Tema 3 - Grafuri şi arbori rădăcină

1. **Graf orientat.** Se consideră un graf orientat $G = (N, A)$ memorat prin liste de adiacenţă.

   a. Să se scrie o funcţie care verifică pentru orice vârfuri $x, y \in N$ dacă există drum de la x la y şi afişează acest drum (dacă există). (1p)

   b. Să se scrie o funcţie care verifică dacă există lanţ de la x la y şi în caz afirmativ afişează un astfel de lanţ.(2p).

   c. Să se scrie o funcţie care retrunează numărul de componente conexe ale grafului şi pentru fiecare dintre acestea afişează nodurile sale. (1p)

   Se cere utilizarea unei structuri (sau clase) GRAF. NU cu recursivitate!!!!

   Exemplu: Se consideră graful din figura 1 cu listele de adiacenţă corespunzătoare.

   a. Între nodurile 0 şi 4 există drumul $\{(0,2),(2,7),(7,4)\}$. Între nodurile 5 şi 1 există drumul $\{(5,3),(3,1)\}$.

   b. Între nodurile 4 şi 1 nu există nici un drum.Între nodurile 4 şi 0 există lanţ, de exemplu: $\{(4,2),(0,2)\}$.

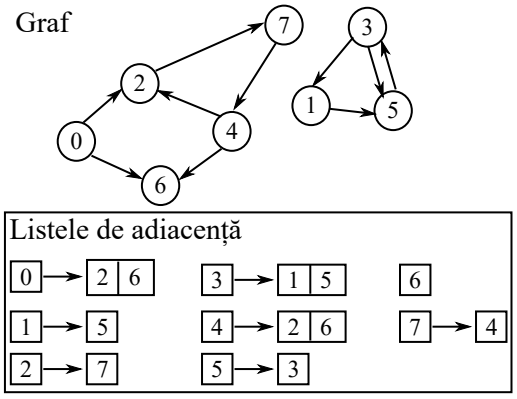


Figure 1: Graf orientat cu reprezentarea cu liste de adiacenţă

c. Graful are două componente conexe. Prima conţine vârfurile $\{0, 2, 4, 6, 7\}$, a doua conţine vârfurile $\{1, 3, 5\}$

2. **Graf neorientat.** Se consideră un graf neorientat conex $G = (N, A)$ memorat prin liste de adiacenţă. Pentru un vârf $v \in N$ vom numi *excentricitate* a lui v lungimea celui mai scurt drum de la v la vârful cel mai depărtat de v. Vom numi *diametru* al grafului excentricitatea maximă a vârfurilor din graf. Se consideră *raza* grafului cea mai mică excentricitate a unui vârf din G. Să se implementeze funcţia excentricitate şi să se determine diametrul şi raza grafului. Dacă graful nu este conex, se semnalează eroare. (2p)
NU cu recursivitate!!! **Exemplu:** Pentru graful din imaginea 2, excentricitatea nodului 0 este 4, excentricitatea nodului 2 este 3, excentricitatea nodului 5 este 2. Diametrul este 4 şi raza este 2.

