. Fie graful G reprezentat în Figura 4.1.1.

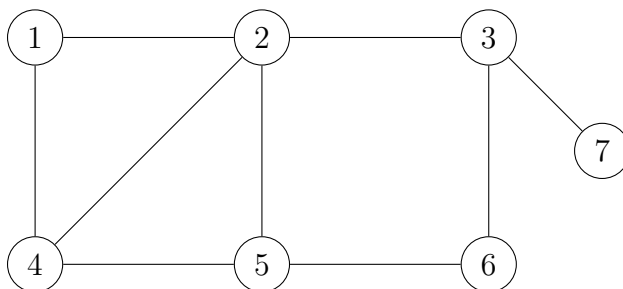


Figura 4.1.1:

Numărul său cicromatic este $\gamma(G) = m - n + k = 9 - 7 + 1 = 3$.

Considerăm arborele parțial T reprezentat în Figura 4.1.2.

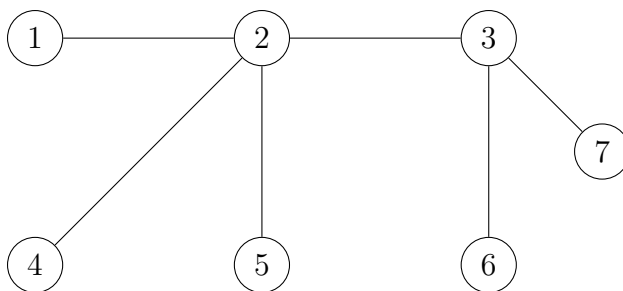


Figura 4.1.2:

Aplicând algoritmul din observația anterioară se obține baza de cicluri $\{C_1, C_2, C_3\}$ unde

$$C_1 = \{[1, 4], [4, 2], [2, 1]\}, C_2 = \{[4, 5], [5, 2], [2, 4]\} \text{ și } C_3 = \{[5, 6], [6, 3], [3, 2], [2, 5]\}.$$

4.2 Teorema de caracterizare a arborilor

Teorema 4.2.1 (de caracterizare a arborilor). Fie $G = (V, E)$ un graf cu n noduri. Următoarele afirmații sunt echivalente:

- 1) G este un arbore (adică este conex și fără cicluri);
- 2) G este fără cicluri și are $m = n - 1$ muchii;
- 3) G este conex și are $m = n - 1$ muchii;
- 4) G este fără cicluri și maximal cu această proprietate, adică dacă se adaugă o muchie între oricare două noduri graful obținut conține cicluri;
- 5) G este conex și minimal cu această proprietate, adică dacă se elimină o muchie oarecare graful obținut devine neconex;
- 6) între oricare două noduri ale lui G există un unic lanț elementar.

Demonstrație. 1) \Rightarrow 2) Fie G un arbore, adică și conex și fără cicluri. Deci $0 = \gamma(G) = m - n + 1$ și astfel $m = n - 1$.

2) \Rightarrow 3) Fie G fără cicluri și cu $m = n - 1$ muchii. Avem $0 = \gamma(G) = n - 1 - n + k$, unde k reprezintă numărul de componente conexe ale lui G , deci $k = 1$, adică G este conex.

3) \Rightarrow 4) Fie G conex și cu $m = n - 1$ muchii. Avem $\gamma(G) = n - 1 - n + 1 = 0$, deci G este fără cicluri. Fie $G + e = (V, E \cup \{e\})$, $e \notin E$. Graful $G + e$ rămâne conex și are n noduri și $m + 1 = n$ muchii. Astfel $\gamma(G + e) = n - n + 1 = 1$, adică $G + e$ conține un ciclu (elementar).

4) \Rightarrow 5) Fie G fără cicluri maximal. Dacă G nu ar fi conex, atunci adăugând o muchie între două noduri x și y din componente conexe diferite ale lui G s-ar obține un graf fără cicluri (deoarece nu există lanț între x și y în G), contradicție cu ipoteza. Deci G este conex.

Fie $G - e = (V, E \setminus \{e\})$, $e \in E$, $e = [x, y]$. Dacă graful $G - e$ ar fi conex, atunci ar exista un lanț elementar $[x, v_1, \dots, v_k, y]$ în $G - e$ și astfel $[x, v_1, \dots, v_k, y, x]$ ar fi un ciclu în G , contradicție cu ipoteza. Deci $G - e$ este neconex.

5) \Rightarrow 6) Fie G conex minimal. Fie $x, y \in V$.

Dacă $x \neq y$, rezultă că există un lanț elementar $[x, \dots, y]$.

Dacă $x = y$ există de asemenea lanțul elementar $[x]$ de lungime zero.

Dacă între nodurile x și y ar exista două lanțuri elementare distincte $\mu_1 = [x = v_1, v_2, \dots, v_i = y]$ și $\mu_2 = [x = w_1, w_2, \dots, w_j = y]$, atunci notând cu $e = [v_k, v_{k+1}]$ prima muchie a lanțului μ_1 ce nu aparține lanțului μ_2 , graful $G - e = (V, E \setminus \{e\})$ ar rămâne conex (deoarece între v_k și v_{k+1} avem lanțul $[v_k, v_{k-1}, \dots, v_1 = x = w_1, w_2, \dots, w_j = y = v_i, v_{i-1}, \dots, v_{k+1}]$), contradicție cu ipoteza. Deci între x și y există un unic lanț elementar.

6) \Rightarrow 1) Fie G cu proprietatea că între orice două noduri există un unic lanț elementar. Evident, G este conex. Dacă G ar avea un ciclu elementar $C = [x, v_1, \dots, v_k, x]$ atunci $[x, v_1]$ și $[v_1, \dots, v_k, x]$ ar fi două lanțuri elementare între x și v_1 , contradicție cu ipoteza. Deci G nu are cicluri elementare, și nici neelementare (deoarece orice ciclu neelementar s-ar descompune în cicluri elementare muchie-disjuncte). \square

Observația 4.2.1. Un arbore de ordinul 1 este format dintr-un singur nod izolat (de grad zero), iar arbore de ordin $n \geq 2$ nu are noduri izolate (fiind conex), deci are gradele nodurilor mai mari sau egale cu 1.

Propoziția 4.2.1 (de caracterizare a secvenței gradelor nodurilor unui arbore). Fie $d_1, d_2, \dots, d_n \in \mathbb{N}^*$, unde $n \in \mathbb{N}$, $n \geq 2$. Atunci numerele d_1, d_2, \dots, d_n sunt gradele nodurilor unui arbore de ordinul n dacă și numai dacă

$$\sum_{i=1}^n d_i = 2n - 2.$$

Demonstrație. " \Rightarrow " Fie $T = (V, E)$ un arbore cu $V = \{v_1, \dots, v_n\}$ astfel încât $d_T(v_i) = d_i$, $\forall i \in \{1, \dots, n\}$. Conform Corolarului 4.1.1 rezultă că T are $m = n - 1$ muchii, deci aplicând Propoziția 3.3.2 avem

$$\sum_{i=1}^n d_i = 2m = 2n - 2.$$

" \Leftarrow " Demonstrăm că dacă $\sum_{i=1}^n d_i = 2n - 2$ cu $d_1, d_2, \dots, d_n \in \mathbb{N}^*$, $n \in \mathbb{N}$, $n \geq 2$, atunci există un arbore $T = (V_n, E)$ cu $V_n = \{v_1, \dots, v_n\}$ astfel încât $d_T(v_i) = d_i$, $\forall i \in \{1, \dots, n\}$, prin inducție după n .

Pentru $n = 2$ rezultă că $d_1 + d_2 = 2$, deci $d_1 = d_2 = 1$. Luând $E = \{[v_1, v_2]\}$ rezultă că $T = (\{v_1, v_2\}, E)$ este un arbore (fiind conex și având $2 - 1 = 1$ muchii) și $d_T(v_i) = 1 = d_i$, $\forall i \in \{1, 2\}$.

Presupunem afirmația adevărată pentru $n - 1$ și o demonstrăm pentru n , unde $n \in \mathbb{N}$, $n \geq 3$. Fie $\sum_{i=1}^n d_i = 2n - 2$, $d_1, d_2, \dots, d_n \in \mathbb{N}^*$. Presupunem că $d_1 \geq d_2 \geq \dots \geq d_n$. Atunci $d_n = 1$ (deoarece dacă $d_n \geq 2$ ar rezulta că $\sum_{i=1}^n d_i \geq 2n > 2n - 2$, fals) și $d_1 \geq 2$ (deoarece dacă $d_1 = 1$ ar rezulta că $\sum_{i=1}^n d_i = n < 2n - 2$, fals). Fie

$$d'_i = \begin{cases} d_1 - 1, & \text{dacă } i = 1, \\ d_i, & \text{dacă } i \in \{2, \dots, n-1\}. \end{cases}$$

Atunci $d'_1, d'_2, \dots, d'_{n-1} \in \mathbb{N}^*$ și

$$\sum_{i=1}^{n-1} d'_i = d_1 - 1 + \sum_{i=2}^{n-1} d_i = \sum_{i=1}^n d_i - 2 = 2n - 4 = 2(n-1) - 2,$$

deci, conform ipotezei de inducție, există un arbore $T' = (V_{n-1}, E')$ cu $V_{n-1} = \{v_1, \dots, v_{n-1}\}$ astfel încât $d_{T'}(v_i) = d'_i$, $\forall i \in \{1, \dots, n-1\}$. Fie graful

$$T = (V_n, E' \cup \{[v_1, v_n]\}),$$

unde $V_n = \{v_1, \dots, v_n\}$, cu $v_n \notin V_{n-1}$. Cum T' este conex și are $n-1-1 = n-2$ muchii, rezultă că și T este conex (nodul nou, v_n , este adiacent cu v_1) și are $n-2+1 = n-1$ muchii, deci și T este un arbore. Avem $d_T(v_1) = d_{T'}(v_1) + 1 = d'_1 + 1 = d_1$, $d_T(v_n) = 1 = d_n$ și $d_T(v_i) = d_{T'}(v_i) = d'_i = d_i$ pentru orice $i \in \{2, \dots, n-1\}$. Astfel $d_T(v_i) = d_i$, $\forall i \in \{1, \dots, n\}$.

Demonstrația prin inducție este încheiată. □

Observația 4.2.2. Demonstrația propoziției anterioare este constructivă, ea indicând un algoritm de generare a unui arbore cu gradele nodurilor date.

Exemplul 4.2.1. Numerele 1, 1, 1, 2, 4, 5, 5, 5 din Exemplul 3.3.7 nu pot fi gradele nodurilor unui arbore, deoarece suma $\sum_{i=1}^8 d_i = 24 \neq 2 \cdot 8 - 2$.

Exemplul 4.2.2. Numerele 1, 1, 1, 1, 1, 2, 3, 4 pot fi gradele nodurilor unui arbore, deoarece suma $\sum_{i=1}^8 d_i = 14 = 2 \cdot 8 - 2$. Mai mult, conform demonstrației propoziției anterioare (partea " \Leftarrow "), un arbore având gradele nodurilor numerele date se construiește adăugând succesiv câte o muchie între un nod de grad maxim și un nod de grad 1, grade care se micșorează cu 1, până când gradele devin toate egale cu zero. Acest procedeu este evidențiat în tabelul următor:

Pas	Gradele rămase	Muchia adăugată
1	1, 1, 1, 1, 1, 2, 3, 4	$[v_8, v_1]$
2	0, 1, 1, 1, 1, 2, 3, 3	$[v_8, v_2]$
3	0, 0, 1, 1, 1, 2, 3, 2	$[v_7, v_3]$
4	0, 0, 0, 1, 1, 2, 2, 2	$[v_8, v_4]$
5	0, 0, 0, 0, 1, 2, 2, 1	$[v_7, v_5]$
6	0, 0, 0, 0, 0, 2, 1, 1	$[v_6, v_7]$
7	0, 0, 0, 0, 0, 1, 0, 1	$[v_8, v_6]$
—	0, 0, 0, 0, 0, 0, 0, 0	—

Arborele obținut este reprezentat în Figura 4.2.1.

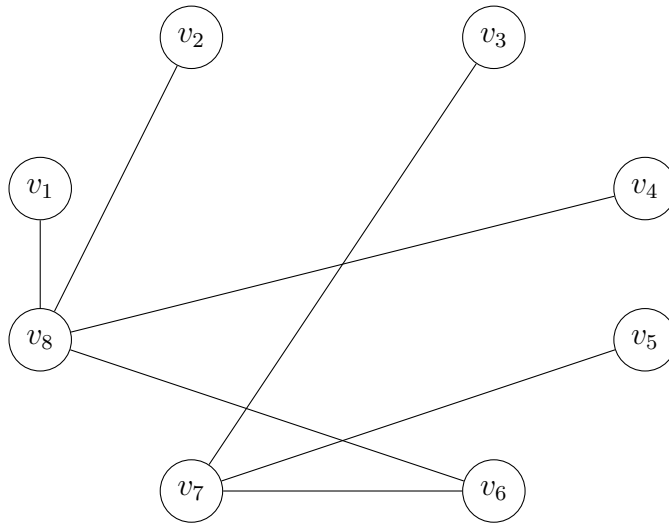


Figura 4.2.1:

4.3 Teorema de caracterizare a arborescențelor

Definiția 4.3.1. Fie $G = (V, E)$ un graf orientat. Un nod $x \in V$ se numește **rădăcină** a grafului G dacă pentru orice nod $y \in V \setminus \{x\}$ există cel puțin un drum de la x la y .

Exemplul 4.3.1. În graful din Exemplul 3.1.2 nodurile 1, 2 și 3 sunt rădăcini, iar nodurile 4 și 5 nu sunt rădăcini.

Definiția 4.3.2. a) O **arborescență** este un arbore care are o rădăcină.

b) O **arborescență parțială** a unui graf orientat $G = (V, E)$ este un graf parțial al lui G ce este arborescență (adică o arborescență $T = (V, F)$ cu $F \subseteq E$).

Exemplul 4.3.2. Graful din Exemplul 3.1.2 nu este arborescență (are cicluri, deci nu este arbore). Două arborescențe parțiale ale sale sunt reprezentate în Figurile 4.3.1 și 4.3.2.

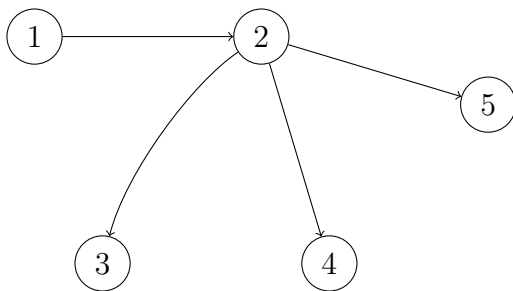


Figura 4.3.1:

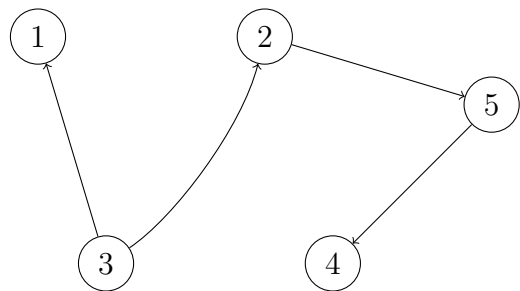


Figura 4.3.2:

Definiția 4.3.3. Un graf orientat $G = (V, E)$ se numește **quasi-tare conex** dacă pentru orice două noduri $x, y \in V$ există un nod $z \in V$ și drumuri de la z la x și de la z la y .

Observația 4.3.1. Ca și în definițiile conexității și tare-conexității, și în Definițiile 4.3.1 și 4.3.3 putem înlocui termenul de "drum" cu "drum elementar" (conform Observației 3.4.5).

Observația 4.3.2. Evident, orice graf quasi-tare conex este conex. Reciproca nu este adevărată. De exemplu, graful reprezentat în Figura 4.3.3 este conex, dar nu este quasi-tare conex (pentru nodurile $x = 1$ și $y = 4$ nu există noduri z din care să avem drumuri la x și la y).

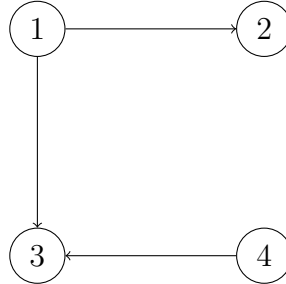


Figura 4.3.3:

Observația 4.3.3. Orice graf tare-conex este quasi-tare conex (deoarece pentru orice noduri x, y luând $z = x$ există drumuri (z, \dots, x) și (z, \dots, y)). Reciproca nu este adevărată. De exemplu, graful din Exemplul 3.1.2 este quasi-tare conex, dar nu este tare-conex.

Propoziția 4.3.1. *Un graf orientat este quasi-tare conex dacă și numai dacă are (cel puțin) o rădăcină.*

Demonstrație. " \Rightarrow " Fie $G = (V, E)$ un graf quasi-tare conex, $V = \{v_1, \dots, v_n\}$. Demonstrăm prin inducție după $k \in \{1, \dots, n\}$ că există un nod $x_k \in V$ cu drumuri $(x_k, \dots, v_1), (x_k, \dots, v_2), \dots, (x_k, \dots, v_k)$.

Pentru $k = 1$ luăm $x_1 = v_1$.

Presupunem adevărată afirmația pentru $k - 1$ și o demonstrăm pentru k . Deoarece graful este quasi-tare conex, rezultă că există un nod $x_k \in V$ cu drumuri (x_k, \dots, x_{k-1}) și (x_k, \dots, v_k) . Conform ipotezei de inducție, există drumuri $(x_{k-1}, \dots, v_1), \dots, (x_{k-1}, \dots, v_{k-1})$, deci există drumuri

$$(x_k, \dots, x_{k-1}, \dots, v_1), \dots, (x_k, \dots, x_{k-1}, \dots, v_{k-1}), (x_k, \dots, v_k).$$

Demonstrația prin inducție este încheiată.

Luând $k = n$, nodul x_n este o rădăcină a grafului G .

" \Leftarrow " Fie $G = (V, E)$ și $z \in V$ o rădăcină a lui G . Pentru orice noduri $x, y \in V$ există drumuri (z, \dots, x) și (z, \dots, y) , deci G este quasi-tare conex. \square

Teorema 4.3.1 (de caracterizare a arborescențelor). *Fie $G = (V, E)$ un graf orientat cu n noduri. Următoarele afirmații sunt echivalente:*

- 1) G este quasi-tare conex și fără cicluri;
- 2) G este quasi-tare conex și are $m = n - 1$ arce;
- 3) G este o arborescență (adică arbore cu rădăcină);
- 4) există un nod $x \in V$ astfel încât pentru orice nod $v \in V$ există un unic drum de la x la v ;
- 5) G este quasi-tare conex și minimal cu această proprietate, adică dacă se elimină un arc oarecare graful obținut nu mai este quasi-tare conex;
- 6) G este conex și există un nod $x \in V$ astfel încât $d^-(x) = 0$ și $d^-(v) = 1 \forall v \in V \setminus \{x\}$;

7) G este fără cicluri și există un nod $x \in V$ astfel încât $d^-(x) = 0$ și $d^-(v) = 1 \forall v \in V \setminus \{x\}$.

Demonstrație. 1) \Rightarrow 2) Fie G quasi-tare conex și fără cicluri. Fiind quasi-tare conex, G este conex. Deci G este un arbore și astfel are $m = n - 1$ arce (conform Teoremei de caracterizare a arborilor).

2) \Rightarrow 3) Fie G quasi-tare conex și cu $m = n - 1$ arce. Fiind quasi-tare conex, G este conex și are rădăcină (conform propoziției anterioare). Fiind conex cu $m = n - 1$ arce, G este arbore (conform teoremei amintite mai sus).

3) \Rightarrow 4) Fie G arborescență și x o rădăcină a lui G . Deci pentru orice nod $v \in V$ există un drum de la x la v . Graful G fiind un arbore, acest drum, fiind și lanț, este unic.

4) \Rightarrow 5) Fie G cu proprietatea 4). Deci x este o rădăcină a lui G și astfel G este quasi-tare conex (conform propoziției anterioare). Fie $e = (y, z) \in E$ un arc arbitrar fixat. Demonstrăm că $G - e = (V, E \setminus \{e\})$ nu este quasi-tare conex prin reducere la absurd. Dacă $G - e$ ar fi quasi-tare conex, atunci ar exista un nod $v \in V$ și drumuri (v, \dots, y) și (v, \dots, z) în $G - e$. x fiind rădăcină în G , s-ar obține două drumuri distincte $(x, \dots, v, \dots, y, z)$ și (x, \dots, v, \dots, z) de la x la z în G , contradicție cu ipoteza.

5) \Rightarrow 6) Fie G quasi-tare conex minimal. Fiind quasi-tare conex, G este conex. Conform propoziției anterioare, G are o rădăcină x . Demonstrăm că $d^-(x) = 0$ prin reducere la absurd. Dacă am avea $d^-(x) > 0$, atunci ar exista un arc $e = (y, x) \in E$, iar $G - e$ ar avea în continuare rădăcina x (deoarece există drum elementar (x, \dots, y) în G , deci și în $G - e$) și astfel $G - e$ ar rămâne quasi-tare conex (conform propoziției anterioare), contradicție cu ipoteza.

Fie acum $v \in V \setminus \{x\}$ un nod arbitrar fixat. x fiind rădăcină, există un drum (x, \dots, y, v) de lungime nenulă ($v \neq x$), deci $d^-(v) \geq 1$ ($\exists (y, v) \in E$). Demonstrăm că $d^-(v) = 1$ prin reducere la absurd. Dacă am avea $d^-(v) > 1$, atunci ar exista două arce diferite $e_1 = (y_1, v), e_2 = (y_2, v) \in E$, iar $G - e_1$ ar avea în continuare rădăcina x (deoarece există un drum (x, \dots, y_2, v) în $G - e_1$) și astfel $G - e_1$ ar rămâne quasi-tare conex (conform propoziției anterioare), contradicție cu ipoteza.

6) \Rightarrow 7) Fie G cu proprietatea 6). Conform Propoziției 3.3.2, $m = \sum_{t \in V} d^-(t) = n - 1$, deci G este un arbore (conform Teoremei de caracterizare a arborilor) și astfel nu are cicluri.

7) \Rightarrow 1) Fie G cu proprietatea 7). Din nou, $m = \sum_{t \in V} d^-(t) = n - 1$, deci G este un arbore și astfel este conex. Rezultă că pentru orice nod $y \in V$ există un lanț elementar unic $[x, v_1, \dots, v_k, y]$.

Deoarece $d^-(x) = 0$ nu putem avea $(v_1, x) \in E$, deci $(x, v_1) \in E$. Cum $(x, v_1) \in E$ și $d^-(v_1) = 1$, nu putem avea $(v_2, v_1) \in E$, deci $(v_1, v_2) \in E$.

Continuând în acest mod (inducție!) obținem $(v_2, v_3) \in E, \dots, (v_k, y) \in E$, deci lanțul $[x, v_1, \dots, v_k, y]$ este un drum de la x la y .

Rezultă că x este o rădăcină în G și astfel, conform propoziției anterioare, G este quasi-tare conex. \square

Corolarul 4.3.1. Orice arborescență $G = (V, E)$ are o unică rădăcină x și aceasta verifică proprietățile 4), 6) și 7) din teorema anterioară.

Demonstrație. Concluzia rezultă imediat din demonstrația teoremei anterioare. \square

4.4 Numărarea arborilor parțiali

Definiția 4.4.1. Fie $G = (V, E)$ un graf fără bucle, unde $V = \{v_1, \dots, v_n\}$. **Matricea de admitanță (matricea gradelor)** asociată grafului G este matricea $M = (m_{ij})_{i,j=1,n}$ definită prin:

$$m_{ii} = d_G(v_i), \quad \forall i \in \{1, \dots, n\};$$

$-m_{ij} = \text{numărul de muchii sau arce dintre nodurile } v_i \text{ și } v_j, \forall i, j \in \{1, \dots, n\}, i \neq j.$

Exemplul 4.4.1. Fie G graful reprezentat în Figura 4.4.1.

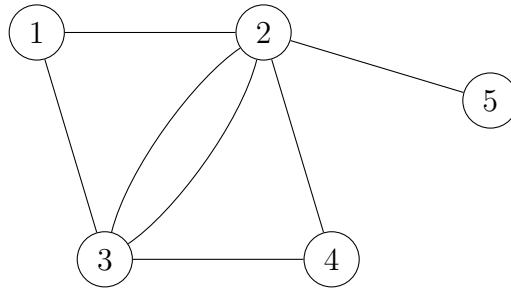


Figura 4.4.1:

Matricea de admitanță a acestui graf este

$$M = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 5 & -2 & -1 & -1 \\ -1 & -2 & 4 & -1 & 0 \\ 0 & -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{pmatrix}.$$

Observația 4.4.1. Matricea de admitanță este o matrice simetrică (și pentru grafuri orientate!) și are toate sumele pe linii și pe coloane egale cu zero.

Observația 4.4.2. Definiția 4.4.1 se poate extinde și pentru grafuri cu bucle, considerând drept matrice de admitanță a unui astfel de graf matricea de admitanță a grafului obținut prin eliminarea tuturor buclelor.

Teorema 4.4.1 (Kirchoff-Trent, de numărare a arborilor parțiali). Fie $G = (V, E)$ un graf fără bucle având matricea de admitanță $M = (m_{ij})_{i,j=\overline{1,n}}$, $n \geq 2$. Atunci numărul $t(G)$ de arbori parțiali ai grafului G verifică egalitatea

$$t(G) = (-1)^{i+j} \det[M]_{ij}, \quad \forall i, j \in \{1, \dots, n\},$$

unde $[M]_{ij}$ reprezintă matricea obținută din M prin eliminarea liniei i și coloanei j .

Observația 4.4.3. Teorema anterioară este valabilă și pentru grafuri cu bucle, folosind Observația 4.4.2.

Exemplul 4.4.2. Aplicând teorema anterioară rezultă că numărul arborilor parțiali ai grafului G din Exemplul 4.4.1 este

$$t(G) = \det[M]_{22} = \begin{vmatrix} 2 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} = 12.$$

Definiția 4.4.2. a) Graful neorientat simplu $K_n = (V, E)$ definit prin

$$V = \{v_1, \dots, v_n\}, \quad E = \{[v_i, v_j] | i, j \in \{1, \dots, n\}, i \neq j\}$$

se numește **graf complet** de ordinul n , unde $n \in \mathbb{N}^*$.

b) Graful neorientat simplu $K_{p,q} = (V, E)$ definit prin

$$V = \{x_1, \dots, x_p, y_1, \dots, y_q\}, E = \{[x_i, y_j] | i \in \{1, \dots, p\}, j \in \{1, \dots, q\}\}$$

se numește **graf bipartit complet**, unde $p, q \in \mathbb{N}^*$.

În particular, un graf bipartit complet de forma $K_{1,q}$ se numește **graf stea**.

Exemplul 4.4.3. Graful complet K_6 este reprezentat în Figura 4.4.2.

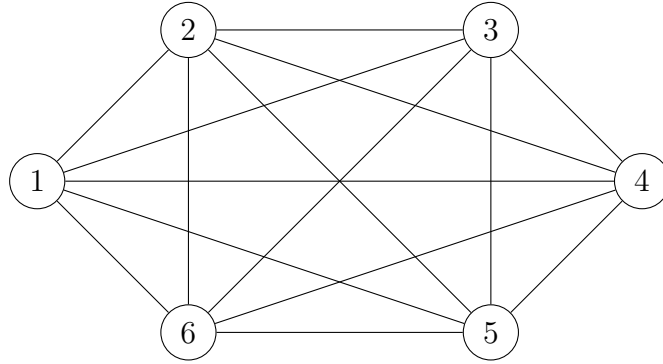


Figura 4.4.2:

Graful bipartit complet $K_{3,2}$ este reprezentat în Figura 4.4.3 ($\{x_1, x_2, x_3\} = \{1, 2, 3\}$ și $\{y_1, y_2\} = \{4, 5\}$), iar graful stea $K_{1,6}$ este reprezentat în Figura 4.4.4 ($x_1 = 1$ și $\{y_1, \dots, y_6\} = \{2, \dots, 7\}$).

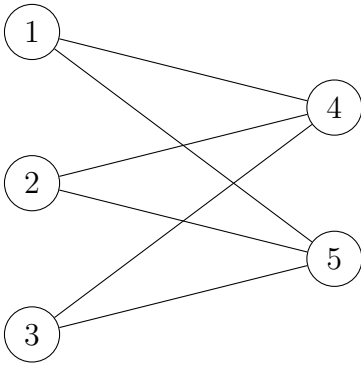


Figura 4.4.3:

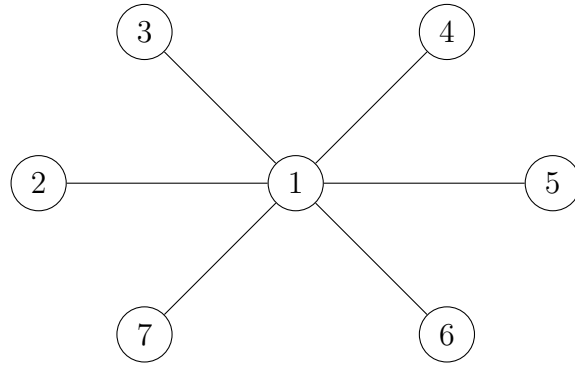


Figura 4.4.4:

Corolarul 4.4.1. Graful complet K_n are n^{n-2} arbori parțiali.

Demonstrație. Matricea de admitanță a grafului K_n este

$$M = \begin{pmatrix} n-1 & -1 & \dots & -1 \\ -1 & n-1 & \dots & -1 \\ \dots & \dots & \dots & \dots \\ -1 & -1 & \dots & n-1 \end{pmatrix} \text{ (de tipul } n \times n\text{)},$$

deci conform teoremei anterioare avem

$$t(K_n) = \det[M]_{11} = \begin{vmatrix} n-1 & -1 & \dots & -1 \\ -1 & n-1 & \dots & -1 \\ \dots & \dots & \dots & \dots \\ -1 & -1 & \dots & n-1 \end{vmatrix}$$

(determinant de ordinul $n - 1$). Adunând toate liniile la prima linie, apoi adunând linia obținută la celelalte linii obținem

$$t(K_n) = \begin{vmatrix} 1 & 1 & \dots & 1 \\ -1 & n-1 & \dots & -1 \\ \dots & \dots & \dots & \dots \\ -1 & -1 & \dots & n-1 \end{vmatrix} = \begin{vmatrix} 1 & 1 & \dots & 1 \\ 0 & n & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & n \end{vmatrix} = n^{n-2}.$$

□

Observația 4.4.4. Corolarul anterior poate fi reformulat astfel: numărul arborilor ce pot fi construiți cu n noduri date (numiți și **arbori etichetați**) este egal cu n^{n-2} .

Corolarul 4.4.2. Graful bipartit complet $K_{p,q}$ are $p^{q-1}q^{p-1}$ arbori parțiali.

4.5 Numărarea arborescențelor parțiale

Definiția 4.5.1. Fie $G = (V, E)$ un graf orientat fără bucle, unde $V = \{v_1, \dots, v_n\}$. **Matricea gradelor de intrare** asociată grafului G este matricea $\overline{M} = (\overline{m}_{ij})_{i,j=\overline{1},n}$ definită prin:

$$\overline{m}_{ii} = d_G^-(v_i), \quad \forall i \in \{1, \dots, n\};$$

$$-\overline{m}_{ij} = \text{numărul de arce de la nodul } v_i \text{ la nodul } v_j, \quad \forall i, j \in \{1, \dots, n\}, \quad i \neq j.$$

Exemplul 4.5.1. Pentru graful orientat G din Exemplul 3.1.2, matricea gradelor de intrare este

$$\overline{M} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 2 & -1 & -1 & -1 \\ -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}.$$

Observația 4.5.1. Matricea gradelor de intrare are toate sumele pe coloane egale cu zero.

Observația 4.5.2. Definiția 4.5.1 se poate extinde și pentru grafuri orientate cu bucle, considerând drept matricea gradelor de intrare a unui astfel de graf matricea gradelor de intrare a grafului obținut prin eliminarea tuturor buclelor.

Teorema 4.5.1 (de numărare a arborescențelor parțiale). Fie $G = (V, E)$ un graf orientat fără bucle cu $V = \{v_1, \dots, v_n\}$, $n \geq 2$, și fie $\overline{M} = (\overline{m}_{ij})_{i,j=\overline{1},n}$ matricea gradelor de intrare asociată grafului G .

- a) Pentru orice nod $v_i \in V$, numărul de arborescențe parțiale ale grafului G ce au ca rădăcină nodul v_i este egal cu

$$\det[\overline{M}]_{ii}$$

(unde $[\overline{M}]_{ii}$ reprezintă matricea obținută din \overline{M} prin eliminarea liniei i și coloanei i).

- b) Numărul total de arborescențe parțiale ale grafului G este egal cu

$$\sum_{i=1}^n \det[\overline{M}]_{ii}.$$

Observația 4.5.3. Teorema anterioară este valabilă și pentru grafuri cu bucle, folosind Observația 4.5.2.

Exemplul 4.5.2. Aplicând teorema anterioară pentru graful G din Exemplul 3.1.2 rezultă că numărul arborescențelor parțiale având rădăcina 1 este egal cu

$$\det[\overline{M}]_{11} = \begin{vmatrix} 2 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & -1 & 1 \end{vmatrix} = 2.$$

Numărul total de arborescențe parțiale ale grafului G este egal cu

$$\begin{aligned} & \det[\overline{M}]_{11} + \det[\overline{M}]_{22} + \det[\overline{M}]_{33} + \det[\overline{M}]_{44} + \det[\overline{M}]_{55} \\ &= 2 + 2 + 4 + 0 + 0 = 8. \end{aligned}$$

Tema 5

Arbori parțiali de cost minim

Problema determinării arborilor parțiali de cost minim are numeroase aplicații practice. În continuare vom prezenta doi algoritmi fundamentali pentru rezolvarea acestei probleme.

5.1 Expunerea problemei

Definiția 5.1.1. Un **graf ponderat** este o pereche (G, c) , unde $G = (V, E)$ este un graf iar $c : E \rightarrow \mathbb{R}$ este o funcție numită **pondere (cost)**. Pentru orice $e \in E$, $c(e)$ se numește **ponderea (costul)** muchiei sau arcului e .

Definiția 5.1.2. Fie (G, c) un graf ponderat, $G = (V, E)$.

a) Dacă $H = (U, F)$ este un subgraf al lui G , atunci **costul (ponderea)** lui H este

$$c(H) = \sum_{e \in F} c(e)$$

(adică suma costurilor muchiilor sau arcelor sale).

b) Un arbore parțial $T^* = (V, F)$ al lui G cu proprietatea că

$$c(T^*) = \min\{c(T) \mid T = \text{arbore parțial al lui } G\}$$

se numește **arbore parțial de cost minim (APM)** al grafului ponderat (G, c) .

Observația 5.1.1. Conform Propoziției 4.1.7, un graf ponderat are arbori parțiali de cost minim dacă și numai dacă este conex.

Observația 5.1.2. Pentru un graf ponderat conex dat, arborele parțial de cost minim nu este neapărat unic. De exemplu, graful ponderat din Figura 5.1.1 are 4 arbori parțiali de cost minim, reprezentați în Figurile 5.1.2–5.1.5.

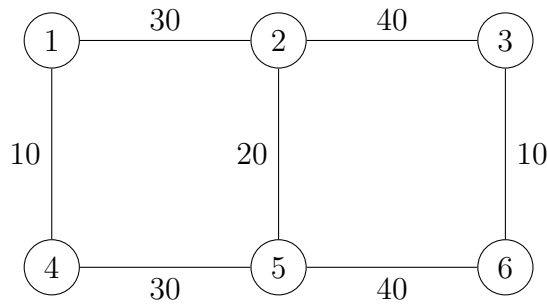


Figura 5.1.1:

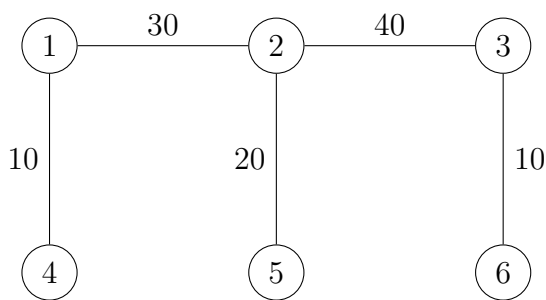


Figura 5.1.2:

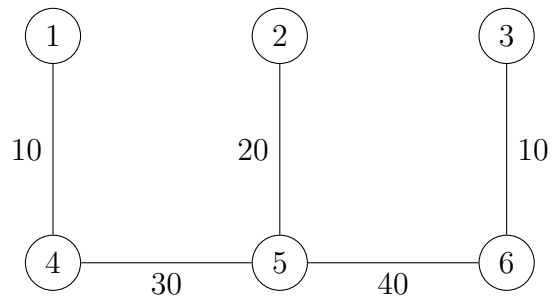


Figura 5.1.3:

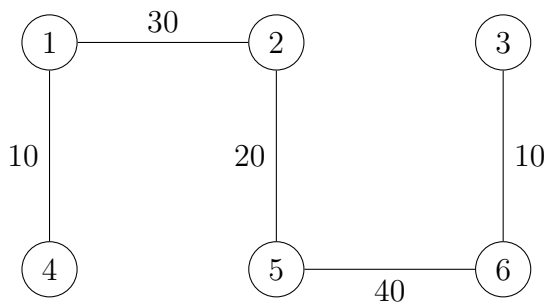


Figura 5.1.4:

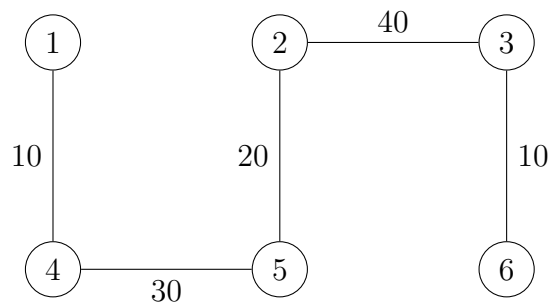


Figura 5.1.5:

În cazul în care costurile muchiilor sunt diferite două câte două avem următorul rezultat.

Propoziția 5.1.1. *Un graf ponderat conex având costurile muchiilor diferite două câte două are un unic arbore parțial de cost minim.*

Demonstrație. Fie (G, c) un graf ponderat conex cu $G = (V, E)$, astfel încât $c(e_1) \neq c(e_2)$ pentru orice $e_1 \neq e_2$, $e_1, e_2 \in E$. Demonstrăm că acest graf are un unic APM prin reducere la absurd.

Presupunem că există doi arbori parțiali de cost minim $T_1 = (V, F_1)$ și $T_2 = (V, F_2)$. Fie $f = [x, y] \in E$ astfel încât

$$c(f) = \min\{c(e) \mid e \in F_1 \Delta F_2\}, \quad (5.1.1)$$

unde $F_1 \Delta F_2 = (F_1 \setminus F_2) \cup (F_2 \setminus F_1)$ este diferența simetrică a mulțimilor F_1 și F_2 . Analizăm cazul $f \in F_1 \setminus F_2$ (cazul $f \in F_2 \setminus F_1$ fiind analog).

Conform Teoremei numărului ciclomatic, graful $H = T_2 + f = (V, F_2 \cup \{f\})$ obținut din arborele parțial T_2 prin adăugarea muchiei f conține un ciclu elementar unic C și $f \in E(C)$. Arborele T_1 nu

conține cicluri, deci există o muchie $f' \in E(C)$ astfel încât $f' \notin F_1$. Cum $f \in F_1$ rezultă că $f' \neq f$, deci $f' \in E(C) \setminus \{f\}$. Cum $E(C) \setminus \{f\} \subseteq F_2$, deducem că $f' \in F_2 \setminus F_1$, deci $f' \in F_1 \Delta F_2$. Atunci, din alegerea muchiei f conform proprietății (5.1.1) și din ipoteza că muchiile grafului G au costurile diferite, avem

$$c(f) < c(f').$$

Fie

$$T = (T_2 + f) - f'$$

graful parțial al lui G obținut din T_2 prin adăugarea muchiei f și eliminarea muchiei f' , adică $T = (V, F)$, unde $F = (F_2 \cup \{f\}) \setminus \{f'\}$.

Evident, T este conex (între nodurile x și y există lanțul obținut din ciclul C prin eliminarea muchiei f) și are $n-1$ muchii (deoarece T_2 are $n-1$ muchii). Conform Teoremei numărului ciclomatic rezultă că T este un arbore parțial al grafului G .

Deoarece $c(f) < c(f')$ obținem

$$c(T) = c(T_2) + c(f) - c(f') < c(T_2),$$

ceea ce contrazice faptul că arborele parțial T_2 are costul minim. □

Observația 5.1.3. Reciproca propoziției anterioare nu este adevărată. De exemplu, graful ponderat din Figura 5.1.6 are un unic arbore parțial de cost minim, reprezentat în Figura 5.1.7, deși nu are costurile muchiilor diferite ($c([1, 2]) = c([2, 3]) = c([2, 5])$ și $c([1, 4]) = c([3, 6])$).

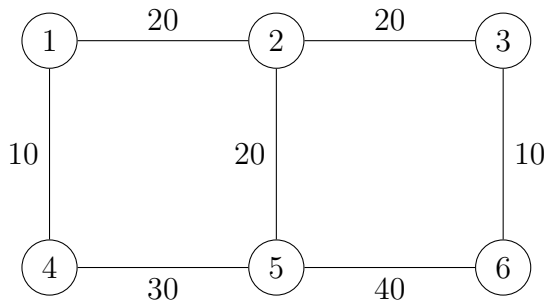


Figura 5.1.6:

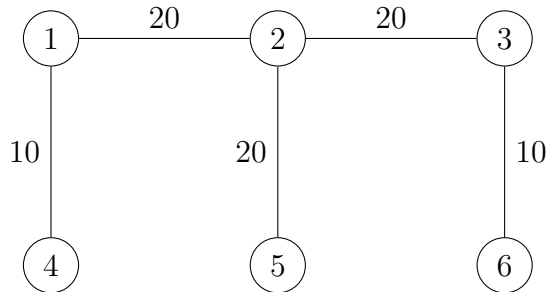


Figura 5.1.7:

5.2 Algoritmii lui Kruskal și Prim

Algoritmul 5.2.1 (Kruskal). Fie (G, c) un graf ponderat conex cu $G = (V, E)$, $V = \{v_1, \dots, v_n\}$. Algoritmul are $n-1$ pași.

- La pasul i , $i = \overline{1, n-1}$, dintre muchiile neselectate la pașii anteriori se selectează o muchie $e_i \in E$ de cost minim cu proprietatea că nu formează cicluri cu muchiile $\{e_1, \dots, e_{i-1}\}$ selectate la pașii anteriori.

Algoritmul 5.2.2 (Prim). Fie (G, c) un graf ponderat conex cu $G = (V, E)$, $V = \{v_1, \dots, v_n\}$. Algoritmul are n pași.

- La pasul 0 se selectează un nod arbitrar $x_0 \in V$.

- La pasul i , $i = \overline{1, n-1}$, se selectează o muchie $e_i = [x_j, x_i] \in E$ de cost minim cu proprietatea că are ca extremități un nod $x_j \in V$ selectat la un pas anterior și celălalt nod $x_i \in V$ neselectat la pașii anteriori; se selectează și nodul x_i .

Teorema 5.2.1 (de corectitudine a algoritmilor Kruskal și Prim). *În contextul Algoritmilor Kruskal sau Prim, fie $F = \{e_1, \dots, e_{n-1}\}$ mulțimea muchiilor selectate. Atunci $T = (V, F)$ este un arbore parțial de cost minim al grafului ponderat (G, c) .*

Demonstrație. Vom prezenta o demonstrație comună a corectitudinii celor doi algoritmi. Fie $T_0 = (V, \emptyset)$ și $T_i = (V, \{e_1, \dots, e_i\})$, $\forall i \in \{1, 2, \dots, n-1\}$, unde e_1, e_2, \dots, e_{n-1} sunt muchiile selectate, în această ordine, de Algoritmul Kruskal sau de Algoritmul Prim.

Graful G fiind conex, selectarea muchiei e_i este posibilă la fiecare pas i , iar T_i este o pădure parțială a lui G (afirmație evidentă pentru Algoritmul Kruskal, iar pentru Algoritmul Prim este o consecință a faptului că nodurile neselectate la pasul i sunt izolate în T_i).

Demonstrăm prin inducție după $i \in \{0, 1, \dots, n-1\}$ că există un APM $T^* = (V, F^*)$ astfel încât

$$T_i \subseteq T^* \text{ (adică } \{e_1, \dots, e_i\} \subseteq F^*). \quad (5.2.1)$$

Pentru $i = 0$ afirmația este evidentă, luând T^* orice APM (există, deoarece G este conex).

Presupunem (5.2.1) adevărată pentru $i-1$, adică există $T^* = (V, F^*)$ APM cu $T_{i-1} \subseteq T^*$ și o demonstrăm pentru i ($i \in \{1, 2, \dots, n-1\}$).

Fie $e_i = [x_j, x_i]$ muchia selectată de Algoritmul Kruskal sau de Algoritmul Prim la pasul i . În cazul Algoritmului Prim, x_j este nodul selectat la un pas anterior ($j \leq i-1$). Avem două cazuri.

Cazul 1) $e_i \in F^*$. Atunci $\{e_1, \dots, e_i\} \subseteq F^*$, deci $T_i \subseteq T^*$.

Cazul 2) $e_i \notin F^*$. Conform Teoremei numărului ciclomatic, graful $H = T^* + e_i = (V, F^* \cup \{e_i\})$ obținut din T^* prin adăugarea muchiei e_i conține un ciclu elementar unic

$$C_i = [x_j, w_0, \dots, w_k, x_i, x_j]$$

având ultima muchie chiar muchia e_i . Cum $T_i = T_{i-1} + e_i$ este o pădure, rezultă că x_j și x_i sunt în componente conexe diferite ale lui T_{i-1} (în caz contrar T_i ar conține un ciclu de forma $[x_j, y_0, \dots, y_r, x_i, x_j]$ având ultima muchie e_i). Rezultă că lanțul elementar

$$\mu_i = [x_j, w_0, \dots, w_k, x_i]$$

(obținut din ciclul C_i prin eliminarea muchiei e_i) conține o muchie $e'_i = [x', y']$ astfel încât x' este în aceeași componentă conexă cu x_j în pădurea T_{i-1} , iar y' nu este în această componentă conexă. Evident, rezultă că e'_i nu este muchie a lui T_{i-1} , adică $e'_i \notin \{e_1, \dots, e_{i-1}\}$, iar graful $T'_i = T_{i-1} + e'_i$ nu conține cicluri. De asemenea, $e'_i \neq e_i$, deci $e'_i \in F^*$.

Pe de o parte, din descrierea Algoritmului Kruskal deducem că

$$c(e_i) \leq c(e'_i)$$

(deoarece muchia e'_i nu formează cicluri cu $\{e_1, \dots, e_{i-1}\}$, deci e_i fiind muchia selectată la pasul i verifică această inegalitate).

Pe de altă parte, din descrierea Algoritmului Prim deducem că la pasul i nodul x' era deja selectat la un pas anterior (fiind în aceeași componentă conexă cu x_j în T_{i-1}), iar nodul y' nu era selectat la un pas anterior (nefiind în acea componentă conexă), deci din nou avem

$$c(e_i) \leq c(e'_i)$$

(e_i fiind muchia selectată la pasul i verifică această inegalitate).

Continuăm demonstrația comună pentru cei doi algoritmi. Fie

$$T^{**} = (T^* + e_i) - e'_i$$

graful parțial al lui G obținut din T^* prin adăugarea muchiei e_i și eliminarea muchiei e'_i , adică $T^{**} = (V, F^{**})$, unde $F^{**} = F^* \cup \{e_i\} \setminus \{e'_i\}$.

Evident, T^{**} este conex (T^* este conex iar între nodurile x' și y' după eliminarea muchiei e'_i rămâne lanțul obținut din ciclul C_i prin eliminarea acestei muchii) și are $n - 1$ muchii (deoarece T^* are $n - 1$ muchii). Conform Teoremei numărului ciclomatic rezultă că T^{**} este un arbore parțial al grafului G .

Deoarece $c(e_i) \leq c(e'_i)$ obținem

$$c(T^{**}) \leq c(T^*)$$

și cum T^* este un APM rezultă că și T^{**} este un APM (și, în plus, $c(T^{**}) = c(T^*)$). Cum $T_i \subseteq T^{**}$, demonstrația prin inducție a relației (5.2.1) este completă.

Luând $i = n - 1$ în această relație obținem că $T \subseteq T^*$, unde $T = T_{n-1} = (V, F)$, $F = \{e_1, \dots, e_{n-1}\}$ (mulțimea muchiilor selectate de algoritm) iar T^* este un APM. Dar T și T^* au fiecare câte $n - 1$ muchii, deci $T = T^*$ și astfel T este un APM al grafului dat. \square

Exemplul 5.2.1. Fie graful ponderat (G, c) reprezentat în Figura 5.2.1, unde costul fiecărei muchii este scris lângă segmentul corespunzător acesteia.

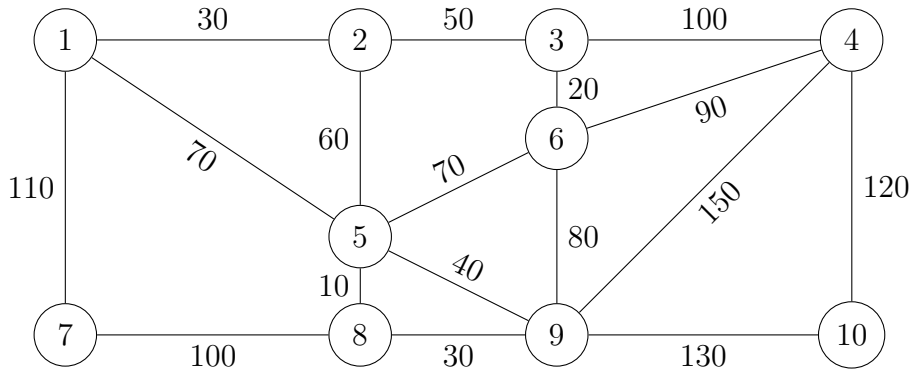


Figura 5.2.1:

Aplicarea Algoritmului Kruskal este evidențiată în următorul tabel:

Pas	Muchia selectată	Costul ei
1	[5, 8]	10
2	[3, 6]	20
3	[1, 2]	30
4	[8, 9]	30
5	[2, 3]	50
6	[2, 5]	60
7	[4, 6]	90
8	[7, 8]	100
9	[4, 10]	120

Arborele parțial de cost minim obținut este reprezentat în Figura 5.2.2. Costul acestui APM este de 510.

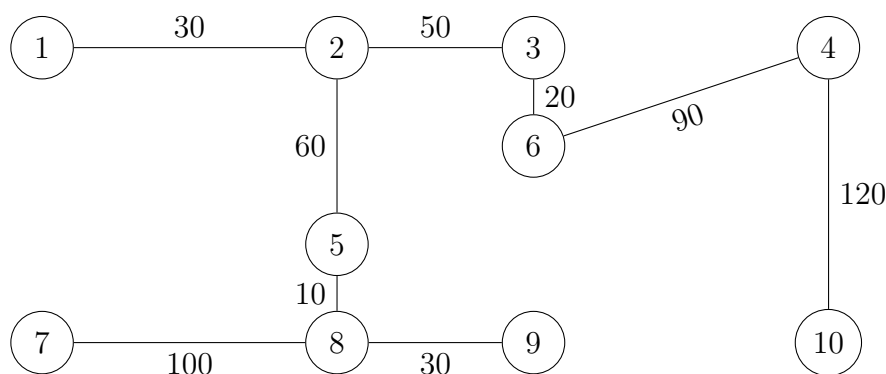


Figura 5.2.2:

Aplicarea Algoritmului Prim pentru același graf este evidențiată în următorul tabel:

Pas	Muchia selectată	Costul ei	Nodul selectat
0	-	-	1
1	[1, 2]	30	2
2	[2, 3]	50	3
3	[3, 6]	20	6
4	[2, 5]	60	5
5	[5, 8]	10	8
6	[8, 9]	30	9
7	[6, 4]	90	4
8	[8, 7]	100	7
9	[4, 10]	120	10

Arborele parțial de cost minim obținut este deci același cu cel obținut prin aplicarea Algoritmului Kruskal.

Observația 5.2.1. Algoritmii Kruskal și Prim sunt specifici **metodei de programare Greedy**. Algoritmul Kruskal selectează muchii, în ordinea crescătoare a costurilor, subgrafurile induse pe parcurs de acestea nefiind neapărat conexe. Algoritmul Prim selectează muchii și noduri, nu neapărat în ordinea crescătoare a costurilor muchiilor, iar subgrafurile induse pe parcurs de muchiile selectate sunt conexe.

În implementări optime, se poate arăta că Algoritmul Kruskal are complexitatea $\mathcal{O}(m \ln n)$ (fiind necesară sortarea muchiilor după cost), iar Algoritmul Prim are complexitatea $\mathcal{O}(n^2)$ în cazul memorării grafului prin matricea de adiacență (o astfel de implementare va fi prezentată în continuare), unde n și m reprezintă numerele de noduri, respectiv de muchii ale grafului dat. Graful fiind conex, $m \geq n - 1$.

Pentru grafuri simple, $m \leq \frac{n(n-1)}{2}$. Folosind și inegalitatea $\ln n \leq n - 1$, obținem că Algoritmul Kruskal este mai eficient pentru grafuri "sărace" în muchii, iar Algoritmul Prim este mai eficient pentru grafuri "bogate" în muchii.

Observația 5.2.2. Pentru implementarea Algoritmului Kruskal, memorăm graful ponderat conex (G, c) , unde $G = (V, E)$, $V = \{1, \dots, n\}$, $E = \{e_1, \dots, e_m\}$, într-o matrice cu 3 linii și m coloane

$P = (p_{ik})_{\substack{i = \overline{1,3} \\ k = \overline{1,m}}}$ având semnificația:

dacă $e_k = [x_k, y_k] \in E$, atunci $p_{1k} = x_k$, $p_{2k} = y_k$ și $p_{3k} = c(e_k)$.

Utilizăm un vector S cu semnificația

$$S[k] = \begin{cases} 1, & \text{dacă } e_k \text{ a fost selectată,} \\ 0, & \text{în caz contrar,} \end{cases}$$

$\forall k \in \{1, \dots, m\}$ și un vector CC cu semnificația

$CC[i] =$ numărul componente conexe în care se află nodul i în graful indus de muchiile selectate,
 $\forall i \in \{1, \dots, n\}$.

Astfel o muchie $[x, y]$ nu formează cicluri cu muchiile selectate dacă și numai dacă

$$CC[x] \neq CC[y].$$

Descrierea în pseudocod a algoritmului are următoarea formă.

KRUSKAL:

```

SORTARE( $P$ );                                // se sortează coloanele matricei  $P$ 
                                              // crescător după costurile muchiilor

for  $i = \overline{1, m}$  do  $S[i] \leftarrow 0$ ;
for  $i = \overline{1, n}$  do  $CC[i] \leftarrow i$ ;
 $cost \leftarrow 0$ ;                                // costul APM
 $poz \leftarrow 0$ ;                                // căutarea următoarei muchii  $e_k$  ce va fi
                                              // selectată începe de pe poziția  $poz + 1$ 

for  $l = \overline{1, n-1}$  do                                // pasul  $l$ 
     $k \leftarrow poz$ ;
    repeat
         $k \leftarrow k + 1$ ;  $x \leftarrow p_{1k}$ ;  $y \leftarrow p_{2k}$ ;  $c \leftarrow p_{3k}$ ;
    while ( $CC[x] = CC[y]$ );
     $S[k] \leftarrow 1$ ;                                // selectăm  $e_k = [x, y]$ 
     $cost \leftarrow cost + c$ ;  $poz \leftarrow k$ ;
     $aux \leftarrow CC[y]$ ;                                // actualizăm vectorul  $CC$  prin unificarea
                                              // componentelor conexe ale lui  $x$  și  $y$ 

    for  $i = \overline{1, n}$  do
        if ( $CC[i] = aux$ ) then  $CC[i] \leftarrow CC[x]$ ;

```

Observația 5.2.3. Pentru implementarea Algoritmului Prim, memorăm graful ponderat conex și simplu (G, c) , unde $G = (V, E)$, $V = \{1, \dots, n\}$, $E = \{e_1, \dots, e_m\}$, cu ajutorul unei matrice $C = (c_{ij})_{i,j=\overline{1,n}}$ a **costurilor (directe)** având semnificația

$$c_{ij} = \begin{cases} c([i, j]), & \text{dacă } [i, j] \in E, \\ 0, & \text{dacă } i = j, \\ \infty, & \text{în rest,} \end{cases} \quad (5.2.2)$$

$\forall i, j \in \{1, \dots, n\}$. Pentru grafuri neorientate, matricea C este simetrică. În cazul grafurilor nesimple putem lua

$$c_{ij} = \min\{c(e) | e = [i, j] \in E\}.$$

Utilizăm un vector S cu semnificația

$$S[i] = \begin{cases} 1, & \text{dacă nodul } i \text{ a fost selectat,} \\ 0, & \text{în caz contrar} \end{cases}$$

și doi vectori t și $TATA$ având semnificația

$$\begin{aligned} t[i] &= \text{costul minim al unei muchii } [i, j] \text{ de la nodul } i \text{ la un nod selectat } j, \\ TATA[i] &= \text{nodul } j \text{ ce atinge minimul în } t[i], \forall i \in \{1, \dots, n\}. \end{aligned}$$

Descrierea în pseudocod a algoritmului are următoarea formă.

PRIM:

```

 $S[1] \leftarrow 1;$  // selectăm nodul 1
 $cost \leftarrow 0;$  // costul APM
for  $i = \overline{2, n}$  do // inițializări
     $S[i] \leftarrow 0; t[i] \leftarrow c_{i1}; TATA[i] \leftarrow 1;$ 
for  $l = \overline{1, n-1}$  do // căutăm nodul  $y$  și muchia  $[x, y]$ 
    // ce vor fi selectate la pasul  $l$ 
     $min \leftarrow \infty;$ 
    for  $i = \overline{2, n}$  do
        if  $(S[i] = 0) \text{ and } (t[i] < min)$  then
             $min \leftarrow t[i]; y \leftarrow i;$ 
     $S[y] \leftarrow 1;$  // selectăm nodul  $y$ 
     $x \leftarrow TATA[y];$  // și muchia  $[x, y]$ 
     $cost \leftarrow cost + c_{xy};$ 
    for  $i = \overline{2, n}$  do // actualizăm vectorii  $t$  și  $TATA$ 
        if  $(S[i] = 0) \text{ and } (t[i] > c_{iy})$  then
             $t[i] \leftarrow c_{iy}; TATA[i] \leftarrow y;$ 

```

Tema 6

Clase particulare de grafuri

6.1 Grafuri euleriene

Definiția 6.1.1. a) Fie $G = (V, E)$ un graf. Un lanț simplu, ciclu, drum simplu sau circuit în graful G ce conține toate muchiile sau arcele lui G se numește **eulerian**.

b) Un graf neorientat se numește **eulerian** dacă are (cel puțin) un ciclu eulerian.

c) Un graf orientat se numește **eulerian** dacă are (cel puțin) un circuit eulerian.