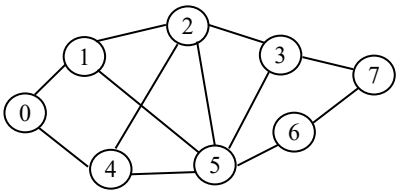


Figure 2: Graf neorientat

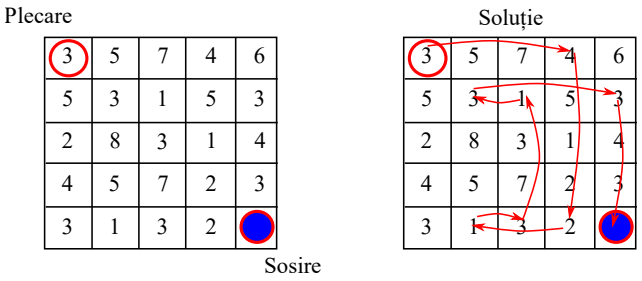


Figure 3: Joc numere exemplu de rezolvare

3. **Joc numere**. Se consideră o matrice $n \times n$ de numere întregi pozitive. Un personaj pleacă din colţul stânga-sus şi trebuie să ajungă în colţul drepta jos. La fiecare mutare personajul poate să se deplaseze în sus, în jos, în stânga sau în dreapta cu exact atâtea poziţii câte indică numărul din căsuţa pe care se află la momentul curent. Nu are voie însă să depăşească marginile matricii.
Să se determine numărul minim de mutări necesare pentru a ajunge din colţul stânga sus în colţul dreapta jos. Dacă nu este posibil, să se semnaleze acest lucru.Un exemplu este prezentat în figura 3. [1]

[1] problemă preluată din "Algorithms" de Jeff Erickson, http://jeffe.cs.illinois.edu/teaching/algorithms/#book

NU cu recursivitate!!!

Variantă: se citesc punctul de start şi punctul de sosire de la tastatură. (2p)

4. **Arbore sintactic.** Se citeşte din fişier o expresie aritmetică formată din numere, operatorii de bază (+, -, *, /) şi paranteze. Să se construiască un arbore sintactic corespunzător expresiei. Să se afişeze arborele pe niveluri. **Observaţie**: într-un astfel de arbore sintactic, care este arbore binar, nodurile interne conţin operatorii aritmetici, iar frunzele conţin operanzii. O operaţie dintr-un nod intern se efectuează între valorile reprezentate de rezultatele subarborilor copii ai acestui nod. Arborele sintactic permite evaluarea expresiei aritmetice în manieră *bottom-up*. Acest lucru înseamnă că operatorii care se vor aplica mai întâi se află pe niveluri mai mari decât cei care se vor aplica mai apoi. Un exemplu este prezentat în figura 4 a.(2p)
Exemplu: Pentru expresia aritmetică 4+2*((5-1+2)*3 +2*(3-1)) arboerele corespunzător este cel din figura 4b. Indicaţie: se poate folosi forma poloneză postfixată pentru construcţia arborelui.

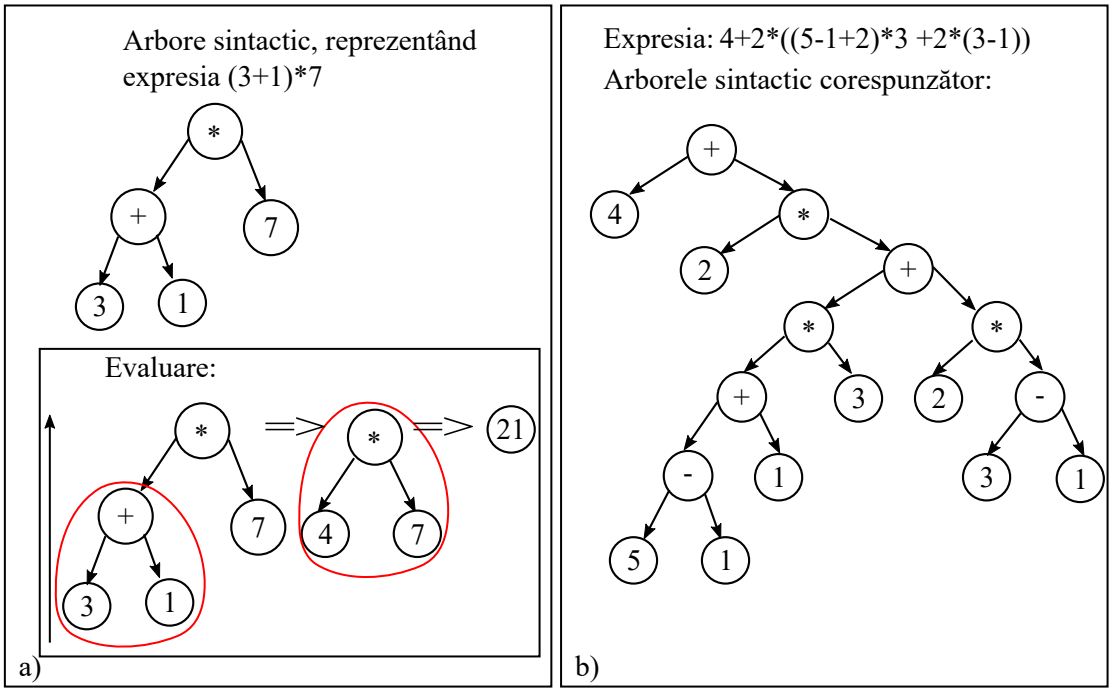
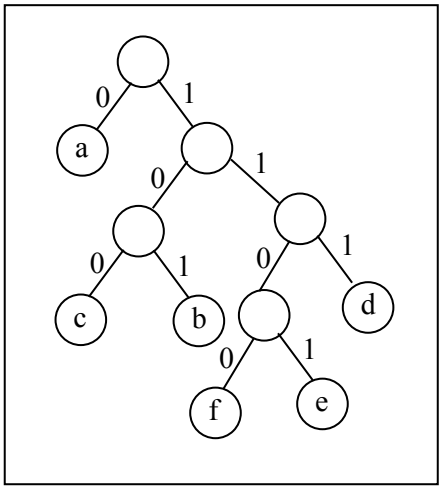