Exemplul 6.1.1. Graful neorientat reprezentat în Figura 6.1.1 este eulerian, un ciclu eulerian al său fiind $C = [1, 2, 3, 4, 10, 9, 3, 6, 2, 5, 8, 7, 5, 1]$.

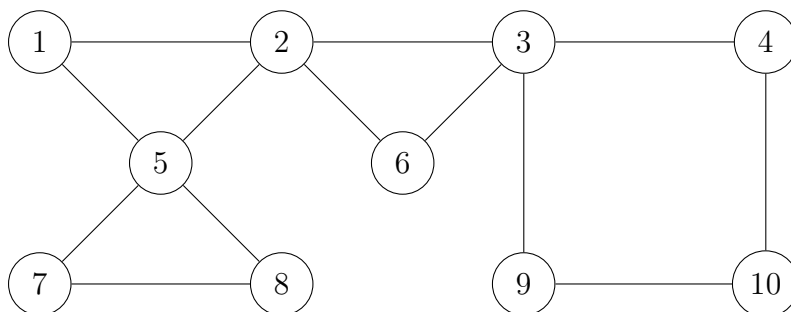


Figura 6.1.1:

Exemplul 6.1.2. Fie graful orientat reprezentat în Figura 6.1.2. Un drum eulerian în acest graf este $(1, 2, 3, 4, 6, 3, 5, 6, 1, 5, 4)$.

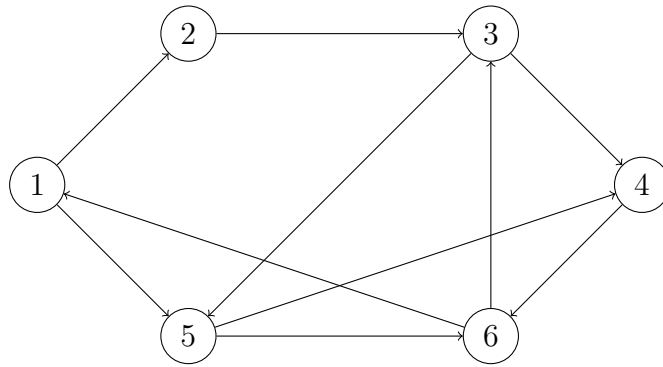


Figura 6.1.2:

Observația 6.1.1. Existența lanțurilor, ciclurilor, drumurilor sau circuitelor euleriene nu este influențată de prezența unor eventuale noduri izolate, deci putem analiza în continuare doar grafurile fără noduri izolate.

Teorema 6.1.1 (Euler, de caracterizare a grafurilor euleriene neorientate). *Un graf neorientat fără noduri izolate este eulerian dacă și numai dacă este conex și par.*

Demonstrație. " \Rightarrow " Fie $G = (V, E)$ un graf neorientat eulerian, fără noduri izolate. Fie C un ciclu eulerian al lui G . Neexistând noduri izolate, orice nod $x \in V$ are o muchie incidentă, și cum această muchie aparține lui C rezultă că și nodul x aparține lui C . Deci C conține și toate nodurile lui G . Rezultă că G este conex, între orice două noduri distincte $x, y \in V$ existând un lanț elementar conținut în ciclul C .

Cum C conține toate muchiile lui G rezultă că

$$d_G(x) = d_C(x) = \text{par}, \quad \forall x \in V$$

(orice ciclu este evident un subgraf par).

" \Leftarrow " Fie $G = (V, E)$ un graf neorientat conex, par și fără noduri izolate. Conform Propoziției 4.1.2, graful G are o descompunere în cicluri elementare muchie-disjuncte C_1, \dots, C_k , $k \geq 1$.

Dacă $k = 1$, atunci C_1 este un ciclu eulerian în G .

Dacă $k \geq 2$, graful G fiind conex, ciclul C_1 are un nod comun cu un alt ciclu din $\{C_2, \dots, C_k\}$. Fie C_2 acest ciclu și C'_2 ciclul obținut pornind din nodul comun și parcurgând întâi C_1 apoi C_2 . Spunem că C'_2 se obține prin **concatenarea (alipirea)** ciclurilor C_1 și C_2 .

Dacă $k = 2$, atunci C'_2 este un ciclu eulerian în G .

Dacă $k \geq 3$, din conexitatea lui G obținem că C'_2 are un nod comun cu un alt ciclu din $\{C_3, \dots, C_k\}$, fie acesta C_3 , și fie C'_3 ciclul obținut prin concatenarea ciclurilor C'_2 și C_3 . Continuând acest procedeu de concatenare a ciclurilor, rezultă (prin inducție după k) că obținem un ciclu C'_k ce este eulerian în G . \square

Exemplul 6.1.3. Graful din Exemplul 6.1.1 este conex, par și fără noduri izolate, deci este eulerian.

Observația 6.1.2. Demonstrația teoremei anterioare este constructivă, ea indicând un algoritm de determinare a unui ciclu eulerian într-un graf neorientat conex, par și fără noduri izolate.

Prezentăm în continuare un algoritm pentru determinarea unui ciclu eulerian bazat pe arborele DF al grafului dat.

Algoritmul 6.1.1 (de determinare a unui ciclu eulerian). Fie $G = (V, E)$ un graf neorientat conex, par și fără noduri izolate. Verificarea conexității, cu neglijarea nodurilor izolate, poate fi efectuată cu ajutorul parcurgerii DF (Algoritmul 3.5.3), iar verificarea parității poate fi efectuată cu ajutorul Propoziției 3.3.1. Fie $T = (V, F)$ arborele DF al lui G , considerând ca rădăcină a parcurgerii DF un nod $x_1 \in V$ arbitrar fixat. Un ciclu eulerian în G poate fi obținut astfel:

- se pornește din nodul rădăcină x_1 ;
- se parcurg cu prioritate muchiile (neparcurse anterior) ce nu aparțin arborelui DF (adică dacă există o astfel de muchie $[x, y]$, incidentă cu nodul curent x , se parcurge această muchie și nodul curent devine y , iar dacă nu există o astfel de muchie se parcurge, dacă există, o muchie $[x, z]$ a arborelui DF , incidentă cu nodul curent x , și nodul curent devine z).
- se continuă parcurgerea în modul descris mai sus, cât timp este posibil.

Exemplul 6.1.4. Arborele $DF(1)$ al grafului din Exemplul 6.1.1 este reprezentat în Figura 6.1.3. Aplicând algoritmul anterior obținem ciclul eulerian

$[1, 5, 8, 7, 5, 2, 6, 3, 9, 10, 4, 3, 2, 1]$.

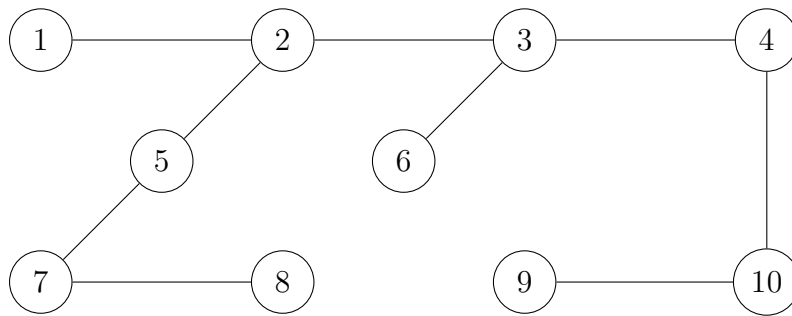


Figura 6.1.3:

Observația 6.1.3. Datorită necesității parcurgerii DF , dar și a tuturor muchiilor grafului dat, complexitatea Algoritmului 6.1.1 este $O(m)$, unde m reprezintă numărul de muchii ale grafului dat.

Observația 6.1.4. Pentru implementarea Algoritmului 6.1.1, în cazul grafurilor simple putem memora arborele DF tot în matricea de adiacență A considerând

$$a_{ij} = 2 \text{ dacă muchia } [i, j] \text{ aparține arborelui } DF.$$

Algoritmul descris în pseudocod are forma

CICLU_EULERIAN(x) :

```

 $i \leftarrow x;$                                 //  $i$  este nodul curent al ciclului eulerian
repeat
    VIZITEAZĂ( $i$ );                            // se vizitează nodul  $i$ ,
                                                // de exemplu se afișează  $i$ 
     $j \leftarrow 1;$ 
    while ( $a[i, j] \neq 1$  and  $j \leq n$ ) do        // căutăm  $[i, j] \notin DF$ 
         $j \leftarrow j + 1;$ 
    if ( $j \leq n$ ) then                            // există, deci  $[i, j] \in$  ciclu eulerian
         $a[i, j] \leftarrow 0; a[j, i] \leftarrow 0;$     // eliminăm  $[i, j]$  din graf
         $i \leftarrow j;$ 
    else
         $j \leftarrow 1;$ 
        while ( $a[i, j] \neq 2$  and  $j \leq n$ ) do        // căutăm  $[i, j] \in DF$ 
             $j \leftarrow j + 1;$ 
        if ( $j \leq n$ ) then                            // există, deci  $[i, j] \in$  ciclu eulerian
             $a[i, j] \leftarrow 0; a[j, i] \leftarrow 0;$     // eliminăm  $[i, j]$  din graf
             $i \leftarrow j;$ 
while ( $j \leq n$ );

```

Observația 6.1.5. În cazul grafurilor nesimple, pentru implementarea Algoritmului 6.1.1 graful dat poate fi memorat cu ajutorul matricei de adiacență, iar arborele DF poate fi memorat cu ajutorul vectorului $TATA$. Verificarea dacă $[i, j] \in DF$ revine la " $TATA[i] = j$ sau $TATA[j] = i$ ", iar eliminarea unei muchii $[i, j]$ din graf revine la " $a_{ij} \leftarrow a_{ij} - 1$ și $a_{ji} \leftarrow a_{ji} - 1$ ".

Corolarul 6.1.1. *Un graf neorientat fără noduri izolate are un lanț eulerian deschis dacă și numai dacă este conex și are exact două noduri de grade impare. În plus, orice lanț eulerian are ca extremități cele două noduri de grade impare.*

Demonstrație. " \Rightarrow " Fie $G = (V, E)$ un graf neorientat fără noduri izolate, având un lanț eulerian deschis $\mu = [x, v_1, \dots, v_k, y]$. Evident, G este conex.

Fie $G' = (V, E \cup \{e\})$, unde $e = [y, x]$, $e \notin E$ (muchie nouă). Adăugând muchia e la lanțul μ obținem un ciclu eulerian $C = [x, v_1, \dots, v_k, y, x]$ în graful G' . Conform Teoremei 6.1.1, G' este conex și par. Rezultă că

$$\begin{aligned}
 d_G(x) &= d_{G'}(x) - 1 = \text{impar}, \\
 d_G(y) &= d_{G'}(y) - 1 = \text{impar}, \\
 d_G(z) &= d_{G'}(z) = \text{par } \forall z \in V \setminus \{x, y\}.
 \end{aligned}$$

" \Leftarrow " Fie $G = (V, E)$ un graf neorientat conex, fără noduri izolate, cu exact două noduri x și y de grade impare. Fie G' graful definit ca mai sus. Rezultă că G' este conex și par, deci conform Teoremei 6.1.1 el conține un ciclu eulerian $C = [x, v_1, \dots, v_k, y, x]$. Eliminând muchia $e = [y, x]$ de pe ciclul C obținem un lanț eulerian deschis $\mu = [x, v_1, \dots, v_k, y]$ în graful G . \square

Observația 6.1.6. Folosind trecerea între grafurile G și G' și între lanțul eulerian deschis μ și ciclul eulerian C din demonstrația corolarului anterior, precum și Algoritmul 6.1.1, se obține un algoritm pentru determinarea lanțurilor euleriene deschise într-un graf dat.

Teorema 6.1.2 (de caracterizare a grafurilor euleriene orientate). *Un graf orientat fără noduri izolate este eulerian dacă și numai dacă este conex și $d^+(x) = d^-(x)$, pentru orice nod x .*

Demonstrație. Demonstrația este analogă cu cea a Teoremei 6.1.1, înlocuind "ciclu" cu "circuit" și " $d(x) = \text{par}$ " cu " $d^+(x) = d^-(x)$ ".

Existența unui circuit se obține pornind dintr-un nod arbitrar și parcurgând succesiv arce cât timp este posibil (revenirea în nodul inițial este asigurată de condiția $d^+(x) = d^-(x) \forall x$). \square

Corolarul 6.1.2. *Un graf orientat fără noduri izolate are un drum eulerian deschis dacă și numai dacă este conex și are două noduri x și y astfel încât $d^+(x) = d^-(x) + 1$, $d^+(y) = d^-(y) - 1$, $d^+(z) = d^-(z)$ pentru orice nod z diferit de x și y . În plus, orice drum eulerian are ca extremitate inițială nodul x și ca extremitate finală nodul y .*

Demonstrație. Demonstrația este analogă cu cea a Corolarului 6.1.1, utilizând acum Teorema 6.1.2. \square

Exemplul 6.1.5. Graful din Exemplul 6.1.2 este conex, fără noduri izolate și $d^+(1) = d^-(1) + 1$, $d^+(4) = d^-(4) - 1$, $d^+(z) = d^-(z) \forall z \in V \setminus \{1, 4\}$, deci graful are drumuri euleriene de forma $(1, \dots, 4)$.

Observația 6.1.7. Demonstrația Teoremei 6.1.2 este constructivă, ea indicând un algoritm de determinare a unui circuit eulerian într-un graf orientat ce verifică ipotezele teoremei. De asemenea, analog Observației 6.1.6, acest algoritm poate fi utilizat și pentru determinarea unui drum eulerian deschis într-un graf orientat ce verifică ipotezele Corolarului 6.1.2.

6.2 Grafuri hamiltoniene

Definiția 6.2.1. a) Fie $G = (V, E)$ un graf. Un lanț elementar, ciclu elementar, drum elementar sau circuit elementar în graful G ce conține toate nodurile lui G se numește **hamiltonian**.

b) Un graf neorientat se numește **trasabil** dacă are un singur nod sau are (cel puțin) un lanț hamiltonian deschis.

c) Un graf neorientat se numește **hamiltonian** dacă are (cel puțin) un ciclu hamiltonian.

d) Un graf orientat se numește **trasabil** dacă are un singur nod sau are (cel puțin) un drum hamiltonian deschis.

e) Un graf orientat se numește **hamiltonian** dacă are (cel puțin) un circuit hamiltonian.

Observația 6.2.1. Orice graf hamiltonian este trasabil, deoarece prin eliminarea unei muchii a unui ciclu hamiltonian rămâne un lanț hamiltonian deschis, iar prin eliminarea unui arc al unui circuit hamiltonian rămâne un drum hamiltonian deschis. Reciproca nu este adevărată. De exemplu, un graf format dintr-un drum deschis elementar este trasabil, dar nu este hamiltonian.

Exemplul 6.2.1. Graful orientat din Exemplul 6.1.2 este hamiltonian, un circuit hamiltonian fiind $C = (1, 2, 3, 5, 4, 6, 1)$. El este și trasabil, 6 drumuri hamiltoniene deschise obținându-se prin eliminarea a câte unuia din arcele circuitului C și anume $\mu_1 = C \setminus \{(1, 2)\} = (2, 3, 5, 4, 6, 1)$, $\mu_2 = C \setminus \{(2, 3)\} = (3, 5, 4, 6, 1, 2)$, $\mu_3 = C \setminus \{(3, 5)\} = (5, 4, 6, 1, 2, 3)$, $\mu_4 = C \setminus \{(5, 4)\} = (4, 6, 1, 2, 3, 5)$, $\mu_5 = C \setminus \{(4, 6)\} = (6, 1, 2, 3, 5, 4)$, $\mu_6 = C \setminus \{(6, 1)\} = (1, 2, 3, 5, 4, 6)$.

Definiția 6.2.2. Fie $G = (V, E)$ un graf. Un nod $x \in V$ se numește **punct (nod, vârf) de articulație** al grafului G dacă $G_x \setminus \{x\}$ este un graf neconex, unde G_x este componenta conexă a lui G care conține nodul x .

Exemplul 6.2.2. Graful din Exemplul 6.1.1 are puncte de articulație și anume nodurile 2, 3 și 5.

Lema 6.2.1. a) Orice graf neorientat hamiltonian este conex și nu are puncte de articulație.
 b) Orice graf orientat hamiltonian este tare-conex și nu are puncte de articulație.

Demonstrație. a) Un ciclu hamiltonian C al grafului G este (induce) un graf parțial, conex și fără puncte de articulație al lui G , deci și graful G este conex și nu are puncte de articulație. Analog se obține afirmația de la punctul b), utilizând faptul că un circuit hamiltonian este tare-conex. \square

Observația 6.2.2. Reciprocele afirmațiilor din lema anterioară nu sunt adevărate. De exemplu, graful orientat din Figura 6.2.1 este tare-conex și nu are puncte de articulație, dar nu este hamiltonian (deoarece nodurile 2 și 4 au gradele egale cu 2, deci un circuit hamiltonian C ar trebui să conțină arcele $(1, 2), (2, 3), (3, 4), (4, 1)$, iar pentru a conține și nodul 5 circuitul C ar trebui să traverseze de două ori nodul 1, fals). Analog, dacă eliminăm orientările arcelor acestui graf, obținem un graf neorientat conex și fără puncte de articulație, dar care nu este hamiltonian.

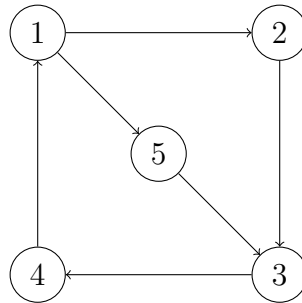


Figura 6.2.1:

Exemplul 6.2.3. Graful eulerian din Exemplul 6.1.1 nu este hamiltonian, deoarece are puncte de articulație.

Exemplul 6.2.4. Graful reprezentat în Figura 6.2.2 este hamiltonian, un ciclu hamiltonian al său fiind $C = [1, 2, 3, 4, 8, 7, 6, 5, 1]$. Acest graf nu este eulerian, având 4 noduri de grade impare.

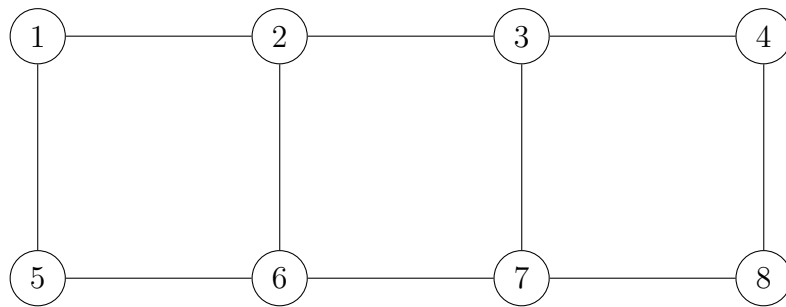


Figura 6.2.2:

Observația 6.2.3. Spre deosebire de problemele euleriene studiate în secțiunea anterioară, problemele de testare a hamiltonității sau a trasabilității unui graf și de determinare a unui ciclu, lanț deschis, circuit sau drum deschis hamiltonian optim într-un graf ponderat sunt probleme pentru care nu se cunosc (până în prezent) algoritmi de rezolvare cu complexitate polinomială (numite și **probleme NP**). Dispunem totuși de numeroase condiții fie necesare, fie suficiente, pentru ca un graf dat să fie hamiltonian sau trasabil. Câteva astfel de condiții vor fi prezentate în continuare.

Observația 6.2.4. Este evident că într-un graf cu $n \geq 3$ noduri orice lanț, ciclu, drum sau circuit hamiltonian nu utilizează bucle și nici muchii sau arce multiple, deci este suficient să studiem hamiltonicitatea pentru grafurile simple.

Lema 6.2.2 (Bondy, Chvátal). Fie $G = (V, E)$ un graf neorientat simplu cu $n \geq 3$ noduri. Fie $x, y \in V$ astfel încât

$$x \neq y, [x, y] \notin E \text{ și } d(x) + d(y) \geq n.$$

Atunci graful G este hamiltonian dacă și numai dacă graful

$$G + [x, y] = (V, E \cup \{[x, y]\})$$

este hamiltonian.

Demonstrație. Evident, dacă G este hamiltonian atunci orice ciclu hamiltonian în G este ciclu hamiltonian și în $G + [x, y]$, deci $G + [x, y]$ este hamiltonian.

Reciproc, fie $G + [x, y]$ hamiltonian și C un ciclu hamiltonian în acest graf.

Cazul 1) Dacă C nu conține muchia $[x, y]$, atunci C este un ciclu hamiltonian în graful G .

Cazul 2) Dacă C conține muchia $[x, y]$, atunci prin eliminarea muchiei $[x, y]$ din ciclul C obținem un lanț hamiltonian

$$\mu = [x = v_1, v_2, \dots, v_n = y]$$

în graful G . Fie

$$\begin{aligned} A &= \{i \in \{1, \dots, n-1\} \mid [x, v_{i+1}] \in E\}, \\ B &= \{i \in \{1, \dots, n-1\} \mid [v_i, y] \in E\}. \end{aligned}$$

Avem $\text{card}(A) = d(x)$ și $\text{card}(B) = d(y)$, deci, conform ipotezei,

$$\text{card}(A) + \text{card}(B) \geq n.$$

Cum $A \cup B \subseteq \{1, \dots, n-1\}$, avem $\text{card}(A \cup B) \leq n-1$, deci

$$\text{card}(A \cap B) = \text{card}(A) + \text{card}(B) - \text{card}(A \cup B) \geq n - (n-1) = 1,$$

adică $A \cap B \neq \emptyset$. Fie $k \in A \cap B$. Atunci

$$[x = v_1, v_2, \dots, v_k, y = v_n, v_{n-1}, \dots, v_{k+1}, x]$$

este un ciclu hamiltonian în graful G . □

Definiția 6.2.3. Fie G un graf neorientat simplu cu $n \geq 3$ noduri. **Închiderea lui G** , notată cu $\text{cl}(G)$, este graful neorientat simplu obținut din G prin adăugarea repetată a tuturor muchiilor $[x, y]$ între noduri distincte și neadiacente x, y cu $d(x) + d(y) \geq n$ în graful curent.

Observația 6.2.5. Dacă $H = \text{cl}(G)$, atunci pentru orice noduri distincte x, y cu $[x, y] \notin E(H)$ avem $d_H(x) + d_H(y) < n$, unde $E(H)$ este mulțimea muchiilor grafului H .

Observația 6.2.6. Graful $\text{cl}(G)$ este bine definit, nedepinzând de ordinea de adăugare a muchiilor. Acest fapt poate fi demonstrat prin inducție după numărul muchiilor adăugate.

Exemplul 6.2.5. Fie G graful reprezentat în Figura 6.2.3.

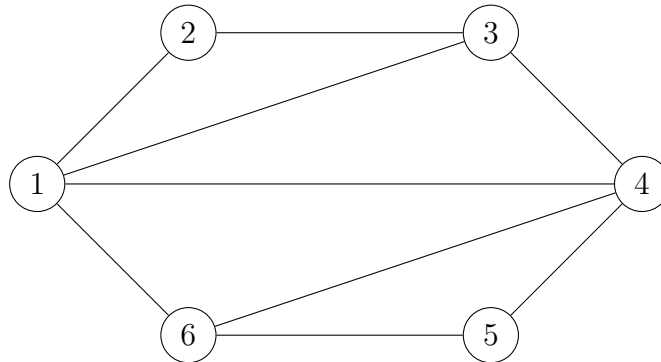


Figura 6.2.3:

Adăugând succesiv muchiile $[1, 5]$ (deoarece $d(1) + d(5) = 4 + 2 \geq n = 6$), $[2, 4]$ (deoarece $d(2) + d(4) = 2 + 4 \geq n$, în graful curent), $[2, 5]$ (deoarece $d(2) + d(5) = 3 + 3 \geq n$, în graful curent!), $[2, 6]$, $[3, 5]$ și $[3, 6]$, obținem închiderea $\text{cl}(G) = K_6$, unde K_6 este graful complet cu 6 noduri.

Lema 6.2.3 (Lema închiderii; Bondy, Chvátal). Fie G un graf neorientat simplu cu $n \geq 3$ noduri. Atunci G este hamiltonian dacă și numai dacă graful $\text{cl}(G)$ este hamiltonian.

Demonstrație. Concluzia este evidentă conform Lemei 6.2.2 și definiției anterioare. \square

Lema 6.2.4. Fie G un graf neorientat simplu cu $n \geq 3$ noduri. Dacă avem $\text{cl}(G) = K_n$, atunci G este hamiltonian.

Demonstrație. Concluzia rezultă folosind lema anterioară și faptul că graful complet K_n , $n \geq 3$, este evident hamiltonian (orice succesiune de forma $[v_1, v_2, \dots, v_n, v_1]$ cu nodurile v_1, \dots, v_n distincte este un ciclu hamiltonian în K_n). \square

Observația 6.2.7. Reciproca lemei anterioare nu este adevărată. De exemplu graful G reprezentat în Figura 6.2.4 este hamiltonian, dar $\text{cl}(G) = G \neq K_6$.

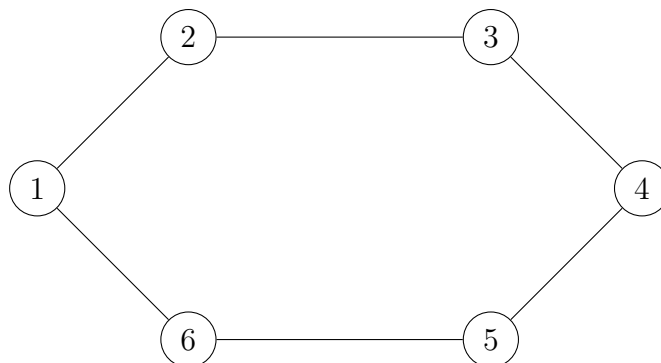


Figura 6.2.4:

Teorema 6.2.1 (Chvátal). Fie $G = (V, E)$ un graf neorientat simplu cu $n \geq 3$ noduri având gradele

$$d_1 \leq d_2 \leq \dots \leq d_n.$$

Dacă pentru orice $k \in \mathbb{N}^*$ este verificată proprietatea

$$d_k \leq k < \frac{n}{2} \Rightarrow d_{n-k} \geq n - k,$$

atunci G este hamiltonian.

Demonstrație. Dacă am avea $d_1 = 0$, atunci din ipoteză ar rezulta că $d_{n-1} \geq n - 1$, și cum graful G este simplu s-ar obține că nodul de grad d_{n-1} ar fi adiacent cu toate celelalte noduri, contradicție cu $d_1 = 0$. Deci $d_1 \geq 1$ și astfel graful G nu are noduri izolate.

Demonstrăm că are loc egalitatea $\text{cl}(G) = K_n$ și conform lemei anterioare rezultă astfel că G este hamiltonian. Fie

$$H = \text{cl}(G).$$

Presupunem prin reducere la absurd că $H \neq K_n$, deci $\exists x, y \in V$, $x \neq y$ astfel încât $[x, y] \notin E(H)$ (unde $E(H)$ reprezintă mulțimea muchiilor lui H). Conform Observației 6.2.5, pentru orice $x, y \in V$, $x \neq y$ cu $[x, y] \notin E(H)$ avem

$$d_H(x) + d_H(y) \leq n - 1. \quad (6.2.1)$$

Fie atunci $x, y \in V$, $x \neq y$ cu $[x, y] \notin E(H)$ și de sumă $d_H(x) + d_H(y)$ maximă.

Presupunem că $d_H(x) \leq d_H(y)$. Fie $k = d_H(x)$.

Evident, $k \geq d_G(x) \geq 1$, iar din (6.2.1) obținem

$$k \leq \frac{n-1}{2} \text{ și } d_H(y) \leq n - 1 - k. \quad (6.2.2)$$

Fie

$$A = \{v \in V \mid v \neq y, [v, y] \notin E(H)\}.$$

Avem $\text{card}(A) = n - 1 - d_H(y)$, deci din (6.2.2) avem

$$\text{card}(A) \geq k. \quad (6.2.3)$$

Din alegerea lui x și y , pentru orice $v \in A$ avem

$$d_H(v) + d_H(y) \leq d_H(x) + d_H(y),$$

deci $d_H(v) \leq d_H(x)$. Cum $d_G(v) \leq d_H(v)$ obținem că

$$d_G(v) \leq k, \quad \forall v \in A. \quad (6.2.4)$$

Din (6.2.3) și (6.2.4) rezultă că în G există cel puțin k noduri v având $d_G(v) \leq k$, și folosind ordonarea $d_1 \leq \dots \leq d_n$ a gradelor în G deducem că

$$d_k \leq k. \quad (6.2.5)$$

Fie

$$B = \{v \in V \mid v \neq x, [x, v] \notin E(H)\}.$$

Avem $\text{card}(B) = n - 1 - d_H(x)$, deci

$$\text{card}(B) = n - 1 - k. \quad (6.2.6)$$

Din alegerea lui x și y , pentru orice $v \in B$ avem

$$d_H(x) + d_H(v) \leq d_H(x) + d_H(y),$$

deci $d_H(v) \leq d_H(y)$, și conform (6.2.2) rezultă că

$$d_H(v) \leq n - 1 - k.$$

Cum $d_G(v) \leq d_H(v)$ obținem că

$$d_G(v) \leq n - 1 - k, \quad \forall v \in B. \quad (6.2.7)$$

Din (6.2.6) și (6.2.7) rezultă că în G există cel puțin $n - 1 - k$ noduri v diferite de x și având $d_G(v) \leq n - 1 - k$. Cum

$$d_G(x) \leq d_H(x) = k \leq n - 1 - k,$$

rezultă că în G există cel puțin $n - k$ noduri v având $d_G(v) \leq n - 1 - k$, și folosind ordonarea $d_1 \leq \dots \leq d_n$ a gradelor în G deducem că

$$d_{n-k} \leq n - 1 - k. \quad (6.2.8)$$

Din (6.2.2), (6.2.5) și (6.2.8) obținem

$$d_k \leq k < \frac{n}{2} \text{ și } d_{n-k} < n - k,$$

contradicție cu ipoteza teoremei. Demonstrația este încheiată. \square

Observația 6.2.8. Dacă graful G verifică ipoteza Teoremei lui Chvátal atunci $\text{cl}(G) = K_n$. Reciproca nu este adevărată. De exemplu, graful G din Exemplul 6.2.5 are $\text{cl}(G) = K_n$, dar nu verifică ipoteza Teoremei lui Chvátal (are gradele, în ordine crescătoare, 2, 2, 3, 3, 4, 4, deci $d_2 \leq 2 < \frac{n}{2}$ dar $d_4 < 4$).

Astfel nici reciproca Teoremei lui Chvátal nu este adevărată.

Corolarul 6.2.1 (Teorema lui Bondy). *Fie G un graf neorientat simplu cu $n \geq 3$ noduri având gradele*

$$d_1 \leq d_2 \leq \dots \leq d_n.$$

Dacă pentru orice $p, q \in \mathbb{N}^$ este verificată proprietatea*

$$d_p \leq p, \quad d_q \leq q, \quad p \neq q \Rightarrow d_p + d_q \geq n,$$

atunci G este hamiltonian.

Demonstrație. Se arată că graful G verifică ipoteza Teoremei lui Chvátal și astfel este hamiltonian. \square

Observația 6.2.9. Teorema lui Chvátal generalizează Teorema lui Bondy. Această generalizare este strictă. De exemplu, graful reprezentat în Figura 6.2.5 are gradele, în ordine crescătoare, 3, 3, 3, 4, 5, 5, 5, 6, deci verifică ipoteza Teoremei lui Chvátal dar nu verifică ipoteza Teoremei lui Bondy ($d_3 = 3 \leq 3$, $d_4 = 4 \leq 4$, dar $d_3 + d_4 = 7 < n = 8$).

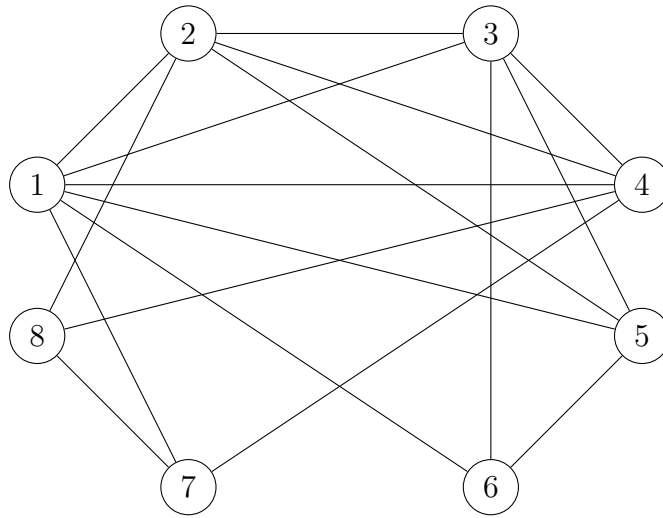


Figura 6.2.5:

Corolarul 6.2.2 (Teorema lui Pósa). Fie G un graf neorientat simplu cu $n \geq 3$ noduri având gradele

$$d_1 \leq d_2 \leq \dots \leq d_n.$$

Dacă

$$\begin{cases} d_k > k, \quad \forall k < \frac{n-1}{2}, \\ d_{k+1} > k, \quad \text{pentru } k = \frac{n-1}{2} \text{ și } n = \text{impar}, \end{cases}$$

atunci G este hamiltonian.

Demonstrație. Se arată că graful G verifică ipoteza Teoremei lui Bondy (corolarul anterior) și astfel este hamiltonian. \square

Observația 6.2.10. Teorema lui Bondy generalizează Teorema lui Pósa. Generalizarea este strictă. De exemplu, graful reprezentat în Figura 6.2.6 are gradele, ordonate crescător, 2, 2, 4, 4, 4, 4, deci verifică ipoteza Teoremei lui Bondy, dar nu verifică ipoteza Teoremei lui Pósa.

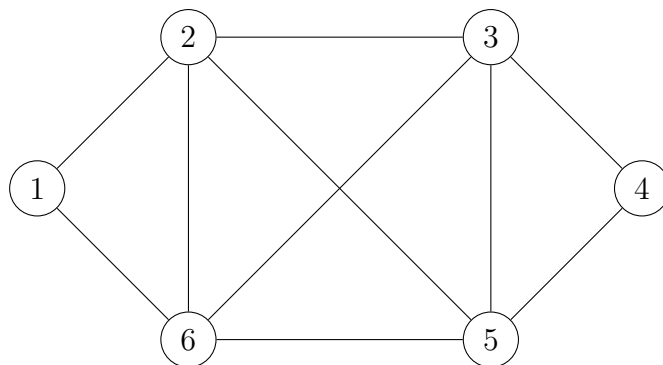


Figura 6.2.6:

Corolarul 6.2.3 (Teorema lui Ore). Fie $G = (V, E)$ un graf neorientat simplu cu $n \geq 3$ noduri. Dacă pentru orice două noduri x, y distincte și neadiacente avem

$$d(x) + d(y) \geq n,$$

atunci G este hamiltonian.

Demonstrație. Se arată că graful G verifică ipoteza Teoremei lui Pósa (corolarul anterior) și astfel este hamiltonian. \square

Observația 6.2.11. Teorema lui Pósa generalizează Teorema lui Ore. Această generalizare este strictă. De exemplu, graful reprezentat în Figura 6.2.7 are gradele, ordonate crescător, 2, 3, 3, 4, 4, 4, deci verifică ipoteza Teoremei lui Pósa, dar nu verifică ipoteza Teoremei lui Ore.

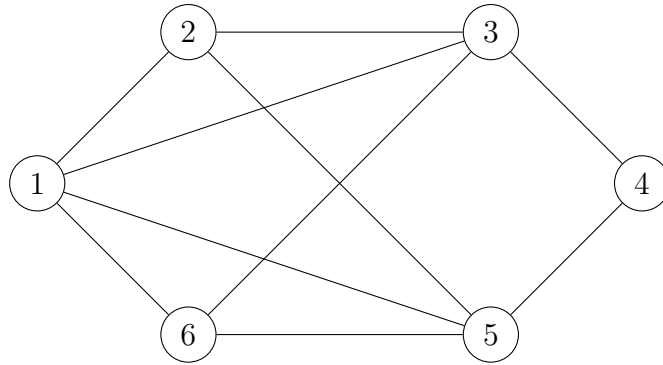


Figura 6.2.7:

Corolarul 6.2.4 (Teorema lui Dirac). Fie $G = (V, E)$ un graf neorientat simplu cu $n \geq 3$ noduri. Dacă

$$d(x) \geq \frac{n}{2}, \quad \forall x \in V,$$

atunci G este hamiltonian.

Demonstrație. Se arată că graful G verifică ipoteza Teorema lui Ore (corolarul anterior), deci este hamiltonian. \square

Observația 6.2.12. Teorema lui Ore generalizează Teorema lui Dirac. Această generalizare este strictă. De exemplu, graful reprezentat în Figura 6.2.8 verifică ipoteza Teoremei lui Ore, dar nu verifică ipoteza Teoremei lui Dirac.

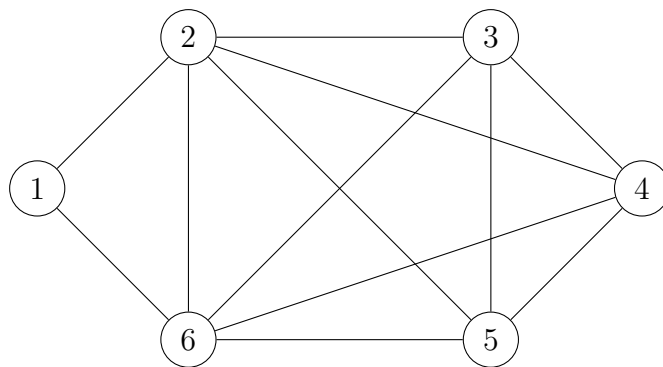


Figura 6.2.8:

Corolarul 6.2.5. Fie $G = (V, E)$ un graf neorientat simplu cu $n \geq 2$ noduri. Dacă

$$d(x) \geq \frac{n-1}{2}, \quad \forall x \in V,$$

atunci G este trasabil.

Demonstrație. Fie $v_0 \notin V$ un nod nou și fie graful neorientat $G' = (V', E')$ definit prin

$$V' = V \cup \{v_0\}, \quad E' = E \cup \{[v_0, x] \mid x \in V\}.$$

Evident, G' este un graf simplu cu $n' = n + 1 \geq 3$ noduri,

$$d_{G'}(x) = d(x) + 1 \geq \frac{n+1}{2} = \frac{n'}{2}, \quad \forall x \in V,$$

$$d_{G'}(v_0) = n \geq \frac{n+1}{2} = \frac{n'}{2},$$

deci conform Teoremei lui Dirac (corolarul anterior) rezultă că G' este hamiltonian. Fie C un ciclu hamiltonian în G' . Atunci lanțul deschis obținut prin eliminarea nodului v_0 din ciclul C este elementar (deoarece ciclul C este elementar) și conține toate nodurile lui G (deoarece ciclul C conține toate nodurile lui G'), deci este un lanț hamiltonian în G și astfel graful G este trasabil. \square

În continuare studiem hamiltoneitatea pentru grafurile orientate.

Definiția 6.2.4. Fie G un graf orientat și H un subgraf al lui G . Un H -ocol în graful G este un drum deschis elementar μ de lungime $l(\mu) \geq 2$ având doar extremitățile în H .

Observația 6.2.13. În contextul definiției anterioare avem $2 \leq \text{card}(V(H)) \leq \text{card}(V(G)) - 1$ (unde $V(G)$ și $V(H)$ reprezintă mulțimile de noduri ale grafurilor G , respectiv H).

Lema 6.2.5 (Lema ocolului). Fie G un graf orientat tare-conex fără puncte de articulație și fie H un subgraf al lui G astfel încât

$$2 \leq \text{card}(V(H)) \leq \text{card}(V(G)) - 1.$$

Atunci G conține cel puțin un H -ocol.

Demonstrație. Cum $V(G) \setminus V(H) \neq \emptyset$ și G este conex, rezultă că există două noduri x și y adiacente în G cu $x \in V(H)$ și $y \in V(G) \setminus V(H)$. Fie $z \in V(H)$, $z \neq x$. Cum x nu este punct de articulație în G , rezultă că graful $G \setminus \{x\}$ este conex, deci există un lanț elementar de la y la z în G care nu conține nodul x . Fie $\mu = [y = t_0, t_1, \dots, t_r = z]$ acest lanț și fie $j \in \{1, \dots, r\}$ cel mai mic indice astfel încât $t_j \in V(H)$ (există, deoarece $t_r = z \in V(H)$). Atunci $[x, y, t_1, \dots, t_j]$ este un lanț deschis elementar de lungime mai mare sau egală cu 2 având doar extremitățile x și t_j în H .

Fie acum $\mu^* = [v_0, v_1, \dots, v_k]$ un lanț deschis elementar de lungime $l(\mu^*) = k \geq 2$ având doar extremitățile v_0 și v_k în H (există, conform construcției anterioare) cu proprietatea că are un număr minim de arce de forma $(v_i, v_{i-1}) \in E(G)$ (numite **arce inverse** pentru lanțul μ^*). Demonstrăm prin reducere la absurd că acest număr minim este chiar egal cu zero și astfel va rezulta că toate arcele lui μ^* au forma $(v_{i-1}, v_i) \in E(G)$ (numite **arce directe** pentru lanțul μ^*), adică μ^* este un drum de la v_0 la v_k , deci μ^* este un H -ocol în G .

Presupunem prin absurd că μ^* conține un arc invers $(v_i, v_{i-1}) \in E(G)$, unde $i \in \{1, \dots, k\}$. Cum G este tare-conex, rezultă că există un drum elementar în G de forma $\mu_1 = (s = w_0, w_1, \dots, w_p = v_i)$ cu $s \in V(H)$, $w_1, \dots, w_p \in V(G) \setminus V(H)$. Dacă $i = k$, atunci $p = 0$ și $\mu_1 = (s)$, cu $s = v_k$.

Analog, există un drum elementar în G de forma $\mu_2 = (v_{i-1} = u_0, u_1, \dots, u_q = t)$ cu $t \in V(H)$, $u_0, \dots, u_{q-1} \in V(G) \setminus V(H)$. Dacă $i = 1$, atunci $q = 0$ și $\mu_2 = (t)$, cu $t = v_0$. Avem două cazuri.

Cazul 1) $s \neq t$. Atunci $(s = w_0, w_1, \dots, w_p = v_i, v_{i-1} = u_0, u_1, \dots, u_q = t)$ este un drum în G cu proprietatea că $s, t \in V(H)$ și $w_1, \dots, w_p, u_0, \dots, u_{q-1} \in V(G) \setminus V(H)$, deci, eliminând toate eventualele porțiuni dintre noduri egale, se obține un H -ocol de la s la t în G , contradicție cu alegerea lui μ^* .

Cazul 2) $s = t$. Atunci cum $v_0 \neq v_k$, rezultă că $t \neq v_0$ sau $s \neq v_k$. Avem două subcazuri.

a) $t \neq v_0$. Atunci $\mu' = [v_0, v_1, \dots, v_{i-1} = u_0, u_1, \dots, u_q = t]$ este un lanț în G cu proprietatea că $v_0, t \in V(H)$ și $v_1, \dots, v_{i-1}, u_1, \dots, u_{q-1} \in V(G) \setminus V(H)$. Arcele $(u_0, u_1), \dots, (u_{q-1}, u_q)$ sunt directe în μ' , deci arcele inverse ale lui μ' se găsesc doar în porțiunea $[v_0, v_1, \dots, v_{i-1}]$. Cum această porțiune apare și în μ^* și nu conține arcul (v_i, v_{i-1}) , care este arc invers în μ^* , rezultă că numărul de arce inverse ale lui μ' este mai mic sau egal cu $p - 1$. Eliminând toate eventualele porțiuni dintre noduri egale din μ' , se obține un lanț elementar deschis de la v_0 la t de lungime mai mare sau egală cu 2 cu proprietatea că doar extremitățile v_0 și t sunt în H și având numărul de arce inverse mai mic sau egal cu $p - 1$, contradicție cu alegerea lui μ^* .

b) $s \neq v_k$. Atunci $\mu'' = [s = w_0, w_1, \dots, w_p = v_i, v_{i+1}, \dots, v_k]$ este un lanț în G cu proprietatea că $s, v_k \in V(H)$ și $w_1, \dots, w_p, v_{i+1}, \dots, v_{k-1} \in V(G) \setminus V(H)$. Arcele $(w_1, w_2), \dots, (w_{p-1}, w_p)$ sunt directe în μ'' , deci arcele inverse ale lui μ'' se găsesc doar în porțiunea $\mu' = [v_i, v_{i+1}, \dots, v_k]$. Cum această porțiune apare și în μ^* și nu conține arcul (v_i, v_{i-1}) , care este arc invers în μ^* , rezultă că numărul de arce inverse ale lui μ'' este mai mic sau egal cu $p - 1$. Eliminând toate eventualele porțiuni dintre noduri egale din μ'' , se obține un lanț elementar deschis de la s la v_k de lungime mai mare sau egală cu 2 cu proprietatea că doar extremitățile s și v_k sunt în H și având numărul de arce inverse mai mic sau egal cu $p - 1$, contradicție cu alegerea lui μ^* .

Demonstrația prin reducere la absurd este încheiată. \square

Lema 6.2.6 (Kőnig-Veress). Fie G un graf orientat simplu, μ un drum elementar deschis de la x la y în G și $z \in V(G) \setminus V(\mu)$.

a) Dacă nu există niciun drum elementar μ' de la x la y în G astfel încât $V(\mu') = V(\mu) \cup \{z\}$, atunci

$$d_\mu(z) \leq l(\mu) + 2,$$

unde $d_\mu(z)$ reprezintă numărul de noduri ale lui μ ce sunt adiacente cu z în G , numit **gradul nodului z relativ la drumul μ** , iar $l(\mu)$ reprezintă lungimea drumului μ .

b) Dacă μ este un drum elementar deschis de lungime maximă în G , atunci

$$d_\mu(z) \leq l(\mu).$$

Demonstrație. a) Fie $\mu = (x = v_0, v_1, \dots, v_k = y)$, deci $l(\mu) = k$. Fie

$$\begin{aligned} A &= \{i \in \{1, \dots, k\} \mid (z, v_i) \in E(G)\}, \\ B &= \{i \in \{1, \dots, k\} \mid (v_{i-1}, z) \in E(G)\}. \end{aligned}$$

Notând cu $d_\mu^+(z)$ numărul de noduri $v_i \in V(\mu)$ cu proprietatea că $(z, v_i) \in E(G)$ (numit **gradul de ieșire al nodului z relativ la drumul μ**) și cu $d_\mu^-(z)$ numărul de noduri $v_i \in V(\mu)$ cu proprietatea că $(v_i, z) \in E(G)$ (numit **gradul de intrare al nodului z relativ la drumul μ**), avem

$$\begin{aligned} d_\mu^+(z) &= \begin{cases} \text{card}(A) + 1, & \text{dacă } (z, v_0) \in E(G), \\ \text{card}(A), & \text{dacă } (z, v_0) \notin E(G), \end{cases} \\ d_\mu^-(z) &= \begin{cases} \text{card}(B) + 1, & \text{dacă } (v_k, z) \in E(G), \\ \text{card}(B), & \text{dacă } (v_k, z) \notin E(G) \end{cases} \end{aligned}$$

și $d_\mu(z) = d_\mu^+(z) + d_\mu^-(z)$.

Dacă ar exista un indice $i \in A \cap B$, atunci

$$\mu' = (x = v_0, v_1, \dots, v_{i-1}, z, v_i, v_{i+1}, \dots, v_k = y)$$

ar fi un drum elementar de la x la y în G astfel încât $V(\mu') = V(\mu) \cup \{z\}$, contradicție cu ipoteza. Deci $A \cap B = \emptyset$.

Folosind relațiile anterioare și faptul că $A \cup B \subseteq \{1, 2, \dots, k\}$, rezultă că

$$d_\mu(z) = d_\mu^+(z) + d_\mu^-(z) \leq \text{card}(A) + \text{card}(B) + 2 = \text{card}(A \cup B) + 2 \leq k + 2.$$

b) Dacă μ este un drum elementar deschis de lungime maximă în G , atunci utilizând notațiile și relațiile de la demonstrația punctului a), avem $(z, v_0) \notin E(G)$ (deoarece în caz contrar $(z, v_0, v_1, \dots, v_k)$ ar fi un drum elementar deschis în G de lungime $l(\mu) + 1$, contradicție), $(v_k, z) \notin E(G)$ (deoarece în caz contrar $(v_0, v_1, \dots, v_k, z)$ ar fi un drum elementar deschis în G de lungime $l(\mu) + 1$, contradicție) și, din nou, $A \cap B = \emptyset$ (deoarece dacă ar exista $i \in A \cap B$, atunci $\mu' = (v_0, v_1, \dots, v_{i-1}, z, v_i, v_{i+1}, \dots, v_k)$ ar fi un drum elementar deschis în G de lungime $l(\mu) + 1$, contradicție), deci

$$d_\mu(z) = d_\mu^+(z) + d_\mu^-(z) = \text{card}(A) + \text{card}(B) = \text{card}(A \cup B) \leq k.$$

□

Teorema 6.2.2 (Meyniel). Fie $G = (V, E)$ un graf orientat simplu, tare-conex, cu $n \geq 2$ noduri. Dacă pentru orice două noduri x, y distincte și neadiacente avem

$$d(x) + d(y) \geq 2n - 1,$$

atunci G este hamiltonian.

Demonstrație. Arătăm că G nu conține puncte de articulație prin reducere la absurd. Într-adevăr, dacă ar exista $v \in V$ astfel încât $G \setminus \{v\}$ ar fi neconex, atunci notând cu A și B două componente conexe ale grafului $G \setminus \{v\}$ ar rezulta că pentru orice noduri $a \in A$ și $b \in B$ avem

$$d^+(a) \leq \text{card}(A), \quad d^-(a) \leq \text{card}(A), \quad d^+(b) \leq \text{card}(B), \quad d^-(b) \leq \text{card}(B)$$

(deoarece toate nodurile adiacente cu a în G sunt din $\{v\} \cup A \setminus \{a\}$, toate nodurile adiacente cu b în G sunt din $\{v\} \cup B \setminus \{b\}$, iar G este simplu), deci nodurile a, b sunt distincte și neadiacente și

$$d(a) + d(b) = d^+(a) + d^-(a) + d^+(b) + d^-(b) \leq 2(\text{card}(A) + \text{card}(B)) \leq 2(n - 1),$$

contradicție cu ipoteza teoremei.

Graful G este tare-conex, deci conține circuite elementare. Într-adevăr, pentru orice două noduri distincte s și t există drumuri elementare $(s = x_0, x_1, \dots, x_p = t)$ și $(t = y_0, y_1, \dots, y_q = s)$ în G , deci luând cel mai mic indice $i \in \{1, \dots, q\}$ pentru care există $j \in \{0, \dots, p\}$ astfel încât $y_i = x_j$ (există astfel de indici, deoarece $y_q = x_0$), rezultă că $(t = y_0, y_1, \dots, y_i = x_j, x_{j+1}, \dots, x_p = t)$ este un circuit elementar în G .

Fie C un circuit elementar de lungime maximă în G . Demonstrăm că circuitul C are lungimea $l(C) = n$ prin reducere la absurd și astfel va rezulta că C este un circuit hamiltonian în G .

Presupunem prin absurd că $l(C) < n$. Conform Lemei ocolului rezultă că G conține cel puțin un C -ocol. Fie $\mu_1 = (x, \dots, y)$ un C -ocol în G cu proprietatea că subdrumul elementar μ_2 de la x la y al circuitului C este de lungime minimă.

Fie μ_3 subdrumul elementar de la y la x al circuitului C .

Notăm

$$V_1 = V(\mu_1) \setminus \{x, y\}, V_2 = V(\mu_2) \setminus \{x, y\}, V_3 = V(\mu_3), V_4 = V \setminus (V_1 \cup V_2 \cup V_3).$$

Avem $l(\mu_1) \geq 2$ (din definiția C -ocolului), deci $V_1 \neq \emptyset$.

De asemenea, avem și $V_2 \neq \emptyset$ (deoarece în caz contrar circuitul C' obținut prin parcurgerea întâi a drumului μ_1 apoi a drumului μ_3 ar fi un circuit elementar în G de lungime $l(C') = l(\mu_1) + l(\mu_3) = l(\mu_1) + l(C) - l(\mu_2) = l(\mu_1) + l(C) - 1 \geq l(C) + 1$, contradicție cu alegerea lui C).

Fie nodurile $v_1 \in V_1$ și $v_2 \in V_2$ arbitrar fixate.

Graful G fiind simplu, avem

$$d_{V_1}(v_1) \leq 2\text{card}(V_1) - 2 \quad (6.2.9)$$

(unde, pentru orice $v \in V$ și $U \subseteq V$, $d_U(v)$ reprezintă numărul de arce din G având o extremitate egală cu v și cealaltă extremitate din U , numit **gradul nodului v relativ la U**).

Avem $(v_1, v_2) \notin E$ (deoarece în caz contrar drumul μ'_1 obținut prin parcurgerea întâi a subdrumului dintre x și v_1 al lui μ_1 apoi a arcului (v_1, v_2) ar fi un C ocol în G de la x la v_2 cu proprietatea că subdrumul elementar μ'_2 de la x la v_2 al circuitului C este de lungime $l(\mu'_2) < l(\mu_2)$, contradicție cu alegerea lui μ_1). Analog, avem $(v_2, v_1) \notin E$ (deoarece în caz contrar drumul μ''_1 obținut prin parcurgerea întâi a arcului (v_2, v_1) apoi a subdrumului dintre v_1 și y al lui μ_1 ar fi un C ocol în G de la v_2 la y cu proprietatea că subdrumul elementar μ''_2 de la v_2 la y al circuitului C este de lungime $l(\mu''_2) < l(\mu_2)$, contradicție cu alegerea lui μ_1). Cum nodurile $v_1 \in V_1$ și $v_2 \in V_2$ sunt arbitrare, rezultă că

$$d_{V_2}(v_1) = 0, d_{V_1}(v_2) = 0. \quad (6.2.10)$$

Nu există un drum elementar μ'_3 de la y la x în G cu proprietatea că $V(\mu'_3) = V(\mu_3) \cup \{v_1\}$ (deoarece în caz contrar circuitul C'' obținut prin parcurgerea întâi a drumului μ_2 apoi a drumului μ'_3 ar fi un circuit elementar în G de lungime $l(C'') = l(\mu_2) + l(\mu'_3) = l(\mu_2) + l(\mu_3) + 1 = l(C) + 1$, contradicție cu alegerea lui C), deci conform Lemei König-Veress rezultă că $d_{V_3}(v_1) \leq l(\mu_3) + 2$, adică

$$d_{V_3}(v_1) \leq \text{card}(V_3) + 1. \quad (6.2.11)$$

Nu există $z \in V_4$ astfel încât $(v_1, z), (z, v_2) \in E$ (deoarece în caz contrar drumul μ'_1 obținut prin parcurgerea întâi a subdrumului dintre x și v_1 al lui μ_1 apoi a arcului (v_1, z) , apoi a arcului (z, v_2) ar fi un C ocol în G de la x la v_2 cu proprietatea că subdrumul elementar μ'_2 de la x la v_2 al circuitului C este de lungime $l(\mu'_2) < l(\mu_2)$, contradicție cu alegerea lui μ_1). Cum G este simplu rezultă că

$$d_{V_4}^+(v_1) + d_{V_4}^-(v_2) \leq \text{card}(V_4)$$

(unde, pentru orice $v \in V$ și $U \subseteq V$, $d_U^+(v)$ reprezintă numărul de arce din G având extremitatea inițială egală cu v și extremitatea finală din U , numit **gradul de ieșire al nodului v relativ la U** , iar $d_U^-(v)$ reprezintă numărul de arce din G având extremitatea inițială din U și extremitatea finală egală cu v , numit **gradul de intrare al nodului v relativ la U**). Analog, nu există $z \in V_4$ astfel încât $(v_2, z), (z, v_1) \in E$ (deoarece în caz contrar drumul μ''_1 obținut prin parcurgerea întâi a arcului (v_2, z) , apoi a arcului (z, v_1) , apoi a subdrumului dintre v_1 și y al lui μ_1 ar fi un C ocol în G de la v_2 la y cu proprietatea că subdrumul elementar μ''_2 de la v_2 la y al circuitului C este de lungime $l(\mu''_2) < l(\mu_2)$, contradicție cu alegerea lui μ_1), deci

$$d_{V_4}^+(v_2) + d_{V_4}^-(v_1) \leq \text{card}(V_4).$$

Prin adunarea ultimelor două inegalități rezultă că

$$d_{V_4}(v_1) + d_{V_4}(v_2) \leq 2\text{card}(V_4). \quad (6.2.12)$$

Fie acum μ_3^* un drum elementar de lungime maximă de la y la x în G cu proprietatea

$$V_3 \subseteq V(\mu_3^*) \subseteq V_2 \cup V_3$$

(există astfel de drumuri, de exemplu μ_3).

Notăm

$$V_3^* = V(\mu_3^*), \quad V_2^* = V(C) \setminus V_3^*,$$

deci $V_2^* = (V_2 \cup V_3) \setminus V_3^* = V_2 \setminus V_3^*$ (deoarece $V_3 \subseteq V(\mu_3^*) = V_3^*$), prin urmare $V_2^* \subseteq V_2$.

Avem $V_2^* \neq \emptyset$ (deoarece în caz contrar circuitul C^* obținut prin parcurgerea întâi a drumului μ_1 apoi a drumului μ_3^* ar fi un circuit elementar în G de lungime $l(C^*) = l(\mu_1) + l(\mu_3^*) = l(\mu_1) + l(C) - 1 \geq l(C) + 1$, contradicție cu alegerea lui C).