Figure 4: a) Exemplu de arbore sintactic pentru o expresie aritmetică simplă împreună cu modul de evaluare. b) Expresia aritemetică împreună cu arborele sintactic corespunzător.

5. **Arbore Huffman.** Se citesc dintr-un fişier n caractere împreună cu codul lor asociat şi anume: fiecare caracter va avea asociat un cod formar din cifre

binare. Codurile vor avea lungime variabilă şi niciun cod nu este prefixul altuia. Să se construiască un arbore binar asociat în modul următor. Fiecare caracter va fi inserat într-o frunză a arborelui coborând în arbore de la rădăcină, pentru un caracter 0 pe stânga şi pentru un caracter 1 pe dreapta nodului curent. (3p)
ATENŢIE: cerinţa NU este de a determina codurile! (aceasta va fi o problema la tema 4) ci de a construi arborele, atunci când sunt date codurile!

Exemplu: Se consideră perechile caracter - cod asociat: $a-0, b-101, c-100, d-111, e-1101, f-1100$. Arborele asociat va fi cel din figură



6. **Refacere arbore din parcurgeri.** Să se refacă un arbore binar cunoscându-se parcurgerile sale în preordine şi inordine. Se cere utilizarea unei structuri NOD pentru nodurile arborelui, care sa aibă 3 câmpuri: INFO - informaţie, STANGA - pointer la NOD, DREAPTA - pointer la NOD. Pentru varianta recursivă - 1p. Pentru varianta nerecursivă - 3p.

7. **Arbore genealogic.** Se consideră arborele genealogic al unei familii. Construiţi o structură corespunzătoare, care să conţină: (2p)

   - o funcţie de citire a arborelui din fişier
   - o funcţie de afişare pe niveluri, astfel încât la afişare nivelurile să fie separate prin 2 rânduri libere şi pentru fiecare persoană să se afişeze numele său, numele părintelui şi numărul de copii. (Pentru o afisare grafică a arborelui se oferă suplimentar 2p).
   - o funcţie, care are ca parametru un nume şi afişază numele tatălui şi numele copiilor perosanei respective, dacă aceasta se găseşte în arbore.

Altfel semnalează faptul, că nu a fost găsită persoana. (ce complexitate medie are?)

- o funcţie care afişază numărul de generaţii reprezentate în arbore