Cum nodul $v_2 \in V_2$ a fost arbitrar fixat, iar $\emptyset \neq V_2^* \subseteq V_2$, în continuare considerăm că

$$v_2 \in V_2^*$$

și rezultatele de mai sus rămân valabile. Graful G fiind simplu, avem

$$d_{V_2^*}(v_2) \leq 2\text{card}(V_2^*) - 2.$$

Nu există un drum elementar μ_3'' de la y la x în G cu proprietatea că $V(\mu_3'') = V(\mu_3^*) \cup \{v_2\}$ (deoarece în caz contrar am avea $V_3 \subseteq V(\mu_3'') \subseteq V_2 \cup V_3$ și $l(\mu_3'') = l(\mu_3^*) + 1$, contradicție cu alegerea lui μ_3^*), deci conform Lemei Kőnig-Veress rezultă că $d_{V_3^*}(v_2) \leq l(\mu_3^*) + 2$, adică

$$d_{V_3^*}(v_2) \leq \text{card}(V_3^*) + 1.$$

Prin adunarea ultimelor două inegalități obținem

$$d_{V_2^*}(v_2) + d_{V_3^*}(v_2) \leq 2\text{card}(V_2^*) + \text{card}(V_3^*) - 1.$$

Cum $\text{card}(V_2^*) + \text{card}(V_3^*) = \text{card}(V(C)) = \text{card}(V_2) + \text{card}(V_3)$ și $\text{card}(V_2^*) \leq \text{card}(V_2)$ (deoarece $V_2^* \subseteq V_2$), rezultă că

$$d_{V_2^*}(v_2) + d_{V_3^*}(v_2) \leq 2\text{card}(V_2) + \text{card}(V_3) - 1.$$

Dar, cum $V_2^* \cup V_3^* = V(C) = V_2 \cup V_3$ și $V_2^* \cap V_3^* = V_2 \cap V_3 = \emptyset$, avem

$$d_{V_2^*}(v_2) + d_{V_3^*}(v_2) = d_{V(C)}(v_2) = d_{V_2}(v_2) + d_{V_3}(v_2),$$

deci inegalitatea anterioară devine

$$d_{V_2}(v_2) + d_{V_3}(v_2) \leq 2\text{card}(V_2) + \text{card}(V_3) - 1. \quad (6.2.13)$$

Evident,

$$d(v_1) = d_{V_1}(v_1) + d_{V_2}(v_1) + d_{V_3}(v_1) + d_{V_4}(v_1), \quad d(v_2) = d_{V_1}(v_2) + d_{V_2}(v_2) + d_{V_3}(v_2) + d_{V_4}(v_2)$$

și

$$\text{card}(V_1) + \text{card}(V_2) + \text{card}(V_3) + \text{card}(V_4) = \text{card}(V) = n,$$

deci prin adunarea relațiilor (6.2.9), (6.2.10), (6.2.11), (6.2.12), (6.2.13) rezultă că

$$d(v_1) + d(v_2) \leq 2n - 2$$

și cum nodurile v_1, v_2 sunt distincte și neadiacente se obține o contradicție cu ipoteza teoremei.

Demonstrația prin reducere la absurd este încheiată. \square

Corolarul 6.2.6 (Teorema lui Nash-Williams). Fie $G = (V, E)$ un graf orientat simplu cu $n \geq 2$ noduri. Dacă

$$d^+(x) \geq \frac{n}{2} \text{ și } d^-(x) \geq \frac{n}{2}, \forall x \in V,$$

atunci G este hamiltonian.

Demonstrație. Arătăm că G este tare-conex. Fie $x, y \in V$, $x \neq y$. Avem două cazuri.

Cazul 1. $(x, y) \in E$. Atunci acest arc este un drum de la x la y .

Cazul 2. $(x, y) \notin E$. Fie

$$X = \{z \in V \mid (x, z) \in E\}, Y = \{z \in V \mid (z, y) \in E\}.$$

Avem $X, Y \subseteq V \setminus \{x, y\}$, deci $X \cup Y \subseteq V \setminus \{x, y\}$, prin urmare $\text{card}(X \cup Y) \leq n - 2$.

Graful G fiind simplu, avem

$$\text{card}(X) = d^+(x) \geq \frac{n}{2}, \text{ card}(Y) = d^-(y) \geq \frac{n}{2},$$

deci

$$\text{card}(X \cap Y) = \text{card}(X) + \text{card}(Y) - \text{card}(X \cup Y) \geq \frac{n}{2} + \frac{n}{2} - (n - 2) = 2,$$

prin urmare $X \cap Y \neq \emptyset$. Fie $z \in X \cap Y$. Atunci $(x, z), (z, y) \in E$, deci (x, z, y) este un drum de la x la y .

Conform cazurilor analizate, rezultă că graful G este tare-conex.

Pentru orice $x, y \in V$ avem

$$d(x) + d(y) = d^+(x) + d^-(x) + d^+(y) + d^-(y) \geq 4 \cdot \frac{n}{2} = 2n > 2n - 1,$$

deci aplicând Teorema lui Meyniel rezultă că graful G este hamiltonian. □

Corolarul 6.2.7 (Teorema lui Ghouila-Houri). Fie $G = (V, E)$ un graf orientat simplu, tare-conex, cu $n \geq 2$ noduri. Dacă

$$d(x) \geq n, \forall x \in V,$$

atunci G este hamiltonian.

Demonstrație. Pentru orice $x, y \in V$ avem

$$d(x) + d(y) \geq n + n = 2n > 2n - 1,$$

deci aplicând Teorema lui Meyniel rezultă că graful G este hamiltonian. □

Definiția 6.2.5. Un **graf turneu** (pe scurt **turneu**) este un graf orientat simplu cu proprietatea că între orice două noduri distincte există exact un arc (adică pentru orice noduri $x \neq y$ avem exact unul dintre arcele $(x, y), (y, x)$).

Exemplul 6.2.6. În Figura 6.2.9 este reprezentat un graf turneu cu 6 noduri. El nu este tare-conex (nu există drumuri de la nodul 1 la nodul 2) deci conform Lemei 6.2.1 rezultă că acest graf nu este hamiltonian.

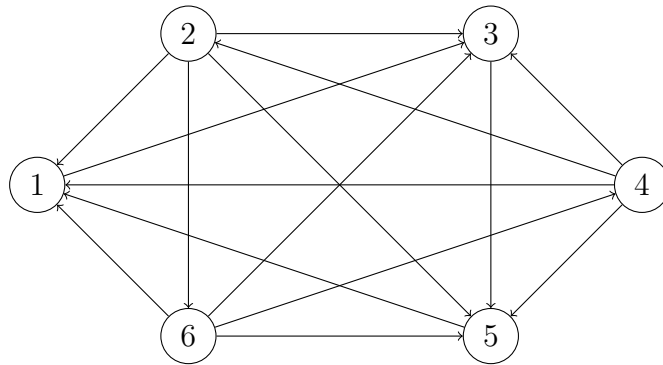


Figura 6.2.9:

Corolarul 6.2.8 (Teorema lui Camion). *Orice graf turneu tare-conex cu $n \geq 2$ noduri este hamiltonian.*

Demonstrație. Din definiția grafului turneu, orice două noduri distincte sunt adiacente, deci sunt verificate ipotezele Teoremei lui Meyniel, prin urmare graful este hamiltonian. \square

Observația 6.2.14. Conform Lemei 6.2.1, reciproca Teoremei lui Camion este adevărată. Astfel un graf turneu cu $n \geq 2$ noduri este hamiltonian dacă și numai dacă este tare-conex.

Corolarul 6.2.9 (Teorema lui Rédei). *Orice graf turneu este trasabil.*

Demonstrație. Fie $G = (V, E)$ un graf turneu cu n noduri. Fie $x, y \notin V$, $x \neq y$ două noduri noi și fie graful orientat $G' = (V', E')$ definit prin

$$V' = V \cup \{x, y\}, \quad E' = E \cup \{(x, y)\} \cup \{(v, x), (y, v) \mid v \in V\}.$$

Evident, G' este un graf turneu cu $n' = n + 2 \geq 3$ noduri. G' este tare-conex, deoarece pentru orice $v_1, v_2 \in V$ el conține drumurile (v_1, x, y, v_2) , (v_1, x) , (v_1, x, y) , (x, y, v_1) , (x, y) , (y, v_1) , (y, v_1, x) . Conform Teoremei lui Camion (corolarul anterior) rezultă că G' este hamiltonian. Fie C un circuit hamiltonian în G' . C conține arcul (x, y) , deoarece acesta este singurul arc din G' incident cu y spre interior, deci în circuitul C nodurile x și y sunt consecutive, în această ordine. Atunci prin eliminarea nodurilor x și y din circuitul C se obține un drum deschis elementar (deoarece circuitul C este elementar) ce conține toate nodurile lui G (deoarece circuitul C conține toate nodurile lui G'), deci este un drum hamiltonian în G și astfel graful G este trasabil. \square

Exemplul 6.2.7. Graful orientat din Figura 6.2.9 este un graf turneu, deci, conform Teoremei lui Rédei, el este trasabil. Un drum hamiltonian în acest graf este $(2, 6, 4, 1, 3, 5)$.

6.3 Colorări în grafuri

Definiția 6.3.1. Fie $G = (V, E)$ un graf neorientat și $k \in \mathbb{N}^*$.

a) O **k -colorare a (nodurilor) grafului G** (o **colorare a (nodurilor) grafului G cu cel mult k culori**) este o funcție $c : V \rightarrow \{1, 2, \dots, k\}$ cu proprietatea că

$$c(x) \neq c(y), \quad \forall x, y \in V \text{ a.î. } x \neq y \text{ și } [x, y] \in E; \quad (6.3.1)$$

pentru orice $x \in V$, $c(x)$ reprezintă **culoarea asociată nodului x** .

b) Graful G se numește **k -colorabil** dacă admite cel puțin o k -colorare.

c) Numărul

$$\chi(G) = \min\{k \in \mathbb{N}^* \mid G \text{ este } k\text{-colorabil}\}$$

se numește **numărul cromatic al grafului** G .

Observația 6.3.1. Condiția (6.3.1) din definiția anterioară exprimă faptul că orice două noduri distincte și adiacente trebuie colorate cu culori diferite.

Observația 6.3.2. Pentru orice graf neorientat $G = (V, E)$ cu n noduri avem

$$\chi(G) \in \{1, 2, \dots, n\}$$

(G este n -colorabil, deoarece putem colora fiecare nod $v_i \in V$ cu culoarea i).

Mai mult, $\chi(G) = 1$ dacă și numai dacă nu există muchii care să nu fie bucle (chiar $E = \emptyset$, dacă G este graf simplu), iar $\chi(G) = n$ dacă și numai dacă $K_n \subseteq G$ (chiar $G = K_n$, dacă G este graf simplu).

Exemplul 6.3.1. Pentru graful G reprezentat în Figura 6.2.7, o 4-colorare este reprezentată în Figura 6.3.1 (cu culorile ROȘU=1, GALBEN=2, ALBASTRU=3, VERDE=4), iar o 3-colorare este reprezentată în Figura 6.3.2.

Acest graf nu are 2-colorări (subgraful generat de submulțimea de noduri $\{1, 2, 3\}$ este complet, deci colorarea nodurilor din această submulțime necesită 3 culori), deci are numărul cromatic $\chi(G) = 3$.

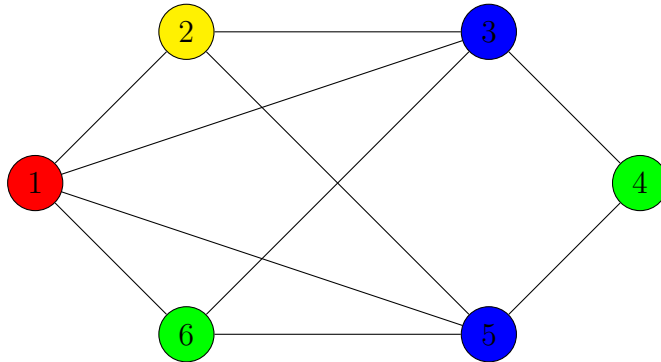


Figura 6.3.1:

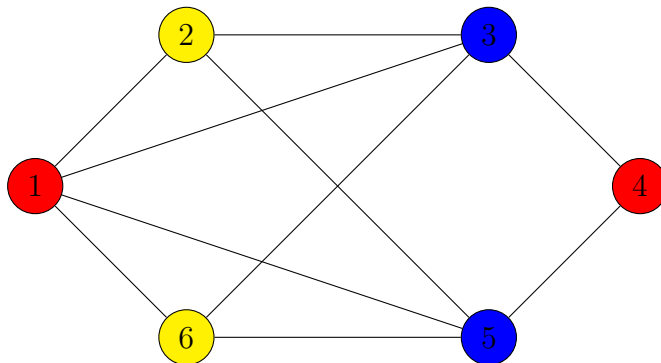


Figura 6.3.2:

Definiția 6.3.2. Fie $G = (V, E)$ un graf neorientat și $k \in \mathbb{N}$.

a) O **k -colorare a muchiilor grafului G (o colorare a muchiilor grafului G cu cel mult k culori)** este o corespondență $c : E \rightarrow \{1, 2, \dots, k\}$ ce asociază fiecărei muchii $e \in E$ o culoare unică $c(e) \in \{1, 2, \dots, k\}$, cu proprietatea că

$$c(e) \neq c(f), \forall e = [x, y], f = [u, v] \in E \text{ a.î. } e \neq f \text{ și } \{x, y\} \cap \{u, v\} \neq \emptyset. \quad (6.3.2)$$

b) Graful G se numește **k -muchie-colorabil** dacă admite cel puțin o k -colorare a muchiilor.

c) Numărul

$$\chi'(G) = \min\{k \in \mathbb{N} \mid G \text{ este } k\text{-muchie-colorabil}\}$$

se numește **numărul muchie-cromatic (indicele cromatic) al grafului G** .

Observația 6.3.3. Condiția (6.3.2) din definiția anterioară exprimă faptul că orice două muchii distincte care au (cel puțin) o extremitate comună trebuie colorate cu culori diferite.

Observația 6.3.4. Pentru orice graf neorientat $G = (V, E)$ cu m muchii avem

$$\chi'(G) \in \{0, 1, \dots, m\}$$

(G este m -muchie-colorabil, deoarece putem colora fiecare muchie $e_i \in E$ cu culoarea i).

Mai mult, $\chi'(G) = 0$ dacă și numai dacă $E = \emptyset$. Dacă G este graf simplu, atunci $\chi'(G) = m$ dacă și numai dacă $m = 1$ sau $G = K_{1,m}$ (graf stea) sau $G = K_3$.

Exemplul 6.3.2. Pentru graful G reprezentat în Figura 6.2.7, o 5-colorare a muchiilor este reprezentată în Figura 6.3.3 (cu culorile ROȘU=1, GALBEN=2, ALBASTRU=3, VERDE=4, GRI=5), iar o 4-colorare a muchiilor este reprezentată în Figura 6.3.4.

Acest graf nu are 3-colorări ale muchiilor (muchiiile $[1, 2]$, $[1, 3]$, $[1, 5]$, $[1, 6]$ au nodul 1 comun, deci colorarea lor necesită 4 culori), deci are numărul muchie-cromatic $\chi'(G) = 4$.

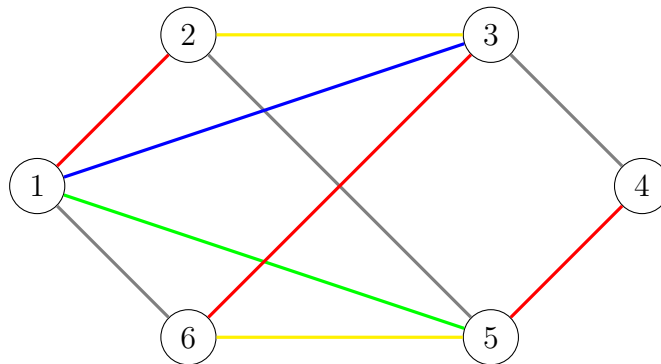


Figura 6.3.3:

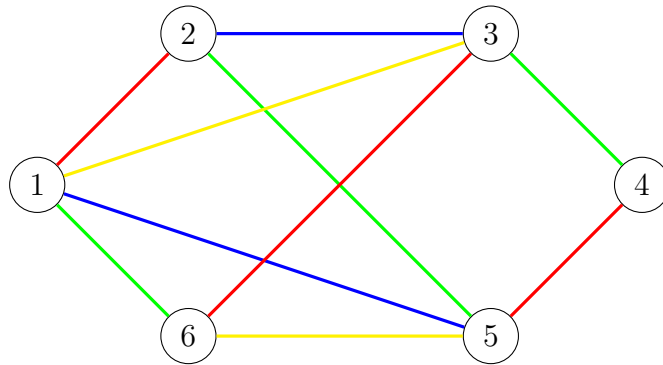


Figura 6.3.4:

6.4 Grafuri bipartite

Definiția 6.4.1. Un graf $G = (V, E)$ se numește **bipartit** dacă există o partiție $V = X \cup Y$ ($X \neq \emptyset$, $Y \neq \emptyset$, $X \cap Y = \emptyset$) a.î. fiecare muchie sau arc al grafului are o extremitate în X și cealaltă extremitate în Y .

Exemplul 6.4.1. Graful reprezentat în Figura 6.2.4 este bipartit, luând partiția nodurilor $V = \{1, 3, 5\} \cup \{2, 4, 6\}$. O altă reprezentare a acestui graf bipartit, cu evidențierea partiției nodurilor prin așezarea în stânga a nodurilor din partea $X = \{1, 3, 5\}$ și în dreapta a nodurilor din partea $Y = \{2, 4, 6\}$, este dată în Figura 6.4.1.

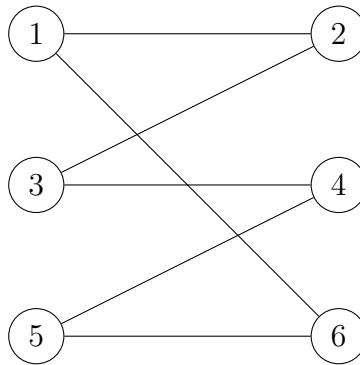


Figura 6.4.1:

Teorema 6.4.1 (de caracterizare a grafurilor bipartite). Un graf cu $n \geq 2$ noduri este bipartit dacă și numai dacă nu conține cicluri de lungime impară.

Demonstrație. " \Rightarrow " Fie $G = (V, E)$ un graf bipartit având partiția nodurilor $V = X \cup Y$ ca în definiția anterioară. Orice ciclu al grafului G are una din formele $[x_1, y_1, x_2, y_2, \dots, x_k, y_k, x_1]$ sau $[y_1, x_1, y_2, x_2, \dots, y_k, x_k, y_1]$ cu $x_1, x_2, \dots, x_k \in X$ și $y_1, y_2, \dots, y_k \in Y$, deci are lungimea $2k$.

" \Leftarrow " Fie $G = (V, E)$ un graf cu $n \geq 2$ noduri ce nu conține cicluri de lungime impară. Demonstrăm că graful G este bipartit în două etape.

Etapă 1) Presupunem că graful G este conex. Fie $v \in V$ un nod arbitrar fixat. Pentru orice nod $x \in V$ definim numărul $d(v, x) \in \mathbb{N}$ prin

$$d(v, x) = \min\{l(\mu) \mid \mu \text{ este lanț de la } v \text{ la } x \text{ în } G\},$$

unde $l(\mu)$ reprezintă lungimea lanțului μ . Definiția este corectă, deoarece graful este conex. Pentru grafuri neorientate $d(v, x)$ se numește **distanța** de la v la x . Fie

$$X = \{x \in V \mid d(v, x) = \text{număr par}\} \text{ și } Y = \{x \in V \mid d(v, x) = \text{număr impar}\}.$$

Evident, $V = X \cup Y$ și $X \cap Y = \emptyset$. Deoarece $d(v, v) = 0$ rezultă că $v \in X$, deci $X \neq \emptyset$. Fie v_1 un nod vecin cu v (există, deoarece graful este conex și are cel puțin două noduri). Atunci $d(v, v_1) = 1$, deci $v_1 \in Y$ și astfel $Y \neq \emptyset$.

Demonstrăm prin reducere la absurd că orice muchie (sau arc) a grafului are o extremitate în X și cealaltă extremitate în Y și astfel va rezulta că graful este bipartit. Fie $[x, y] \in E$ o muchie (sau arc) arbitrar fixată. Fie μ_1 un lanț de lungime minimă de la v la x și μ_2 un lanț de lungime minimă de la v la y , deci $l(\mu_1) = d(v, x)$ și $l(\mu_2) = d(v, y)$. Evident, lanțurile μ_1 și μ_2 sunt elementare. Presupunem prin absurd că $x, y \in X$ sau $x, y \in Y$, adică numerele $d(v, x)$ și $d(v, y)$ au aceeași paritate. Atunci $[x, y]$ nu aparține lanțului μ_1 , deoarece în caz contrar y ar fi penultimul nod al acestui lanț și cum orice sublanț (porțiune) al unui lanț de lungime minimă este un lanț de lungime minimă între extremitățile sale am avea $d(v, y) = d(v, x) - 1$, fals. Analog, $[x, y]$ nu aparține nici lanțului μ_2 . Fie

$$\mu = [v, \dots, x, y, \dots, v]$$

lanțul închis obținut parcurgând întâi lanțul μ_1 de la v la x , apoi muchia $[x, y]$ și apoi lanțul μ_2 de la y la v . Avem

$$l(\mu) = l(\mu_1) + 1 + l(\mu_2) = d(v, x) + 1 + d(v, y) = \text{număr impar}.$$

Deoarece muchia $[x, y]$ nu aparține lanțurilor μ_1 și μ_2 , rezultă că prin eliminarea din lanțul închis μ a eventualelor porțiuni comune L_1, \dots, L_r ale lanțurilor μ_1 și μ_2 rămâne o mulțime nevidă de cicluri (elementare) muchie-disjuncte C_1, \dots, C_s (iar muchia $[x, y]$ aparține unuia din aceste cicluri). Folosind această descompunere, lungimea lanțului închis μ poate fi scrisă sub forma

$$l(\mu) = 2l(L_1) + \dots + 2l(L_r) + l(C_1) + \dots + l(C_s).$$

Cum $l(\mu)$ este un număr impar, rezultă că $l(C_1) + \dots + l(C_s) = \text{număr impar}$, deci cel puțin unul din ciclurile C_1, \dots, C_s are lungimea impară, contradicție cu ipoteza. Demonstrația prin reducere la absurd este încheiată.

Etapă 2) Fie acum G un graf oarecare și $G_1 = (V_1, E_1), \dots, G_k = (V_k, E_k)$ componentele sale conexe. Deoarece G nu conține cicluri de lungime impară, rezultă că și componentele sale conexe G_1, \dots, G_k au această proprietate. Conform etapei 1) rezultă că toate componentele conexe G_i cu cel puțin două noduri sunt grafuri bipartite. Pentru fiecare astfel de componentă conexă, fie $V_i = X_i \cup Y_i$ partiția corespunzătoare grafului bipartit G_i . Dacă există și componente conexe G_i cu un singur nod (adică $\text{card } V_i = 1$ și $E_i = \emptyset$, deoarece G_i nu conține bucle, buclele fiind cicluri de lungime 1), atunci pentru fiecare astfel de componentă conexă definim $X_i = V_i$, $Y_i = \emptyset$ sau $Y_i = V_i$, $X_i = \emptyset$. Fie

$$X = X_1 \cup \dots \cup X_k \text{ și } Y = Y_1 \cup \dots \cup Y_k.$$

Atunci $V = X \cup Y$ este o partiție și fiecare muchie sau arc al grafului G are o extremitate în X și cealaltă extremitate în Y , deci graful G este bipartit. \square

Exemplul 6.4.2. Graful reprezentat în Figura 6.2.3 nu este bipartit, deoarece conține cicluri de lungime impară (de exemplu ciclul $[1, 2, 3, 1]$ de lungime 3).

Corolarul 6.4.1. *Orice graf bipartit conex $G = (V, E)$ are o unică partiție $V = X \cup Y$ ce verifică Definiția 6.4.1.*

Demonstrație. Pentru un nod v arbitrar fixat, submulțimile X și Y definite în etapa 1) a demonstrației teoremei anterioare determină unica partiție cu proprietatea că $v \in X$. Într-adevăr, fie $V = X' \cup Y'$ o partiție arbitrară ce verifică Definiția 6.4.1 și proprietatea $v \in X'$. Pentru orice nod $x \in X$ există $\mu = [v = v_0, v_1, \dots, v_{2k} = x]$ lanț de lungime minimă de la v la x , deci avem, succesiv:

$$v = v_0 \in X', v_1 \in Y', v_2 \in X', \dots, v_{2k} = x \in X'.$$

Analog, pentru orice nod $y \in Y$ există $\mu = [v = w_0, w_1, \dots, w_{2p+1} = y]$ lanț de lungime minimă de la v la y , deci avem, succesiv:

$$v = w_0 \in X', w_1 \in Y', w_2 \in X', \dots, w_{2p+1} = y \in Y'.$$

Rezultă că $X \subseteq X'$ și $Y \subseteq Y'$. Cum $Y = V \setminus X$ și $Y' = V \setminus X'$ rezultă că $X = X'$ și $Y = Y'$. \square

Observația 6.4.1. Demonstrația teoremei anterioare este constructivă, indicând următorul **algoritm de determinare dacă un graf G este bipartit** și, în caz afirmativ, a unei partiții $V = X \cup Y$ a nodurilor sale ce verifică Definiția 6.4.1.

- Se determină componentele conexe G_1, \dots, G_k .
- Pentru fiecare componentă $G_i = (V_i, E_i)$ cu cel puțin două noduri, se parcurg următoarele etape:

- Se fixează un nod $v_i \in V_i$.
- Se calculează distanțele $d(v_i, x)$ pentru toate nodurile $x \in V_i$. De exemplu,

$$d(v_i, x) = \text{lungimea lanțului elementar unic de la } v_i \text{ la } x \text{ în arborele } BF(v_i)$$

(corespunzător parcurgerii BF a grafului neorientat G pornind din nodul v_i ; dacă graful G este orientat se parcurge graful neorientat obținut prin eliminarea orientării arcelor!).

- Se calculează mulțimile

$$X_i = \{x \in V_i \mid d(v_i, x) = \text{număr par}\} \text{ și } Y_i = \{x \in V_i \mid d(v_i, x) = \text{număr impar}\}.$$

- Dacă există o muchie (sau arc) de forma $[x, y] \in E_i$ a.î. $x, y \in X_i$ sau $x, y \in Y_i$, atunci componenta G_i nu este graf bipartit, deci nici graful G nu este bipartit și algoritmul se încheie.
- În caz contrar componenta G_i este graf bipartit și $V_i = X_i \cup Y_i$ este partiția corespunzătoare acestui graf bipartit.
- Pentru fiecare componentă $G_i = (V_i, E_i)$ cu câte un singur nod, adică $V_i = \{v_i\}$, se parcurg următoarele etape:
 - Dacă $[v_i, v_i] \in E_i$ (bucă), atunci graful G nu este bipartit și algoritmul se încheie.
 - În caz contrar se calculează mulțimile $X_i = V_i$, $Y_i = \emptyset$ pentru prima componentă G_i și $Y_i = V_i$, $X_i = \emptyset$ pentru următoarele componente G_i (se asigură astfel că $X \neq \emptyset$ și $Y \neq \emptyset$).
- Toate componentele conexe cu cel puțin două noduri sunt grafuri bipartite, iar toate componentele conexe cu câte un singur nod nu conțin bucle. Atunci graful G este bipartit și o partiție corespunzătoare este $V = X \cup Y$, cu $X = X_1 \cup \dots \cup X_k$ și $Y = Y_1 \cup \dots \cup Y_k$.

Exemplul 6.4.3. Fie graful reprezentat în Figura 6.2.2. Aplicăm algoritmul din observația anterioară. Fixând nodul $v = 1$ (pentru singura componentă conexă), obținem distanțele

$$d(1, 1) = 0, d(1, 2) = d(1, 5) = 1, d(1, 3) = d(1, 6) = 2, d(1, 4) = d(1, 7) = 3, d(1, 8) = 4,$$

deci $X = \{1, 3, 6, 8\}$ și $Y = \{2, 4, 5, 7\}$. Nu există nicio muchie $[x, y]$ a.î. $x, y \in X$ sau $x, y \in Y$, deci graful dat este bipartit și unica partiție corespunzătoare (ce verifică Definiția 6.4.1) este $V = X \cup Y$. O reprezentare a acestui graf bipartit, cu evidențierea partiției nodurilor, este dată în Figura 6.4.2.

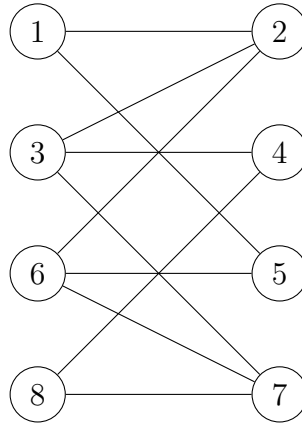


Figura 6.4.2:

Propoziția 6.4.1. Fie $G = (V, E)$ un graf neorientat bipartit și $V = X \cup Y$ o partiție ce verifică Definiția 6.4.1. Dacă G este hamiltonian, atunci

$$\text{card}(X) = \text{card}(Y).$$

Demonstrație. Fie $C = [x_1, x_2, \dots, x_n, x_1]$ un ciclu hamiltonian în graful G , deci $V = \{x_1, x_2, \dots, x_n\}$.

Conform Teoremei 6.4.1 rezultă că n este par. Fie $n = 2k$. Avem două cazuri.

Cazul 1) $x_1 \in X$. Deoarece graful este bipartit, avem, succesiv, $x_2 \in Y$, $x_3 \in X$, \dots , $x_{2k-1} \in X$, $x_{2k} = x_n \in Y$. Rezultă că

$$X = \{x_1, x_3, \dots, x_{2k-1}\}, Y = \{x_2, x_4, \dots, x_{2k}\},$$

deci $\text{card}(X) = \text{card}(Y) = k$.

Cazul 2) $x_1 \in Y$. Deoarece graful este bipartit, avem, succesiv, $x_2 \in X$, $x_3 \in Y$, \dots , $x_{2k-1} \in Y$, $x_{2k} = x_n \in X$, deci

$$X = \{x_2, x_4, \dots, x_{2k}\}, Y = \{x_1, x_3, \dots, x_{2k-1}\},$$

deci din nou $\text{card}(X) = \text{card}(Y) = k$. □

Propoziția 6.4.2. Un graf neorientat cu $n \geq 2$ noduri este bipartit dacă și numai dacă este 2-colorabil.

Demonstrație. " \Rightarrow " Fie $G = (V, E)$ un graf neorientat bipartit având partiția nodurilor $V = X \cup Y$ ca în Definiția 6.4.1. Fie funcția

$$c : V \rightarrow \{1, 2\}, c(x) = \begin{cases} 1, & \text{dacă } x \in X, \\ 2, & \text{dacă } x \in Y. \end{cases}$$

Pentru orice muchie $[x, y] \in E$, deoarece graful G este bipartit, avem două posibilități:

1) $x \in X$, $y \in Y$, deci $c(x) = 1 \neq 2 = c(y)$;

2) $x \in Y$, $y \in X$, deci $c(x) = 2 \neq 1 = c(y)$.

Rezultă că funcția c este o 2-colorare a lui G .

" \Leftarrow " Fie $G = (V, E)$ un graf neorientat 2-colorabil, cu $n \geq 2$ noduri.

Dacă $E = \emptyset$, atunci este evident că graful este bipartit (luând, de exemplu, $X = \{v\}$ și $Y = V \setminus \{v\}$, cu $v \in V$).

Considerăm în continuare că $E \neq \emptyset$. Fie $c : V \rightarrow \{1, 2\}$ o 2-colorare a lui G . Fie

$$X = \{x \in V \mid c(x) = 1\}, Y = \{x \in V \mid c(x) = 2\}.$$

Evident, $V = X \cup Y$ și $X \cap Y = \emptyset$.

Pentru orice muchie $[x, y] \in E$, deoarece c este o 2-colorare a lui G , avem două posibilități:

1) $c(x) = 1$, $c(y) = 2$, deci $x \in X$, $y \in Y$;

2) $c(x) = 2$, $c(y) = 1$, deci $x \in Y$, $y \in X$.

Rezultă că fiecare muchie a grafului G are o extremitate în X și cealaltă extremitate în Y . Graful G conține cel puțin o muchie, deci $X \neq \emptyset$ și $Y \neq \emptyset$. Rezultă că graful G este bipartit. \square

Observația 6.4.2. Conform demonstrației anterioare, o 2-colorare a unui graf neorientat bipartit având partiția mulțimii nodurilor $V = X \cup Y$ (ce verifică Definiția 6.4.1) se obține colorând nodurile din X cu o culoare și nodurile din Y cu cealaltă culoare.

Mai mult, dacă graful bipartit este și conex, atunci conform Corolarului 6.4.1, partiția $V = X \cup Y$ ce verifică Definiția 6.4.1 este unică, deci graful are exact două 2-colorări și anume colorăm nodurile din X cu culoarea 1 și nodurile din Y cu culoarea 2, sau invers (colorăm nodurile din X cu culoarea 2 și nodurile din Y cu culoarea 1).

Corolarul 6.4.2. Fie $G = (V, E)$ un graf neorientat cu $n \geq 2$ noduri și $E \neq \emptyset$. Atunci G este bipartit dacă și numai dacă $\chi(G) = 2$.

Demonstrație. " \Rightarrow " Presupunem că G este bipartit. Conform propoziției anterioare, rezultă că G este 2-colorabil. Cum G are cel puțin o muchie, iar aceasta nu este o buclă (din definiția grafului bipartit, un astfel de graf nu conține bucle), rezultă că G nu este 1-colorabil. Deci avem $\chi(G) = 2$.

" \Leftarrow " Presupunem că $\chi(G) = 2$. Rezultă că G este 2-colorabil. Conform propoziției anterioare rezultă că G este bipartit. \square

Exemplul 6.4.4. Considerăm graful bipartit complet $K_{3,3}$, reprezentat în Figura 6.4.3. Fiind conex, unica partiție a nodurilor sale ce verifică Definiția 6.4.1 este $V = \{1, 2, 3\} \cup \{4, 5, 6\}$.

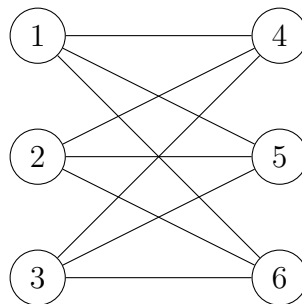


Figura 6.4.3:

Conform corolarului anterior, numărul său cromatic este $\chi(K_{3,3}) = 2$.

Conform observației anterioare, el are exact două 2-colorări, evidențiate în Figurile 6.4.4 și 6.4.5.

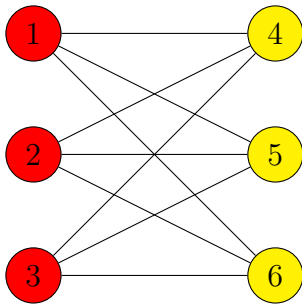


Figura 6.4.4:

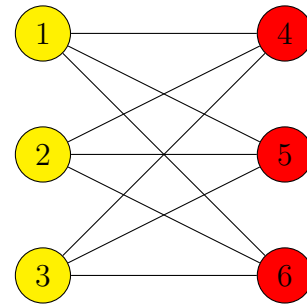


Figura 6.4.5:

Teorema 6.4.2 (König). Fie $G = (V, E)$ un graf neorientat bipartit și

$$\Delta(G) = \max_{x \in V} d(x)$$

gradul maxim al nodurilor din G . Atunci numărul muchie-cromatic al grafului G verifică egalitatea

$$\chi'(G) = \Delta(G).$$

Demonstrație. Fie $k = \Delta(G)$. Din definiția lui $\Delta(G)$ rezultă că există un nod $x \in V$ astfel încât $d(x) = \Delta(G) = k$. Atunci nodul x este extremitate pentru k muchii din E , deci pentru colorarea acestor k muchii trebuie folosite k culori diferite, și astfel rezultă că

$$\chi'(G) \geq k. \quad (6.4.1)$$

Fie $E = \{e_1, e_2, \dots, e_m\}$, $m \in \mathbb{N}$ (unde $E = \emptyset$ pentru $m = 0$).

Demonstrăm prin inducție după $i \in \{0, 1, \dots, m\}$ că graful parțial

$$G_i = (V, \{e_1, e_2, \dots, e_i\}) \text{ este } k\text{-muchie-colorabil} \quad (6.4.2)$$

(unde $\{e_1, e_2, \dots, e_i\} = \emptyset$ pentru $i = 0$).

Pentru $i = 0$ avem $G_i = (V, \emptyset)$, deci G_i este k -muchie-colorabil (fiind chiar 0-muchie-colorabil!).

Presupunem acum că relația (6.4.2) este adevărată pentru $i - 1$ și o demonstrăm pentru i ($i \in \{1, 2, \dots, m\}$). Fie $c : \{e_1, e_2, \dots, e_{i-1}\} \rightarrow \{1, \dots, k\}$ o k -colorare a muchiilor grafului $G_{i-1} = (V, \{e_1, e_2, \dots, e_{i-1}\})$ și fie $e_i = [x_i, y_i]$. Muchia $e_i = [x_i, y_i]$ trebuie colorată cu o culoare diferită de culorile muchiilor din G_{i-1} incidente cu x_i sau cu y_i . Fie

$$F_1 = \{e_j \mid j \in \{1, \dots, i-1\}, e_j \text{ incidentă cu } x_i\}, F_2 = \{e_j \mid j \in \{1, \dots, i-1\}, e_j \text{ incidentă cu } y_i\}.$$

Avem două cazuri.

Cazul 1) $\{1, \dots, k\} \setminus \{c(e_j) \mid e_j \in F_1 \cup F_2\} \neq \emptyset$, adică există (cel puțin) o culoare $p \in \{1, \dots, k\}$ astfel încât $p \neq c(e_j)$ pentru orice muchie $e_j \in F_1 \cup F_2$ (adică muchie din G_{i-1} incidentă cu x_i sau cu y_i). Atunci luând

$$c(e_i) = p$$

(adică se colorează muchia e_i cu culoarea disponibilă, p) obținem că funcția $c : \{e_1, e_2, \dots, e_i\} \rightarrow \{1, \dots, k\}$ este o k -colorare a muchiilor grafului G_i .

Cazul 2) $\{c(e_j) \mid e_j \in F_1 \cup F_2\} = \{1, \dots, k\}$, adică nu există nicio culoare diferită de culorile muchiilor din G_{i-1} incidente cu x_i sau cu y_i . Cum $e_i = [x_i, y_i] \notin F_1 \cup F_2$, rezultă că avem

$$\text{card}(F_1) < d_G(x_i) \leq k, \text{ card}(F_2) < d_G(y_i) \leq k,$$

deci există (cel puțin) o culoare $p_1 \in \{1, \dots, k\}$ astfel încât $p_1 \neq c(e_j)$ pentru orice muchie $e_j \in F_1$ (adică muchie din G_{i-1} incidentă cu x_i) și (cel puțin) o culoare $p_2 \in \{1, \dots, k\}$ astfel încât $p_2 \neq c(e_j)$ pentru orice muchie $e_j \in F_2$ (adică muchie din G_{i-1} incidentă cu y_i). Conform ipotezei acestui caz, $p_1 \neq p_2$. Fie H graful parțial al lui G_{i-1} indus de muchiile colorate cu culorile p_1 sau p_2 . Evident, pentru orice nod v avem $d_H(v) \leq 2$ (printre muchiile lui G_{i-1} incidente cu v există cel mult câte o muchie colorată cu culoarea p_1 , respectiv p_2). Astfel componenta conexă a lui H ce conține nodul y_i este un lanț elementar de forma

$$\mu = [y_1 = z_1, z_2, \dots, z_s]$$

ce are muchiile colorate alternativ cu culorile p_1 și p_2 (adică $c([y_1, z_2]) = p_1$, deoarece nu există muchii colorate cu p_2 incidente cu y_i , apoi $c([z_2, z_3]) = p_2$, $c([z_3, z_4]) = p_1, \dots$).

Dacă am avea că $x_i \in V(\mu)$, atunci sublanțul μ' al lui μ cuprins între y_i și x_i ar începe cu muchia $[y_1, z_2]$ colorată cu culoarea p_1 și s-ar încheia cu o muchie $[z_t, x_i]$ colorată cu culoarea p_2 (deoarece nu există muchii colorate cu p_1 incidente cu x_i). Din alternanța culorilor p_1 și p_2 de-a lungul sublanțului μ' , ar rezulta că μ' are un număr par de muchii, și atunci ciclul $C = [y_1 = z_1, z_2, \dots, z_t, x_i, y_i]$ (obținut prin adăugarea muchiei $e_i = [x_i, y_i]$ la sublanțul μ') ar fi un ciclu de lungime impară în graful bipartit G , contradicție cu Teorema 6.4.1. Deci $x_i \notin V(\mu)$.

Interschimbând culorile p_1 și p_2 de-a lungul lanțului μ și colorând apoi muchia $e_i = [x_i, y_i]$ cu culoarea p_1 (posibil, deoarece după interschimbare nicio muchie din G_{i-1} incidentă cu x_i sau cu y_i nu mai este colorată cu p_1), adică luând colorarea $c' : \{e_1, e_2, \dots, e_i\} \rightarrow \{1, \dots, k\}$ definită prin

$$c'(e) = \begin{cases} p_1, & \text{dacă } e \in E(\mu), c(e) = p_2, \\ p_2, & \text{dacă } e \in E(\mu), c(e) = p_1, \\ c(e), & \text{dacă } e \in \{e_1, \dots, e_{i-1}\} \setminus E(\mu), \\ p_1, & \text{dacă } e = e_i, \end{cases}$$

obținem o k -colorare, c' , a muchiilor grafului G_i . Demonstrația prin inducție a relației (6.4.2) este încheiată.

Aplicând această relație pentru $i = m$ rezultă că graful $G_m = (V, \{e_1, e_2, \dots, e_m\}) = G$ este k -muchie-colorabil, deci

$$\chi'(G) \leq k. \quad (6.4.3)$$

Din relațiile (6.4.1) și (6.4.3) rezultă că $\chi'(G) = k$, deci că $\chi'(G) = \Delta(G)$. \square

Observația 6.4.3. Demonstrația teoremei anterioare este constructivă, indicând următorul **algoritm de determinare a numărului muchie-cromatic și a unei colorări a muchiilor cu număr minim de culori** pentru graful bipartit $G = (V, E)$:

- Se determină numărul muchie-cromatic $\chi'(G) = k$, egal cu gradul maxim al nodurilor, $k = \Delta(G) = \max_{x \in V} d(x)$; mulțimea culorilor va fi $\{1, 2, \dots, k\}$.
- Pentru fiecare muchie $e = [x, y]$ a grafului, se actualizează colorarea curentă astfel:
 - Se verifică dacă există o culoare disponibilă pentru muchia e , adică o culoare p nefolosită de nicio muchie incidentă cu nodul x sau cu nodul y .
 - Dacă DA, se colorează muchia e cu culoarea disponibilă p .
 - În caz contrar, se parcurg următoarele subetape:
 - * Se determină o culoare p_1 nefolosită de nicio muchie incidentă cu nodul x .
 - * Se determină o culoare p_2 nefolosită de nicio muchie incidentă cu nodul y .

- * Pornind din nodul y și cu culoarea p_1 , se determină lanțul (maximal) format cu muchii colorate alternativ cu culorile p_1 și p_2 și, concomitent, se interschimbă culorile p_1 și p_2 pentru aceste muchii.
- * Se colorează muchia e cu culoarea p_1 (care a devenit disponibilă).

Exemplul 6.4.5. Fie graful bipartit complet $K_{3,3}$, reprezentat în Figura 6.4.3. Toate nodurile au gradul 3, deci, conform teoremei anterioare, numărul său muchie-cromatic este

$$\chi'(K_{3,3}) = \Delta(K_{3,3}) = 3.$$

Aplicând algoritmul din observația anterioară, obținem colorarea muchiilor cu 3 culori (ROȘU=1, GALBEN=2, ALBASTRU=3), evidențiată pas cu pas în Figurile 6.4.6-6.4.14.

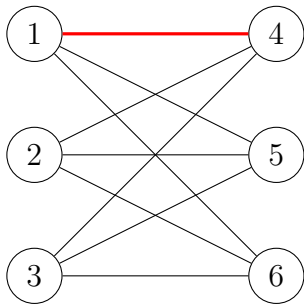


Figura 6.4.6:

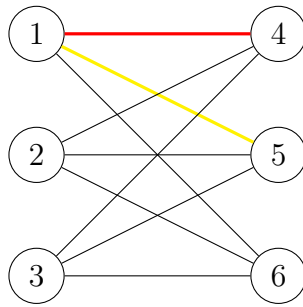


Figura 6.4.7:

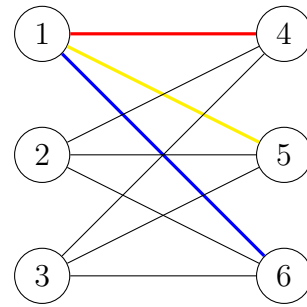


Figura 6.4.8:

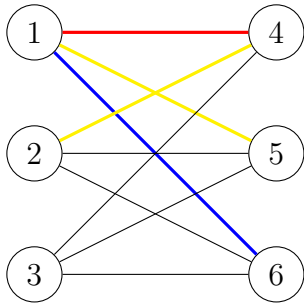


Figura 6.4.9:

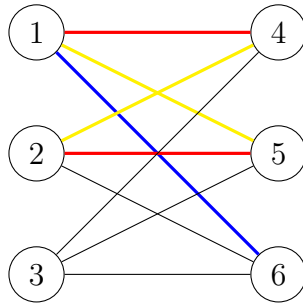


Figura 6.4.10:

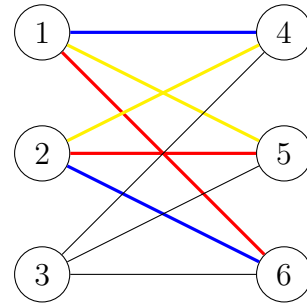


Figura 6.4.11:

Remarcăm că la colorarea muchiei $[2, 6]$ (pornind de la colorarea din Figura 6.4.10), neavând o culoare disponibilă, s-a procedat astfel:

- s-a determinat prima culoare nefolosită de nicio muchie incidentă cu nodul 2, aceasta a fost $p_1 = 3$ (ALBASTRU);
- s-a determinat prima culoare nefolosită de nicio muchie incidentă cu nodul 6, aceasta a fost $p_2 = 1$ (ROȘU);
- pornind din nodul 6 și cu culoarea $p_1 = 3$ (ALBASTRU), s-a determinat lanțul (maximal) format cu muchiile $[6, 1]$, $[1, 4]$ colorate alternativ cu culorile $p_1 = 3$ (ALBASTRU) și $p_2 = 1$ (ROȘU) și s-au interschimbă culorile $p_1 = 3$ (ALBASTRU) și $p_2 = 1$ (ROȘU) pentru aceste muchii;

- s-a colorat muchia $[2, 6]$ cu culoarea $p_1 = 3$ (ALBASTRU), care a devenit disponibilă, obținând colorarea din Figura 6.4.11.

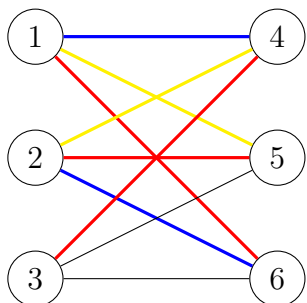


Figura 6.4.12:

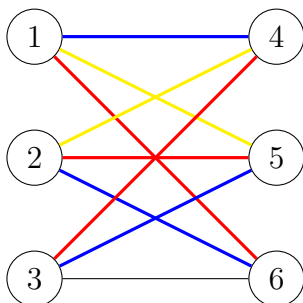


Figura 6.4.13:

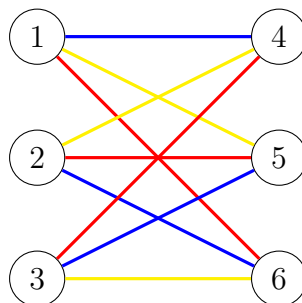


Figura 6.4.14:

Tema 7

Distanțe și drumuri minime

Problema determinării distanțelor și drumurilor minime între nodurile unui graf ponderat apare în numeroase aplicații practice. În continuare vom prezenta doi algoritmi clasici pentru rezolvarea acestei probleme.

7.1 Expunerea problemei

Definiția 7.1.1. Fie (G, c) un graf ponderat, unde $G = (V, E)$, $V = \{v_1, \dots, v_n\}$ iar $c : E \rightarrow \mathbb{R}_+$.

- a) Dacă $\mu = (x_0, e_1, x_1, \dots, x_{k-1}, e_k, x_k)$ este un drum al grafului G , unde $x_0, x_1, \dots, x_k \in V$, $e_1, \dots, e_k \in E$, $k \in \mathbb{N}$, atunci **costul** (**ponderea**) lui μ este

$$c(\mu) = \begin{cases} 0, & \text{dacă } k = 0, \\ \sum_{i=1}^k c(e_i), & \text{dacă } k \geq 1 \end{cases}$$

(adică suma costurilor arcelor sau muchiilor sale).

- b) Fie $x, y \in V$. Un drum $\mu^* = (x, \dots, y)$ în graful G cu proprietatea că

$$c(\mu^*) = \min\{c(\mu) \mid \mu \text{ este drum de la } x \text{ la } y \text{ în } G\}$$

se numește **drum minim** (**drum de cost minim**, **drum de pondere minimă**) de la x la y în graful ponderat (G, c) . Costul $c(\mu^*)$ al acestui drum minim se numește **distanța minimă** de la x la y în graful (G, c) .

Observația 7.1.1. Eliminând eventualele circuite C_1, \dots, C_p dintr-un drum μ de la nodul x la nodul y obținem un drum elementar μ' de la x la y cu proprietatea că $c(\mu') = c(\mu) - \sum_{k=1}^p c(C_k) \leq c(\mu)$.

Dacă drumul μ este minim atunci și drumul elementar μ' este minim și $c(e) = 0$ pentru orice muchie sau arc e al circuitelor C_1, \dots, C_p . Deci existența unui drum minim de la x la y implică existența unui drum minim elementar de la x la y . Astfel distanța minimă de la x la y poate fi considerată ca fiind costul minim al unui drum elementar de la x la y . Mai mult, dacă funcția cost c este strict pozitivă, atunci orice drum minim este elementar.

Observația 7.1.2. Mulțimea drumurilor elementare fiind evident finită, avem echivalența: există drumuri minime de la x la y dacă și numai dacă există drumuri de la x la y .

Observația 7.1.3. Distanța minimă de la un nod x la el însuși este egală cu zero, drumul minim elementar de la x la x fiind drumul de lungime zero, $\mu = (x)$.

Observația 7.1.4. Dacă $\mu = (x_0, x_1, \dots, x_k)$ este un drum minim de la x_0 la x_k , atunci orice subdrum $\mu' = (x_i, x_{i+1}, \dots, x_j)$ ($0 \leq i \leq j \leq k$) al său este un drum minim de la x_i la x_j (**principiul optimalității al lui Bellman**). Afirmarea poate fi justificată ușor prin reducere la absurd.

Exemplul 7.1.1. Fie graful ponderat (G, c) reprezentat în Figura 7.1.1.

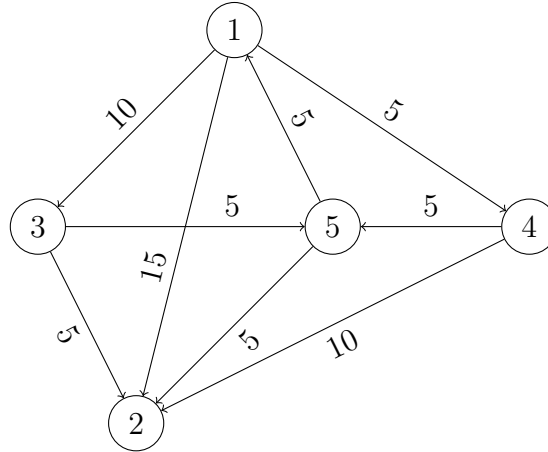


Figura 7.1.1:

Drumurile elementare de la nodul 1 la nodul 5 sunt $\mu_1 = (1, 3, 5)$, având costul $c(\mu_1) = 10 + 5 = 15$, $\mu_2 = (1, 4, 5)$, având costul $c(\mu_2) = 5 + 5 = 10$, deci μ_2 este un drum minim de la 1 la 5. Astfel distanța minimă de la nodul 1 la nodul 5 este $c(\mu_2) = 10$.

Definiția 7.1.2. Fie (G, c) un graf ponderat, unde $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $c : E \rightarrow \mathbb{R}_+$.

- a) **Matricea distanțelor (costurilor) directe** asociată grafului (G, c) este matricea $C = (c_{ij})_{i,j=\overline{1,n}}$ definită prin

$$c_{ij} = \begin{cases} 0, & \text{dacă } i = j, \\ \min\{c(e) | e = (v_i, v_j) \in E\}, & \text{dacă } i \neq j \text{ și } \exists (v_i, v_j) \in E, \\ \infty, & \text{dacă } i \neq j \text{ și } \nexists (v_i, v_j) \in E \end{cases}$$

(pentru grafuri neorientate (v_i, v_j) desemnând de fapt muchia $[v_i, v_j]$).

- b) **Matricea distanțelor (costurilor) minime** asociată grafului (G, c) este matricea $C^* = (c_{ij}^*)_{i,j=\overline{1,n}}$ definită prin

$$c_{ij}^* = \begin{cases} c(\mu^*), & \mu^* = \text{drum minim de la } v_i \text{ la } v_j, \\ & \text{dacă } \exists \mu = (v_i, \dots, v_j) \text{ drum în } G, \\ \infty, & \text{în caz contrar.} \end{cases}$$

Observația 7.1.5. Evident, pentru orice graf neorientat atât matricea distanțelor directe cât și matricea distanțelor minime sunt matrice simetrice.

Observația 7.1.6. Conform Observației 7.1.1, putem să înlocuim termenul de "drum" cu cel de "drum elementar" în definiția anterioară.

Conform Observației 7.1.2, punctul b) din definiția anterioară este o extindere a definiției distanței minime de la punctul b) al Definiției 7.1.1.

Conform Observației 7.1.3, $c_{ii}^* = 0 \forall i \in \{1, \dots, n\}$.

Exemplul 7.1.2. Matricele distanțelor directe, respectiv minime asociate grafului din Exemplul 7.1.1 sunt

$$C = \begin{pmatrix} 0 & 15 & 10 & 5 & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 5 & 0 & \infty & 5 \\ \infty & 10 & \infty & 0 & 5 \\ 5 & 5 & \infty & \infty & 0 \end{pmatrix}, \quad C^* = \begin{pmatrix} 0 & 15 & 10 & 5 & 10 \\ \infty & 0 & \infty & \infty & \infty \\ 10 & 5 & 0 & 15 & 5 \\ 10 & 10 & 20 & 0 & 5 \\ 5 & 5 & 15 & 10 & 0 \end{pmatrix}.$$

7.2 Algoritmul Dijkstra

Vom expune un algoritm pentru determinarea distanțelor minime și a drumurilor minime de la un nod fixat, numit și nod sursă, la toate nodurile grafului ponderat dat.

Algoritmul 7.2.1 (Dijkstra). Fie (G, c) un graf ponderat, $G = (V, E)$, $V = \{v_1, v_2, \dots, v_n\}$, $c : E \rightarrow \mathbb{R}_+$. Fie $C = (c_{ij})_{i,j=1,n}$ matricea distanțelor directe asociată grafului (G, c) și fie $v_s \in V$ un nod arbitrar fixat, numit **nod sursă**. Distanțele minime de la nodul v_s la nodurile grafului sunt calculate și memorate într-un vector $t = (t_1, \dots, t_n)$ astfel:

- La pasul 1 se selectează nodul sursă v_s și se ia $t_s = 0$;
- La pasul k , $2 \leq k \leq n$, se cunosc nodurile $v_{i_1}, \dots, v_{i_{k-1}}$ selectate la pașii anteriori și distanțele corespundente $t_{i_1}, \dots, t_{i_{k-1}}$.
 - a) Dacă nu mai există nicio muchie sau arc de la un nod selectat $v_j \in \{v_{i_1}, \dots, v_{i_{k-1}}\}$ la un nod neselectat $v_i \in V \setminus \{v_{i_1}, \dots, v_{i_{k-1}}\}$, atunci se ia $t_i = \infty$ pentru orice nod neselectat $v_i \in V \setminus \{v_{i_1}, \dots, v_{i_{k-1}}\}$ și algoritmul se încheie.
 - b) În caz contrar se selectează un nod $v_{i_k} \in V \setminus \{v_{i_1}, \dots, v_{i_{k-1}}\}$ cu proprietatea că există un nod selectat $v_{j_k} \in \{v_{i_1}, \dots, v_{i_{k-1}}\}$ astfel încât

$$t_{j_k} + c_{j_k i_k} = \min\{t_j + c_{ji} \mid v_j \in \{v_{i_1}, \dots, v_{i_{k-1}}\}, v_i \in V \setminus \{v_{i_1}, \dots, v_{i_{k-1}}\}\}. \quad (7.2.1)$$

Se ia

$$t_{i_k} = t_{j_k} + c_{j_k i_k} \quad (7.2.2)$$

și se trece la pasul $k + 1$.

Observația 7.2.1. Evident, algoritmul execută cel mult n pași.

Teorema 7.2.1 (de corectitudine a Algoritmului Dijkstra). *În contextul Algoritmului Dijkstra, avem*

$$t_i = c_{si}^*, \quad \forall i \in \{1, \dots, n\}$$

(adică distanța t_i calculată de algoritm este chiar distanța minimă de la v_s la v_i).

Demonstrație. Vom demonstra prin inducție după k că nodul v_{i_k} selectat la pasul k și distanța corespundătoare t_{i_k} calculată la acel pas verifică egalitatea din enunț, adică

$$t_{i_k} = c_{si_k}^*,$$

și, în plus, $t_{i_k} < \infty$.

Pentru $k = 1$ afirmația este evidentă deoarece

$$v_{i_1} = v_s, \quad t_{i_1} = 0 = c_{ss}^*.$$

Presupunem adevărată afirmația pentru orice pas mai mic decât k și o demonstrăm pentru pasul k . Fie $v_{j_k} \in \{v_{i_1}, \dots, v_{i_{k-1}}\}$ un nod ce verifică egalitatea (7.2.1). Din descrierea algoritmului și din ipoteza de inducție (nodul v_{j_k} fiind selectat la un pas anterior), rezultă că $t_{j_k} = c_{sj_k}^* < \infty$ și $c_{j_k i_k} < \infty$, deci $t_{i_k} < \infty$ (conform (7.2.2)).

Dacă (v_s, \dots, v_{j_k}) este un drum minim de la v_s la v_{j_k} (există, deoarece $c_{sj_k}^* < \infty$), atunci $\mu = (v_s, \dots, v_{j_k}, v_{i_k})$ este un drum de la v_s la v_{i_k} , având costul $c(\mu) = c_{sj_k}^* + c_{j_k i_k} = t_{j_k} + c_{j_k i_k} = t_{i_k}$ (conform (7.2.2)), deci

$$c_{si_k}^* \leq t_{i_k} < \infty. \quad (7.2.3)$$

Rezultă că există drumuri minime de la v_s la v_{i_k} . Fie

$$\mu^* = (v_s = x_0, x_1, \dots, x_{p-1}, x_p = v_{i_k})$$

un drum elementar minim de la v_s la v_{i_k} . Fie $l \in \{0, 1, \dots, p-1\}$ indicele maxim astfel încât $x_l \in \{v_{i_1}, \dots, v_{i_{k-1}}\}$ (există, deoarece $x_0 = v_s = v_{i_1}$).

Fie $x_l = v_{i_r}$, $1 \leq r \leq k-1$, și fie $x_{l+1} = v_{i_q}$, $q \geq k$.

Din descrierea pasului q al algoritmului, conform relațiilor (7.2.1) și (7.2.2) rezultă că

$$t_{i_q} \leq t_{i_r} + c_{i_r i_q},$$

iar conform ipotezei de inducție pentru r avem

$$t_{i_r} = c_{si_r}^* < \infty.$$

Deci

$$t_{i_q} \leq c_{si_r}^* + c_{i_r i_q} = c_{si_q}^* \quad (7.2.4)$$

(conform principiului optimalității al lui Bellman pentru subdrumurile dintre $x_0 = v_s$ și $x_l = v_{i_r}$, respectiv dintre $x_0 = v_s$ și $x_{l+1} = v_{i_q}$ ale drumului minim μ^*).

Cazul 1) Dacă $l = p-1$, atunci $x_{l+1} = v_{i_k}$, deci $q = k$ și din (7.2.3) și (7.2.4) rezultă că $t_{i_k} = c_{si_k}^* < \infty$.

Cazul 2) Dacă $l < p-1$, deci $q \neq k$, demonstrăm că

$$t_{i_k} = c_{si_k}^*$$

prin reducere la absurd. Într-adevăr, în caz contrar conform (7.2.3) am avea $c_{si_k}^* < t_{i_k}$.

Cum $q > k$, din descrierea algoritmului avem $t_{i_q} \geq t_{i_k}$ (deoarece valorile t_{i_k} sunt monoton crescătoare de la un pas la altul, inducție!). Utilizând (7.2.4) am obține

$$c_{si_k}^* < t_{i_k} \leq t_{i_q} \leq c_{si_q}^*,$$

ceea ce contrazice faptul că are loc inegalitatea $c_{si_q}^* \leq c_{si_k}^*$ (conform principiului optimalității al lui Bellman pentru subdrumul dintre v_s și v_{i_q} al drumului minim μ^*).

Demonstrația prin inducție este încheiată.

Evident, pentru orice nod v_i rămas neselectat în urma executării ultimului pas al algoritmului avem $t_i = \infty = c_{si}^*$.

Într-adevăr, dacă ar exista un drum elementar minim

$$\mu = (v_s = y_0, y_1, \dots, y_p = v_i),$$

luând $l \in \{0, 1, \dots, p-1\}$ indicele maxim pentru care y_l este selectat (există, deoarece $y_0 = v_s$ este selectat) atunci y_{l+1} ar fi neselectat deși există o muchie sau un arc de la y_l la y_{l+1} , contradicție cu descrierea modului de încheiere a algoritmului. \square

Exemplul 7.2.1. Pentru graful ponderat din Exemplul 7.1.1, luând ca nod sursă nodul 1, aplicarea Algoritmului Dijkstra este evidențiată în următorul tabel:

Pas	Nodul selectat	Distanța minimă
1	1	0
2	4	5
3	3	10
4	5	10
5	2	15

De exemplu, la pasul 3 avem deja selectate nodurile $i_1 = 1$ și $i_2 = 4$, cu distanțele minime $t_1 = c_{11}^* = 0$ și $t_4 = c_{14}^* = 5$. Se selectează nodul $i_3 = 3$, cu distanța minimă $t_3 = 10$, deoarece

$$\begin{aligned} & \min\{t_1 + c_{12}, t_1 + c_{13}, t_1 + c_{15}, t_4 + c_{42}, t_4 + c_{43}, t_4 + c_{45}\} \\ &= \min\{0 + 15, 0 + 10, 0 + \infty, 5 + 10, 5 + \infty, 5 + 5\} = 10 = t_1 + c_{13}. \end{aligned}$$

Observația 7.2.2. Algoritmul Dijkstra este specific **metodei de programare Greedy**, el selectând nodurile în ordinea crescătoare a distanței față de nodul sursă.

Observația 7.2.3. Pentru implementarea Algoritmului Dijkstra, considerăm că $V = \{1, \dots, n\}$ și că nodul sursă este $s \in V$. Utilizăm un vector S având semnificația

$$S[i] = \begin{cases} 1, & \text{dacă nodul } i \text{ a fost selectat,} \\ 0, & \text{în caz contrar,} \end{cases} \quad \forall i \in \{1, \dots, n\}$$

și un vector t având semnificația

$$t[i] = \text{distanța minimă de la nodul sursă } s \text{ la nodul } i, \quad \forall i \in \{1, \dots, n\},$$

calculat conform (7.2.1) și (7.2.2).

Pentru determinarea drumurilor minime de la nodul s la nodurile grafului vom utiliza și un vector $TATA$ având semnificația

$TATA[i] = \text{nodul } j \text{ ce este predecesorul direct al nodului } i \text{ pe drumul minim de la } s \text{ la } i, \forall i \in \{1, \dots, n\}.$

Astfel în vectorul $TATA$ se memorează un arbore compus din drumuri minime de la nodul sursă la nodurile grafului, numit **arborele drumurilor minime**.

Dacă $i = i_k$ este nodul selectat la pasul k , atunci $j = j_k$ se determină conform egalității (7.2.1).

Descrierea în pseudocod a algoritmului are forma următoare.

```

DIJKSTRA( $s$ ) :
for  $i = \overline{1, n}$  do                                     // inițializări
     $S[i] \leftarrow 0$ ;  $t[i] \leftarrow \infty$ ;  $TATA[i] \leftarrow \infty$ ;
 $t[s] \leftarrow 0$ ;  $TATA[s] \leftarrow 0$ ;                     //  $s$  este nodul sursă
repeat
    // selectăm următorul nod  $x$ , în ordinea crescătoare
    // a distanțelor minime de la  $s$  la  $x$ 
     $min \leftarrow \infty$ ;
    for  $i = \overline{1, n}$  do
        if ( $S[i] = 0$ ) and ( $t[i] < min$ ) then
             $min \leftarrow t[i]$ ;
             $x \leftarrow i$ ;
    if ( $min < \infty$ ) then                                 // există  $x$ , îl selectăm
         $S[x] \leftarrow 1$ ;
        for  $i = \overline{1, n}$  do                               // actualizăm vectorii  $t$  și  $TATA$ 
            if ( $S[i] = 0$ ) and ( $c_{xi} < \infty$ ) then
                if ( $t[i] > t[x] + c_{xi}$ ) then
                     $t[i] \leftarrow t[x] + c_{xi}$ ;
                     $TATA[i] \leftarrow x$ ;
while ( $min < \infty$ );

```

Exemplul 7.2.2. Pentru graful ponderat din Exemplul 5.2.1, luând ca nod sursă nodul $s = 1$, aplicarea Algoritmului Dijkstra este evidențiată în următorul tabel:

Pas	Nodul selectat x	$TATA[x]$	Distanța minimă $t[x]$
1	1	0	0
2	2	1	30
3	5	1	70
4	3	2	80
5	8	5	80
6	6	3	100
7	7	1	110
8	9	5	110
9	4	3	180
10	10	9	240

Arborele drumurilor minime, memorat în vectorul $TATA$, este reprezentat în Figura 7.2.1.

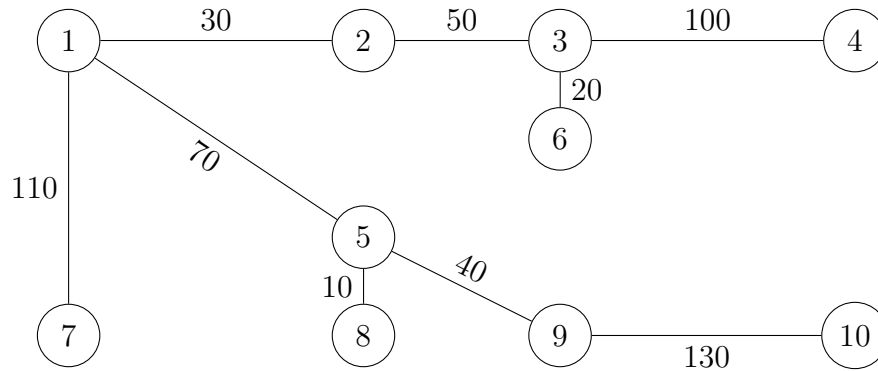


Figura 7.2.1:

Deci drumurile minime determinate de algoritm sunt:

- de la 1 la 1: [1];
- de la 1 la 2: [1, 2];
- de la 1 la 3: [1, 2, 3];
- de la 1 la 4: [1, 2, 3, 4];
- de la 1 la 5: [1, 5];
- de la 1 la 6: [1, 2, 3, 6];
- de la 1 la 7: [1, 7];
- de la 1 la 8: [1, 5, 8];
- de la 1 la 9: [1, 5, 9];
- de la 1 la 10: [1, 5, 9, 10].

Observația 7.2.4. Implementarea anterioară necesită $\mathcal{O}(n^2)$ operații (deoarece blocul ”**repeat**” se execută de cel mult n ori și necesită de fiecare dată cel mult n comparații pentru determinarea nodului selectat x și cel mult n comparații și n adunări pentru actualizarea vectorilor t și $TATA$). Aceasta este de fapt și complexitatea Algoritmului Dijkstra (în implementarea optimă) în cazul memorării grafului prin matricea distanțelor directe.

7.3 Algoritmul Roy-Floyd

Vom expune un algoritm pentru determinarea distanțelor minime și a drumurilor minime între orice două noduri ale grafului ponderat dat. Acest algoritm este asemănător cu Algoritmul Roy-Warshall pentru determinarea matricei drumurilor.

Algoritmul 7.3.1 (Roy-Floyd). Fie (G, c) un graf ponderat, $G = (V, E)$, $V = \{v_1, v_2, \dots, v_n\}$, $c : E \rightarrow \mathbb{R}_+$. Fie $C = (c_{ij})_{i,j=\overline{1,n}}$ matricea distanțelor directe asociată grafului (G, c) . Se calculează matricele

$$C^{(k)} = (c_{ij}^{(k)})_{i,j=\overline{1,n}}, \quad k \in \{0, 1, \dots, n\}$$

definite prin

$$C^{(0)} = C, \quad (7.3.1)$$

$$c_{ij}^{(k)} = \min\{c_{ij}^{(k-1)}, c_{ik}^{(k-1)} + c_{kj}^{(k-1)}\}, \quad \forall k, i, j \in \{1, \dots, n\}. \quad (7.3.2)$$

Teorema 7.3.1 (de corectitudine a Algoritmului Roy-Floyd). *În contextul Algoritmului Roy-Floyd, ultima matrice calculată este chiar matricea distanțelor minime asociată grafului (G, c) , adică*

$$C^{(n)} = C^*.$$

Demonstrație. Vom demonstra prin inducție după $k \in \{0, 1, \dots, n\}$ că pentru orice $i, j \in \{1, \dots, n\}$ avem

$$c_{ij}^{(k)} = \begin{cases} \min\{c(\mu) \mid \mu = (v_i, \dots, v_j), I(\mu) \subseteq \{v_1, \dots, v_k\}\}, & \text{dacă} \\ \quad \exists \mu = (v_i, \dots, v_j) \text{ drum cu } I(\mu) \subseteq \{v_1, \dots, v_k\}, \\ \infty, & \text{în caz contrar,} \end{cases} \quad (7.3.3)$$

unde $I(\mu)$ reprezintă mulțimea nodurilor intermediare ale drumului μ și $\{v_1, \dots, v_k\}$ reprezintă mulțimea $\{v_i \mid 1 \leq i \leq k\}$, deci pentru $k = 0$ această mulțime este \emptyset .

Pentru $k = 0$ avem $c_{ij}^{(0)} = c_{ij}$ (conform (7.3.1)) și egalitatea (7.3.3) este evidentă din definiția matricei C a costurilor directe și faptul că $I(\mu) = \emptyset$ înseamnă că drumul μ este de fapt un arc sau o muchie pentru $i \neq j$, respectiv că μ este o buclă sau drumul (v_i) de lungime 0 pentru $i = j$.

Presupunem acum egalitatea (7.3.3) adevărată pentru $k - 1$ ($1 \leq k \leq n$) și o demonstrăm pentru k . Fie $d \in [0, \infty)$. Folosind (7.3.2), ipoteza de inducție și principiul optimalității al lui Bellman avem echivalențele:

$$\begin{aligned} c_{ij}^{(k)} = d &\Leftrightarrow c_{ij}^{(k-1)} = d \leq c_{ik}^{(k-1)} + c_{kj}^{(k-1)} \text{ sau } c_{ik}^{(k-1)} + c_{kj}^{(k-1)} = d < c_{ij}^{(k-1)} \\ &\Leftrightarrow \exists \mu = (v_i, \dots, v_j) \text{ drum minim cu proprietatea că } I(\mu) \subseteq \{v_1, \dots, v_{k-1}\} \\ &\text{(adică de cost minim dintre toate drumurile de la } v_i \text{ la } v_j \text{ ce satisfac această} \\ &\text{proprietate), } c(\mu) = d \leq c(\mu_1) + c(\mu_2) \quad \forall \mu_1 = (v_i, \dots, v_k), \mu_2 = (v_k, \dots, v_j) \\ &\text{drumuri cu } I(\mu_1), I(\mu_2) \subseteq \{v_1, \dots, v_{k-1}\} \end{aligned}$$

sau

$$\begin{aligned} &\exists \mu'_1 = (v_i, \dots, v_k), \mu'_2 = (v_k, \dots, v_j) \text{ drumuri minime cu proprietatea că} \\ &I(\mu'_1), I(\mu'_2) \subseteq \{v_1, \dots, v_{k-1}\}, c(\mu'_1) + c(\mu'_2) = d < c(\mu''), \quad \forall \mu'' = (v_i, \dots, v_j) \\ &\text{drum cu } I(\mu'') \subseteq \{v_1, \dots, v_{k-1}\} \end{aligned}$$

$$\Leftrightarrow \exists \mu = (v_i, \dots, v_j) \text{ drum minim cu proprietatea că} \\ I(\mu) \subseteq \{v_1, \dots, v_k\}, v_k \notin I(\mu), c(\mu) = d$$

sau

$$\begin{aligned} &\exists \mu' = (v_i, \dots, v_k, \dots, v_j) \text{ drum minim cu proprietatea că} \\ &I(\mu') \subseteq \{v_1, \dots, v_k\}, v_k \in I(\mu'), c(\mu') = d \\ &(\mu' \text{ se obține parcurgând întâi } \mu'_1, \text{ apoi } \mu'_2 \text{ și, reciproc,} \\ &\mu'_1 \text{ și } \mu'_2 \text{ sunt porțiunile din } \mu' \text{ dintre } v_i \text{ și prima apariție a lui } v_k \text{ în } I(\mu'), \\ &\text{respectiv dintre ultima apariție a lui } v_k \text{ în } I(\mu') \text{ și } v_j; \text{ între aceste apariții} \\ &\text{eventualele arce sau muchii ale lui } \mu' \text{ au costul 0 conform Observației 7.1.1)} \\ &\Leftrightarrow \exists \mu = (v_i, \dots, v_j) \text{ drum minim cu proprietatea că} \\ &I(\mu) \subseteq \{v_1, \dots, v_k\}, c(\mu) = d. \end{aligned}$$

Demonstrația prin inducție a egalității (7.3.3) este astfel încheiată.

Pentru $k = n$ condiția $I(\mu) \subseteq \{v_1, \dots, v_n\}$ poate fi eliminată, fiind întotdeauna adevărată, deci din (7.3.3) și Definiția 7.1.2 obținem $c_{ij}^{(n)} = c_{ij}^*$, $\forall i, j \in \{1, \dots, n\}$, adică egalitatea din enunț. \square

Observația 7.3.1. Algoritmul Roy-Floyd are complexitatea $\Theta(n^3)$ (deoarece necesită câte o adunare și o comparație pentru fiecare triplet (k, i, j) , cu $k, i, j \in \{1, \dots, n\}$).

Exemplul 7.3.1. Pentru graful din Exemplul 7.1.1, prin aplicarea Algoritmului Roy-Floyd obținem matricele:

$$C^{(0)} = C = \begin{pmatrix} 0 & 15 & 10 & 5 & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 5 & 0 & \infty & 5 \\ \infty & 10 & \infty & 0 & 5 \\ 5 & 5 & \infty & \infty & 0 \end{pmatrix}; \quad C^{(1)} = \begin{pmatrix} 0 & 15 & 10 & 5 & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 5 & 0 & \infty & 5 \\ \infty & 10 & \infty & 0 & 5 \\ 5 & 5 & 15 & 10 & 0 \end{pmatrix}$$

(deoarece, de exemplu, $c_{53}^{(1)} = \min\{c_{53}^{(0)}, c_{51}^{(0)} + c_{13}^{(0)}\} = \min\{\infty, 5 + 10\} = 15$);

$$C^{(2)} = \begin{pmatrix} 0 & 15 & 10 & 5 & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 5 & 0 & \infty & 5 \\ \infty & 10 & \infty & 0 & 5 \\ 5 & 5 & 15 & 10 & 0 \end{pmatrix}; \quad C^{(3)} = \begin{pmatrix} 0 & 15 & 10 & 5 & 15 \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 5 & 0 & \infty & 5 \\ \infty & 10 & \infty & 0 & 5 \\ 5 & 5 & 15 & 10 & 0 \end{pmatrix};$$

$$C^{(4)} = \begin{pmatrix} 0 & 15 & 10 & 5 & 10 \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 5 & 0 & \infty & 5 \\ \infty & 10 & \infty & 0 & 5 \\ 5 & 5 & 15 & 10 & 0 \end{pmatrix}; \quad C^{(5)} = \begin{pmatrix} 0 & 15 & 10 & 5 & 10 \\ \infty & 0 & \infty & \infty & \infty \\ 10 & 5 & 0 & 15 & 5 \\ 10 & 10 & 20 & 0 & 5 \\ 5 & 5 & 15 & 10 & 0 \end{pmatrix} = C^*$$

(matricea distanțelor minime).

Observația 7.3.2. Conform ecuațiilor (7.3.3), Algoritmul Roy-Floyd este un algoritm specific **metodei programării dinamice**.

Conform ecuațiilor (7.3.2) avem $c_{ik}^{(k)} = c_{ik}^{(k-1)}$ și $c_{kj}^{(k)} = c_{kj}^{(k-1)}$, $\forall k, i, j \in \{1, \dots, n\}$, deci pentru implementarea algoritmului putem utiliza o singură matrice $C^{(k)}$. Astfel obținem următoarea descriere în pseudocod a algoritmului.

ROY_FLOYD:

```

for  $i = \overline{1, n}$  do                                     // inițializări
    for  $j = \overline{1, n}$  do
         $c^*[i, j] \leftarrow c[i, j];$ 
for  $k = \overline{1, n}$  do
    for  $i = \overline{1, n}$  do
        for  $j = \overline{1, n}$  do
            if  $(c^*[i, k] + c^*[k, j] < c^*[i, j])$  then
                 $c^*[i, j] \leftarrow c^*[i, k] + c^*[k, j]$ 

```

Considerând că funcția cost c este strict pozitivă, pentru determinarea tuturor drumurilor minime (elementare) dintre două noduri distincte x, y putem utiliza echivalența:

$$(x, z, \dots, y) = \text{drum minim} \Leftrightarrow \begin{cases} z \neq x, \\ c_{xz} + c_{zy}^* = c_{xy}^*, \end{cases}$$

unde mulțimea nodurilor este mulțimea standard $V = \{1, \dots, n\}$.

Astfel putem determina toate nodurile z ce sunt succesori direcți ai nodului x pe drumuri minime de la x la y , și continuând acest procedeu pentru subdrumurile minime dintre z și y se găsesc toate drumurile minime elementare de la x la y .

Tema 8

Fluxuri în rețele

8.1 Problema fluxului maxim

Definiția 8.1.1. O rețea (rețea de transport) are forma $R = (G, s, t, c)$, unde:

- $G = (V, E)$ este un graf orientat simplu.
 V se numește și **mulțimea nodurilor** rețelei, iar E se numește și **mulțimea arcelor** rețelei;
- $s, t \in V$ sunt două noduri a.î. $s \neq t$.
 Nodul s se numește **nodul sursă (intrarea)** al rețelei, iar nodul t se numește **nodul destinație (ieșirea)** al rețelei;
- $c : E \rightarrow \mathbb{R}_+$ este o funcție numită **funcție capacitate**.
 Pentru orice $e \in E$, $c(e)$ se numește **capacitatea arcului** e .

Pentru simplificarea notațiilor, extindem funcția capacitate

$$c : V \times V \rightarrow \mathbb{R}_+, c(i, j) = \begin{cases} c(i, j), & \text{dacă } (i, j) \in E, \\ 0, & \text{dacă } (i, j) \notin E. \end{cases} \quad (8.1.1)$$

Exemplul 8.1.1. În Figura 8.1.1 este reprezentată o rețea R , capacitățile fiind menționate pe arce. Considerăm că această rețea are intrarea $s = 1$ și ieșirea $t = 6$.

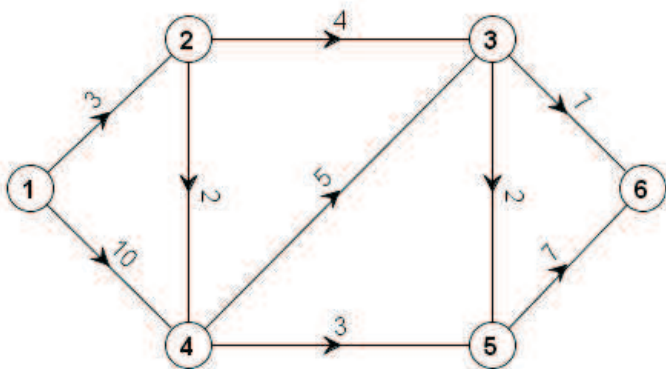


Figura 8.1.1:

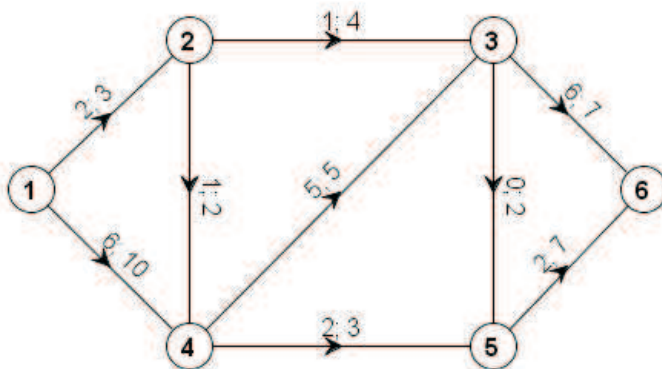


Figura 8.1.2:

Definiția 8.1.2. Fie $R = (G, s, t, c)$ o rețea, unde $G = (V, E)$. Un **flux** în rețeaua R este o funcție $f : V \times V \rightarrow \mathbb{R}$ ce verifică următoarele două proprietăți:

$$0 \leq f(i, j) \leq c(i, j), \quad \forall i, j \in V, \quad (8.1.2)$$

$$\sum_{j \in V} f(j, i) = \sum_{j \in V} f(i, j), \quad \forall i \in V \setminus \{s, t\}. \quad (8.1.3)$$

$f(i, j)$ reprezintă **fluxul transportat pe arcul** (i, j) .

Relația (8.1.2) se numește **condiția de mărginire a fluxului**, iar relația (8.1.3) se numește **condiția de conservare a fluxului**.

Observația 8.1.1. Evident,

$$f(i, j) = 0 \quad \forall (i, j) \notin E,$$

deci, analog funcției capacitate c , și funcția flux f poate fi definită (restrânsă) doar pe mulțimea arcelor rețelei (adică $f : E \rightarrow \mathbb{R}$).

Observația 8.1.2. Definițiile și rezultatele din acest capitol sunt valabile și pentru rețele formate cu grafuri orientate oarecare (nu neapărat simple). În acest caz general funcțiile capacitate și flux se definesc neapărat doar pe colecția (multisetul) E a arcelor rețelei.

Observația 8.1.3. În particular, $f(i, j) = 0 \quad \forall i, j \in V$ este un flux în orice rețea, numit **fluxul nul**.

Exemplul 8.1.2. Pentru rețeaua din Figura 8.1.1, un exemplu de flux este reprezentat în Figura 8.1.2. Pe fiecare arc (i, j) sunt menționate fluxul $f(i, j)$ și capacitatea $c(i, j)$, în această ordine.

Lema 8.1.1. Fie $R = (G, s, t, c)$ o rețea, unde $G = (V, E)$. Pentru orice flux f în rețeaua R are loc egalitatea

$$\sum_{i \in V} f(i, t) - \sum_{i \in V} f(t, i) = - \left(\sum_{i \in V} f(i, s) - \sum_{i \in V} f(s, i) \right).$$

Definiția 8.1.3. Fie $R = (G, s, t, c)$ o rețea, unde $G = (V, E)$. Pentru orice flux f în rețeaua R , numărul

$$v(f) = \sum_{i \in V} f(i, t) - \sum_{i \in V} f(t, i) = - \left(\sum_{i \in V} f(i, s) - \sum_{i \in V} f(s, i) \right)$$

se numește **valoarea fluxului** f (fluxul net ce ajunge la nodul destinație, fluxul net ce iese din nodul sursă).

Exemplul 8.1.3. Fluxul din Figura 8.1.2 are valoarea $v(f) = 2 + 6 = 8$.

Definiția 8.1.4. Fie $R = (G, s, t, c)$ o rețea. Un flux f^* în rețeaua R cu proprietatea că

$$v(f^*) = \max\{v(f) \mid f = \text{flux în } R\}$$

se numește **flux de valoare maximă (flux maxim)** în rețeaua R .

Problema determinării fluxurilor maxime într-o rețea are numeroase aplicații practice. În secțiunea următoare vom prezenta un algoritm pentru rezolvarea acestei probleme.

8.2 Algoritmul Ford-Fulkerson

Începem cu definirea unor noțiuni ajutătoare.

Definiția 8.2.1. Dacă $\mu = [x_0, x_1, \dots, x_k]$ este un lanț elementar în graful orientat $G = (V, E)$, atunci arcele sale de forma $(x_i, x_{i+1}) \in E$ ($i \in \{0, \dots, k-1\}$) se numesc **arce directe** pentru lanțul μ , iar arcele sale de forma $(x_{i+1}, x_i) \in E$ ($i \in \{0, \dots, k-1\}$) se numesc **arce inverse** pentru lanțul μ .

Exemplul 8.2.1. Pentru graful din Figura 8.1.2, lanțul $[1, 2, 3, 4, 5, 6]$ are arcele directe $(1, 2), (2, 3), (4, 5), (5, 6)$ și arcul invers $(4, 3)$.

Definiția 8.2.2. Fie f un flux în rețeaua $R = (G, s, t, c)$. Un **C-lanț** în rețeaua R relativ la fluxul f este un lanț elementar μ în graful G ce verifică următoarele două proprietăți:

$$\begin{aligned} f(i, j) &< c(i, j), \quad \forall (i, j) = \text{arc direct al lui } \mu, \\ f(i, j) &> 0, \quad \forall (i, j) = \text{arc invers al lui } \mu. \end{aligned}$$

Definiția 8.2.3. Fie f un flux în rețeaua $R = (G, s, t, c)$ și fie μ un C-lanț în rețeaua R relativ la fluxul f .

a) Pentru orice arc (i, j) al lui μ , numărul

$$r_\mu(i, j) = \begin{cases} c(i, j) - f(i, j), & \text{dacă } (i, j) \text{ este arc direct al lui } \mu, \\ f(i, j), & \text{dacă } (i, j) \text{ este arc invers al lui } \mu \end{cases}$$

se numește **capacitatea reziduală (reziduul)** a arcului (i, j) relativ la C-lanțul μ .

b) Numărul

$$r(\mu) = \min\{r_\mu(i, j) \mid (i, j) = \text{arc al lui } \mu\}$$

se numește **capacitatea reziduală (reziduul)** a C-lanțului μ .

Observația 8.2.1. Conform definițiilor anterioare, pentru orice C-lanț μ avem $r(\mu) > 0$.

Exemplul 8.2.2. Pentru rețeaua și fluxul reprezentate în Figura 8.1.2, lanțul $\mu = [1, 2, 3, 4, 5, 6]$ este un C-lanț. Reziduurile pe arcele acestui C-lanț sunt

$$r_\mu(1, 2) = 3 - 2 = 1, \quad r_\mu(2, 3) = 4 - 1 = 3, \quad r_\mu(4, 3) = 5, \quad r_\mu(4, 5) = 3 - 2 = 1, \quad r_\mu(5, 6) = 7 - 2 = 5,$$

deci reziduul pe acest C-lanț este $r(\mu) = 1$.

Definiția 8.2.4. Fie f un flux în rețeaua $R = (G, s, t, c)$. Un C-lanț de la s la t în rețeaua R relativ la fluxul f se numește **lanț de creștere** (în rețeaua R relativ la fluxul f).

Exemplul 8.2.3. Lanțul $\mu = [1, 2, 3, 4, 5, 6]$ din exemplul anterior este un lanț de creștere.

Lema 8.2.1. Fie $R = (G, s, t, c)$ o rețea, unde $G = (V, E)$. Fie f un flux în rețeaua R și fie μ un lanț de creștere în rețeaua R relativ la fluxul f . Atunci funcția $f' : V \times V \rightarrow \mathbb{R}$ definită prin

$$f'(i, j) = \begin{cases} f(i, j) + r(\mu), & \text{dacă } (i, j) \text{ este arc direct al lui } \mu, \\ f(i, j) - r(\mu), & \text{dacă } (i, j) \text{ este arc invers al lui } \mu, \\ f(i, j), & \text{dacă } (i, j) \text{ nu este arc al lui } \mu \end{cases}$$

este un flux în rețeaua R și

$$v(f') = v(f) + r(\mu).$$

Observația 8.2.2. În contextul lemei anterioare, conform Observației 8.2.1 avem

$$v(f') = v(f) + r(\mu) > v(f).$$

Inegalitatea justifică denumirea lui μ drept *lanț de creștere* a fluxului f , iar egalitatea justifică denumirea lui $r(\mu)$ drept *capacitatea reziduală* a lanțului μ .

Definiția 8.2.5. Fluxul f' definit în lema anterioară se numește **fluxul obținut prin mărirea fluxului f de-a lungul lanțului de creștere μ** și se notează cu

$$f' = f \oplus r(\mu).$$

Exemplul 8.2.4. Pentru rețeaua R și fluxul f reprezentate în Figura 8.1.2, lanțul $\mu = [1, 2, 3, 4, 5, 6]$ este un lanț de creștere având reziduul $r(\mu) = 1$. Fluxul $f' = f \oplus r(\mu)$ obținut prin aplicarea lemei anterioare este reprezentat în Figura 8.2.1. Valoarea acestui flux este $v(f') = v(f) + r(\mu) = 8 + 1 = 9$.

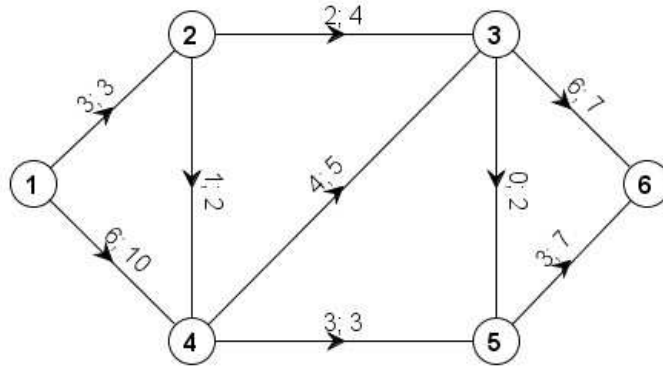


Figura 8.2.1:

Definiția 8.2.6. Fie $R = (G, s, t, c)$ o rețea, unde $G = (V, E)$. O **secțiune (tăietură)** în rețeaua R este o pereche $(S, T) \in V \times V$ ce verifică următoarele proprietăți:

$$\begin{aligned} S \cup T &= V, \quad S \cap T = \emptyset, \\ s &\in S, \quad t \in T. \end{aligned}$$

Observația 8.2.3. Dacă (S, T) este o secțiune în rețeaua $R = (G, s, t, c)$, unde $G = (V, E)$, atunci $V = S \cup T$ este o partiție a mulțimii V a nodurilor rețelei (adică $V = S \cup T$, $S \neq \emptyset$, $T \neq \emptyset$, $S \cap T = \emptyset$) a.î. $s \in S$ și $t \in T$. Rezultă că numărul de secțiuni ale rețelei R este egal cu numărul de submulțimi $S \setminus \{s\} \subseteq V \setminus \{s, t\}$, deci cu

$$2^{n-2}, \text{ unde } n = \text{card}(V).$$

Definiția 8.2.7. Fie $R = (G, s, t, c)$ o rețea, unde $G = (V, E)$. Pentru orice secțiune (S, T) în rețeaua R , numărul

$$c(S, T) = \sum_{i \in S} \sum_{j \in T} c(i, j)$$

se numește **capacitatea secțiunii** (S, T) .

Observația 8.2.4. Conform relației (8.1.1), capacitatea unei secțiuni (S, T) este suma capacităților tuturor arcelor ce au extremitatea inițială în S și extremitatea finală în T .

Exemplul 8.2.5. Pentru rețeaua reprezentată în Figura 8.1.1, $(S, T) = (\{1, 2, 5\}, \{3, 4, 6\})$ este o secțiune având capacitatea

$$c(S, T) = c(1, 4) + c(2, 3) + c(2, 4) + c(5, 6) = 10 + 4 + 2 + 7 = 23.$$

Lema 8.2.2. Fie $R = (G, s, t, c)$ o rețea. Pentru orice flux f și orice secțiune (S, T) în rețeaua R avem

$$v(f) = \sum_{i \in S} \sum_{j \in T} (f(i, j) - f(j, i))$$

(adică valoarea fluxului este egală cu fluxul net ce traversează secțiunea).

Lema 8.2.3. Fie $R = (G, s, t, c)$ o rețea.

a) Pentru orice flux f și orice secțiune (S, T) în rețeaua R avem

$$v(f) \leq c(S, T).$$

b) Dacă f^* este un flux și (S^*, T^*) este o secțiune în rețeaua R astfel încât

$$v(f^*) = c(S^*, T^*),$$

atunci f^* este un flux de valoare maximă și (S^*, T^*) este o secțiune de capacitate minimă în rețeaua R .

Teorema 8.2.1 (de caracterizare a unui flux de valoare maximă). Fie $R = (G, s, t, c)$ o rețea. Un flux f în rețeaua R este un flux de valoare maximă dacă și numai dacă nu există lanțuri de creștere în rețeaua R relativ la fluxul f .

Demonstrație. "⇒" Fie f un flux de valoare maximă în R . Dacă ar exista un lanț de creștere μ în R relativ la fluxul f , atunci conform Lemei 8.2.1 ar rezulta că funcția $f' = f \oplus r(\mu)$ ar fi un flux în R și $v(f') = v(f) + r(\mu) > v(f)$ (conform Observației 8.2.2) contradicție. Deci nu există lanțuri de creștere în R relativ la fluxul f .

"⇐" Fie f un flux în R a.î. nu există lanțuri de creștere în R relativ la f . Fie

$$\begin{cases} S = \{i \in V \mid \text{există } C\text{-lanțuri de la } s \text{ la } i \text{ în } R \text{ relativ la } f\}, \\ T = \{i \in V \mid \text{nu există } C\text{-lanțuri de la } s \text{ la } i \text{ în } R \text{ relativ la } f\}. \end{cases} \quad (8.2.1)$$

Evident, $S \cup T = V$ și $S \cap T = \emptyset$.

De asemenea, avem $s \in S$ (deoarece $[s]$ este un C -lanț de la s la s în R relativ la f) și $t \in T$ (deoarece nu există lanțuri de creștere, adică C -lanțuri de la s la t în R relativ la f). Deci (S, T) este o secțiune în rețeaua R .

Fie $i \in S$ și $j \in T$ arbitrar fixați. Deoarece $i \in S$ rezultă că există un C -lanț $\mu = [s = x_0, x_1, \dots, x_k = i]$ în R relativ la f . Atunci $x_0, x_1, \dots, x_k \in S$ (sublanțurile C -lanțului μ sunt tot C -lanțuri), deci $j \notin V(\mu)$.

Dacă am avea $f(i, j) < c(i, j)$ sau $f(j, i) = 0$, atunci ar rezulta că $[s = x_0, x_1, \dots, x_k = i, j]$ ar fi un C -lanț în R relativ la f , ceea ce ar contrazice apartenența $j \in T$. Deci

$$f(i, j) = c(i, j) \text{ și } f(j, i) = 0,$$

pentru orice $i \in S$ și $j \in T$.

Aplicând Lema 8.2.2, avem

$$v(f) = \sum_{i \in S} \sum_{j \in T} (f(i, j) - f(j, i)) = \sum_{i \in S} \sum_{j \in T} (c(i, j) - 0) = c(S, T),$$

deci conform Lemei 8.2.3 rezultă că f este un flux de valoare maximă în rețeaua R (iar (S, T) este o secțiune de capacitate minimă în rețeaua R). \square

Observația 8.2.5. Dacă se cunoaște un flux f de valoare maximă într-o rețea R , atunci formulele (8.2.1) determină o secțiune (S, T) de capacitate minimă, deci orice algoritm pentru determinarea unui flux de valoare maximă bazat pe caracterizarea dată în teorema anterioară rezolvă și problema determinării unei secțiuni de capacitate minimă.

Definiția 8.2.8. Fie f un flux în rețeaua $R = (G, s, t, c)$, unde $G = (V, E)$.

Funcția $r : V \times V \rightarrow \mathbb{R}_+$ definită prin

$$r(i, j) = \begin{cases} c(i, j) - f(i, j), & \text{dacă } f(i, j) < c(i, j), \\ f(j, i), & \text{dacă } f(j, i) > 0 \text{ și } f(i, j) = c(i, j), \\ 0, & \text{în rest} \end{cases}$$

se numește **capacitatea reziduală (reziduul)** a rețelei R relativ la fluxul f , iar graful orientat $G_f = (V, E_f)$ definit prin

$$E_f = \{(i, j) \in V \times V \mid r(i, j) > 0\}$$

se numește **graful rezidual** al rețelei R relativ la fluxul f .

Observația 8.2.6. Fie f un flux în rețeaua $R = (G, s, t, c)$. Fie r și G_f capacitatea reziduală, respectiv graful rezidual ale rețelei R relativ la fluxul f . Fie $\mu = [x_0, x_1, \dots, x_k]$ un lanț elementar în graful G . Evident, avem echivalențele:

$$\begin{aligned} \mu \text{ este un } C\text{-lanț în } R \text{ relativ la } f &\Leftrightarrow r(x_i, x_{i+1}) > 0, \forall i \in \{0, \dots, k-1\} \Leftrightarrow \\ &\Leftrightarrow (x_0, x_1, \dots, x_k) \text{ este un drum elementar în graful rezidual } G_f; \\ \mu \text{ este un lanț de creștere în } R \text{ relativ la } f &\Leftrightarrow \\ &\Leftrightarrow (x_0, x_1, \dots, x_k) \text{ este un drum elementar de la } s \text{ la } t \text{ în graful rezidual } G_f. \end{aligned}$$

Exemplul 8.2.6. Pentru rețeaua R și fluxul f' reprezentate în Figura 8.2.1, graful rezidual $G_{f'}$ este reprezentat în Figura 8.2.2, capacitățile reziduale fiind menționate pe arce.

$(1, 4, 2, 3, 6)$ este un drum elementar în graful rezidual $G_{f'}$, deci $\mu' = [1, 4, 2, 3, 6]$ este un lanț de creștere în rețeaua R relativ la fluxul f' .

Implementarea Algoritmului Ford-Fulkerson

Conform rezultatelor de mai sus obținem următorul algoritm pentru determinarea unui flux de valoare maximă într-o rețea.

Algoritmul 8.2.1 (Ford-Fulkerson). Fie $R = (G, s, t, c)$ o rețea, unde $G = (V, E)$.

Conform Teoremei 8.2.1, schița algoritmului, descrisă în pseudocod, are forma

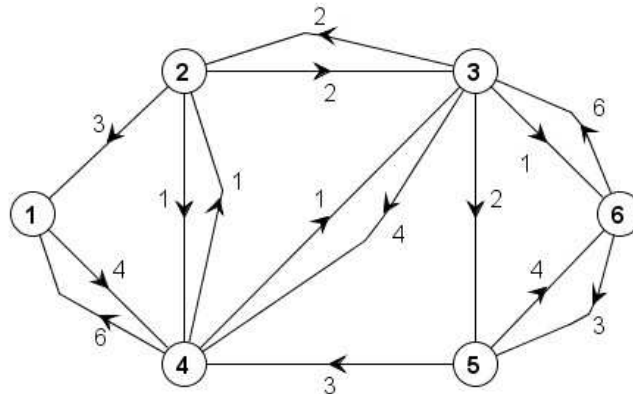


Figura 8.2.2:

FORD_FULKERSON :

```

f ← 0;                                     // sau f ← f0, unde f0 este un flux
                                           // disponibil inițial

repeat
    if (EXISTĂ_LANȚ_DE_CREȘTERE) then
        μ ← LANȚ_DE_CREȘTERE;               // există lanțuri de creștere
                                           // relativ la fluxul curent f
        f ← f ⊕ r(μ);                       // se determină
                                           // un astfel de lanț de creștere
        // se mărește valoarea fluxului curent,
        // de-a lungul lanțului de creștere
    while (EXISTĂ_LANȚ_DE_CREȘTERE);
                                           // nu mai există lanțuri de creștere,
                                           // deci fluxul curent este de valoare maximă
AFIȘARE(f);                                // se afișează fluxul de valoare maximă

```

În continuare detaliem implementarea acestui algoritm. Presupunem că

$$V = \{1, 2, \dots, n\}, \quad s = 1, \quad t = n,$$

cu $n \geq 2$.

Pentru depistarea și memorarea eventualelor lanțuri de creștere relativ la fluxul curent vom utiliza doi vectori SEL și $TATA$ având semnificația

$$SEL[i] = \begin{cases} 1, & \text{dacă există un } C\text{-lanț de la nodul } s = 1 \text{ la nodul } i, \\ 0, & \text{în caz contrar,} \end{cases} \quad \forall i \in \{1, \dots, n\},$$

$$TATA[i] = \text{predecesorul direct al nodului } i \text{ pe } C\text{-lanțul de la nodul } s = 1 \text{ la nodul } i, \quad \forall i \in \{1, \dots, n\}.$$

Algoritmul descris în pseudocod are forma (detaliată)

FORD_FULKERSON :**CITIRE_REȚEA;**

// se citește rețeaua dată

for $i = \overline{1, n}$ **do**└ **for** $j = \overline{1, n}$ **do** $f[i, j] \leftarrow 0;$

// fluxul inițial

 $vmax \leftarrow 0;$

// valoarea fluxul maxim

repeat

// se caută lanțuri de creștere a fluxului curent,

// folosind Observația 8.2.6

CALCUL_REZIDUURI;

// se determină reziduul și

// graful rezidual ale rețelei relativ la fluxul curent

 $SEL[1] \leftarrow 1;$

// se selectează nodul sursă

for $i = \overline{2, n}$ **do** $SEL[i] \leftarrow 0;$ $TATA[1] \leftarrow 0;$

// pt. memorarea C-lanțurilor

// ce pornesc din nodul sursă

DF(1);

// se parcurge graful rezidual, memorând

// C-lanțurile ce pornesc din nodul sursă;

// parcurgerea DF, poate fi înlocuită cu

// parcurgerea BF (*Algoritmul Edmonds-Karp*)**if** ($SEL[n] = 1$) **then**

// s-a selectat nodul destinație,

// deci există lanț de creștere a fluxului

 $rmin \leftarrow \infty;$

// reziduul minim, de-a lungul

// lanțului de creștere

DET_LANT_CR;

// se determină lanțul de creștere

// a fluxului și reziduul minim

MĂRIRE_FLUX;

// se mărește fluxul curent, de-a

// lungul lanțului de creștere

 $vmax \leftarrow vmax + rmin;;$

// se actualizează valoarea

// fluxului curent

while ($SEL[n] = 1$);

// nu mai există lanțuri de creștere a fluxului,

// deci fluxul curent este maxim

AFIȘARE_REZULTATE;

// se afișează fluxul maxim și valoarea sa,

// eventual și secțiunea de capacitate minimă,

// calculată conform (8.2.1) astfel:

// $S = \{i \in \{1, \dots, n\} \mid SEL[i] = 1\},$ // $T = \{i \in \{1, \dots, n\} \mid SEL[i] = 0\}$

Funcțiile utilizate sunt descrise în continuare.

CALCUL_REZIDUURI :

```

for  $i = \overline{1, n}$  do
    for  $j = \overline{1, n}$  do
        if  $(f[i, j] < c[i, j])$  then
            |  $r[i, j] \leftarrow c[i, j] - f[i, j];$ 
        else
            | if  $(f[j, i] > 0)$  then
            | |  $r[i, j] \leftarrow f[j, i];$ 
            | else
            | |  $r[i, j] \leftarrow 0;$ 

```

DF(i) : // recursiv

```

for  $j = \overline{1, n}$  do
    if  $(r[i, j] > 0 \text{ and } SEL[j] = 0)$  then
        |  $TATA[j] \leftarrow i; SEL[j] \leftarrow 1;;$ 
        | DF( $j$ );

```

DET_LANT_CR :

// se determină reziduul minim de-a
// lungul lanțului de creștere, parcurs de la n către 1

```

 $j \leftarrow n;$ 
while  $(j \neq 1)$  do
    |  $i \leftarrow TATA[j];$ 
    | if  $(rmin > r[i, j])$  then  $rmin \leftarrow r[i, j];$ 
    |  $j \leftarrow i;$ 

```

MĂRIRE_FLUX :

// se mărește fluxul de-a lungul
// lanțului de creștere, parcurs de la n către 1

```

 $j \leftarrow n;$ 
while  $(j \neq 1)$  do
    |  $i \leftarrow TATA[j];$ 
    | if  $(c[i, j] > f[i, j])$  then
    | |  $f[i, j] \leftarrow f[i, j] + rmin;$ 
    | else
    | |  $f[j, i] \leftarrow f[j, i] - rmin;$ 
    |  $j \leftarrow i;$ 

```

Teorema 8.2.2 (de corectitudine a Algoritmului Ford-Fulkerson). În contextul Algoritmului 8.2.1, fie $R = (G, s, t, c)$ rețeaua dată, unde $G = (V, E)$. Presupunem că toate capacitățile arcelor rețelei sunt numere întregi, adică

$$c(i, j) \in \mathbb{N}, \forall (i, j) \in E.$$

Fie f_0 fluxul inițial. Presupunem că toate componentele fluxului f_0 sunt numere întregi, adică

$$f_0(i, j) \in \mathbb{N}, \forall (i, j) \in E,$$

de exemplu $f_0 = \text{fluxul nul}$.

Fie $\mu_1, \mu_2, \dots, \mu_k$ lanțurile de creștere succesive obținute, $k \geq 0$, și fie $f_1 = f_0 \oplus r_0(\mu_1)$, $f_2 = f_1 \oplus r_1(\mu_2)$, \dots , $f_k = f_{k-1} \oplus r_{k-1}(\mu_k)$ fluxurile succesive obținute, unde μ_1 este lanț de creștere relativ la f_0 , μ_2 este lanț de creștere relativ la f_1 , \dots , μ_k este lanț de creștere relativ la f_{k-1} , și nu mai există lanțuri de creștere relativ la f_k . Atunci f_k este un flux de valoare maximă în rețeaua R .

Mai mult, toate componentele fluxului f_k și valoarea acestuia sunt numere întregi.

Demonstrație. Conform Lemei 8.2.1, f_1, f_2, \dots, f_k sunt fluxuri și $v(f_1) = v(f_0) + r_0(\mu_1)$, $v(f_2) = v(f_1) + r_1(\mu_2) = v(f_0) + r_0(\mu_1) + r_1(\mu_2)$, \dots , $v(f_k) = v(f_{k-1}) + r_{k-1}(\mu_k) = v(f_0) + r_0(\mu_1) + r_1(\mu_2) + \dots + r_{k-1}(\mu_k)$.

Deoarece toate capacitățile $c(i, j)$, $i, j \in V$, și toate componentele $f_0(i, j)$, $i, j \in V$ ale fluxului inițial sunt numere întregi, conform Definiției 8.2.3 și Lemei 8.2.1 rezultă succesiv că

$$\begin{aligned} (r_0)_{\mu_1}(i, j) &\in \mathbb{N}^*, \forall (i, j) \in E(\mu_1), r_0(\mu_1) \in \mathbb{N}^*, \\ f_1(i, j) &\in \mathbb{N}, \forall i, j \in V, v(f_1) \in \mathbb{N}^*; \\ (r_1)_{\mu_2}(i, j) &\in \mathbb{N}^*, \forall (i, j) \in E(\mu_2), r_1(\mu_2) \in \mathbb{N}^*, \\ f_2(i, j) &\in \mathbb{N}, \forall i, j \in V, v(f_2) \in \mathbb{N}^*; \dots, \\ (r_{k-1})_{\mu_k}(i, j) &\in \mathbb{N}^*, \forall (i, j) \in E(\mu_k), r_{k-1}(\mu_k) \in \mathbb{N}^*, \\ f_k(i, j) &\in \mathbb{N}, \forall i, j \in V, v(f_k) \in \mathbb{N}^*. \end{aligned} \quad (8.2.2)$$

Deci

$$v(f_k) = v(f_0) + r_0(\mu_1) + r_1(\mu_2) + \dots + r_{k-1}(\mu_k) \geq v(f_0) + k.$$

Evident, $(\{s\}, V \setminus \{s\})$ este o secțiune în R având capacitatea

$$c(\{s\}, V \setminus \{s\}) = \sum_{j \in V \setminus \{s\}} c(s, j) \leq (\text{card}(V) - 1)c_{\max},$$

unde

$$c_{\max} = \max\{c(i, j) \mid (i, j) \in E\}$$

(capacitatea maximă a arcelor rețelei). Conform Lemei 8.2.3 avem

$$v(f_k) \leq c(\{s\}, V \setminus \{s\}),$$

și astfel obținem

$$k \leq (\text{card}(V) - 1)c_{\max} - v(f_0),$$

deci numărul k de pași ai algoritmului (numărul de lanțuri de creștere succesive construite) este finit, adică există $k \in \mathbb{N}$ astfel încât rețeaua R nu mai conține lanțuri de creștere relativ la fluxul f_k . Conform Teoremei 8.2.1 rezultă că f_k este un flux de valoare maximă în R . Conform (8.2.2), toate componentele fluxului f_k și valoarea acestuia sunt numere întregi. \square

Exemplu 8.2.7. Pentru rețeaua R din Figura 8.1.1, aplicarea Algoritmului Ford-Fulkerson este evidențiată în Figurile 8.2.3-8.2.8.

Lanțurile succesive de creștere relativ la fluxul curent sunt evidențiate prin îngroșare, și anume:

$$\begin{aligned} \mu_1 &= [1, 2, 3, 5, 6], \\ \mu_2 &= [1, 2, 3, 6], \\ \mu_3 &= [1, 4, 3, 6], \\ \mu_4 &= [1, 4, 5, 3, 6], \\ \mu_5 &= [1, 4, 5, 6]. \end{aligned}$$

Fluxul reprezentat în Figura 8.2.8 este de valoare maximă, deoarece nu mai există lanțuri de creștere în R relativ la acest flux. Valoarea acestui flux este 11.

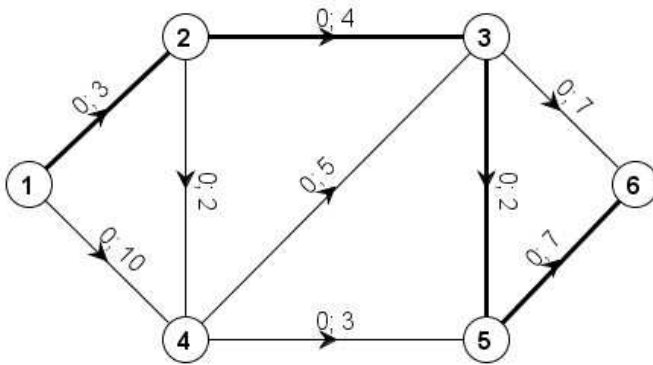


Figura 8.2.3:

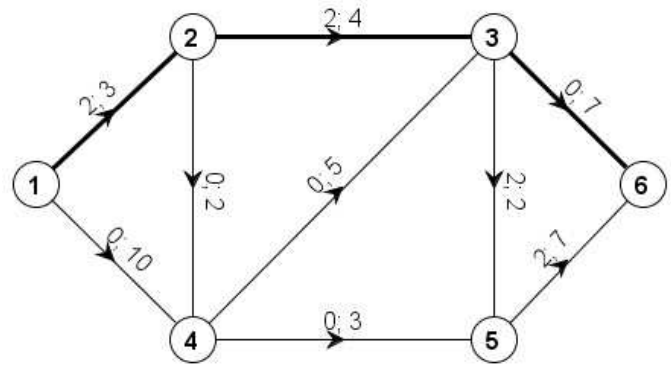


Figura 8.2.4:

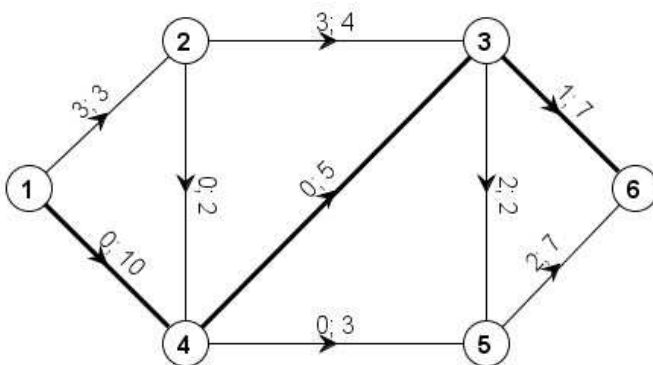


Figura 8.2.5:

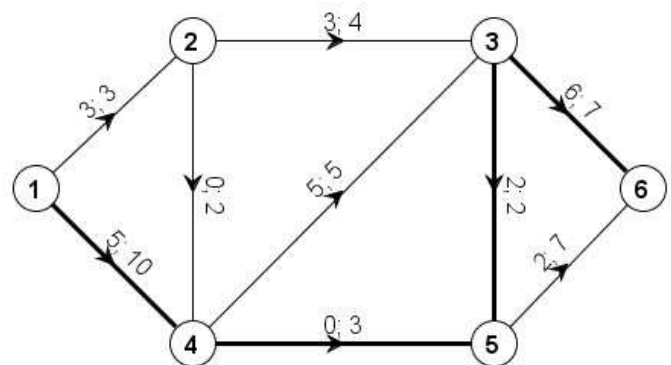


Figura 8.2.6:

Pentru acest flux maxim, avem C -lanțuri de la nodul sursă $s = 1$ doar la nodurile 1 și 4, aceste C -lanțuri sunt evidențiate prin îngroșare în Figura 8.2.8. Conform (8.2.1) rezultă că

$$(S, T) = (\{1, 4\}, \{2, 3, 5, 6\})$$

este o secțiune de capacitate minimă în R .

Observația 8.2.7. Algoritmul Ford-Fulkerson este specific **metodei de programare Greedy**. Pentru o rețea cu n noduri și m arce având toate capacitățile numere întregi și pentru un flux inițial având toate componentele numere întregi, de exemplu fluxul nul, conform demonstrației Teoremei 8.2.2 rezultă că numărul de lanțuri de creștere necesare măririi succesive a fluxului inițial, până se ajunge la un flux de valoare maximă, este de cel mult $(n - 1)c_{max}$, unde c_{max} este capacitatea maximă a arcelor rețelei. Pentru fiecare flux curent, algoritmul necesită calculul reziduurilor celor m arce, parcurgerea acestora pentru depistarea unui eventual lanț de creștere, determinarea acestui lanț (de lungime cel mult m) și mărirea fluxului curent de-a lungul lanțului de creștere, operații având complexitatea $O(m)$. Rezultă că Algoritmul Ford-Fulkerson are complexitatea $O(mnc_{max})$.

Observația 8.2.8. Pentru o rețea având toate capacitățile numere raționale, putem aplica Algoritmul Ford-Fulkerson astfel:

- se înmulțesc toate capacitățile arcelor cu numitorul lor comun M , obținându-se o rețea având toate capacitățile numere întregi;
- se determină un flux de valoare maximă în această rețea, aplicând Algoritmul Ford-Fulkerson (pentru un flux inițial având toate componentele numere întregi, de exemplu fluxul nul);

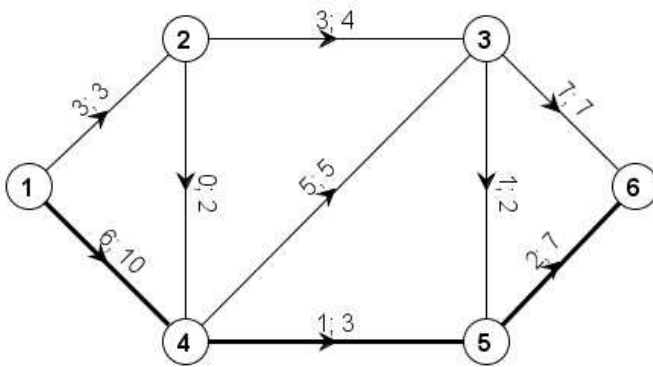


Figura 8.2.7:

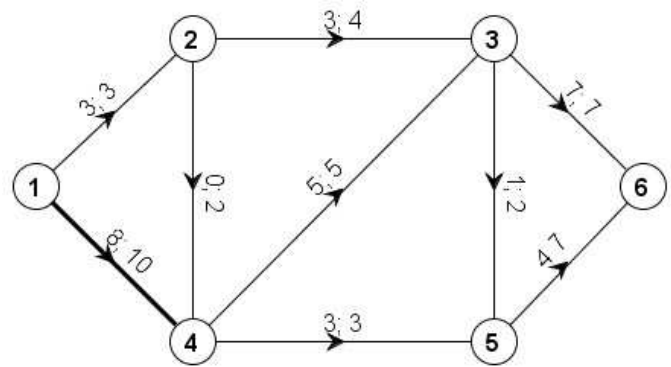


Figura 8.2.8:

- se împart toate componentele acestui flux prin M , obținându-se un flux de valoare maximă în rețeaua inițială.

Evident, toate componentele acestui flux maxim și valoarea lui sunt numere raționale.

Observația 8.2.9. Pentru o rețea având și capacități numere iraționale, Algoritmul Ford-Fulkerson poate să fie inoperabil, fiind posibil ca numărul de pași ai algoritmului (numărul de lanțuri de creștere succesive construite) să fie infinit, deoarece valorile reziduale ale lanțurilor de creștere succesive construite pot forma o serie convergentă.

Totuși, prin impunerea unor modalități judicioase de construire a lanțurilor de creștere succesive în cadrul Algoritmului Ford-Fulkerson, se obțin algoritmi care rezolvă problema fluxului maxim și în cazul capacităților iraționale. Un astfel de algoritm, bazat pe parcurgerea în lățime (BF) a grafului rezidual, este **Algoritmul Edmonds-Karp**.