Programarea Algoritmilor
– LABORATOR NR. 7 –

Tehnica de programare Programare Dinamică

1. Se dă o scară cu *n* trepte (*n* citit de la tastatură). Un om aflat la baza scării (pe treapta 0) poate într-un pas să sară cate 1, 2 sau 3 trepte. În câte moduri poate ajunge pe treapta *n* pornind de pe treapta 0?
Exemplu: n=3 => 4 moduri (câte 1+1+1 sau 1+2 sau 2+1 sau 3 trepte sărite)

2. *(Conjuctura lui Collatz)* Se dă o funcție f : N -> N definită astfel:
f(n) = n / 2, dacă n este par și
3 * n + 1, dacă n este impar
Pentru orice număr natural, dacă se aplică în mod repetat funcția f se ajunge la valoarea 1.
Pentru un număr n dat, să se afișeze după câți pași se ajunge prima dată la valoarea 1.
Exemplu: n=20 => 7 pași (20 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1)

3. Subsecvența de sumă maximă a unui șir: Se dă un șir de numere (în fișier, separate prin spații). Să se afișeze o subsecvență de sumă maximă a șirului (formată cu elemente consecutive).

date.in	**date.out**
1 -2 3 -1 5 2 -6 1 3	3 -1 5 2

Indicație:
Subproblema s[i] = subsecvența de sumă maximă care se termină pe poziția i
Recurența s[i] = max{s[i-1]+v[i], v[i]} (unde v este șirul dat) – subsecvența care se termină pe poziția i este formată din elementul v[i] la care se adaugă eventual subsevența care se termină pe poziția i-1 (dacă astfel se obține o sumă mai mare).

4. Dat un șir de cuvinte formate cu litere mici, să se determine cel mai lung subșir al său astfel încât pentru orice două cuvinte consecutive din subșir ultimele două litere din primul să coincidă cu primele două litere din cel de al doilea.
Exemplu: Pentru șirul: seara, carte, teorema, temperatura, rar, mare, arbore
cel mai lung subşir care verifică cerinţele este - carte, temperatura, rar, arbore

date.in	**date.out**
masa carte sac teatru tema rustic sare	carte teatru rustic

5. Se dau două cuvinte **s** și **t**. Să se găsească un subșir comun de lungime maximă.
Exemplu: s = "S**U**B**SI**R", t = "R**UST**ICE" => "**USI**"

Ideea de rezolvare folosind programarea dinamică este să calculăm o matrice **M** având cu o linie în plus față de lungimea cuvântului **s** și cu o coloană în plus față de lungimea cuvântului **t**, unde **M[i][j]** reprezintă lungimea maximă a unui subșir comun pentru prefixul de lungime **i** al lui **s** și prefixul de lungime **j** al lui **t**.

6. Date o mulțime de n numere naturale și un număr natural M, M < 10000, să se determine, dacă există, o submulțime a mulțimii date de sumă M.

date.in	**date.out**
6 12 1 3 4 5 7 14	3 4 7

7. Moș Crăciun a poposit la bradul a doi frați, unde și-a golit sacul. Când s-au trezit, frații au intrat într-o mare dilemă: cum își vor împărți ei cadourile? Știind că fiecare cadou are o valoare cuprinsă între 1 și 100 și că sunt maxim 100 de cadouri, scrieți un program care să determine sumele cadourilor fraților precum și modul de împărțire, astfel încât sumele obținute să fie cele mai apropiate posibil.
Exemplu: pentru 7 cadouri cu valorile 28, 7, 11, 8, 9, 7, 27 sumele sunt 48 și 49, o împărțire a cadourilor fiind 28, 11, 9, respectiv 7, 8, 7, 27.
(***Indicație:***
problema se reduce la a **determina o submulțime de sumă maximă, care nu depășește însă valoarea M**=jumătate din suma valorilor) **O(nS),** cu S=valoarea totală a cadourilor.

8. Se consideră o tablă de șah nxm (n, m date). Pe fiecare careu al tablei este plasat câte un obiect, fiecare cu o anumită valoare (cunoscută, număr natural). Pe tablă se deplasează un robot astfel: pornește de pe prima linie și prima coloană (un colț al tablei) și se poate deplasa numai în direcțiile **sud** și **est**. La parcurgerea unei celule robotul adună obiectul din celulă. Să se determine un traseu al robotului până în poziția (n, m) (până în colțul opus celui din care a plecat) astfel încât valoarea totală a obiectelor adunate să fie maximă. Se vor afişa valoarea totală obţinută şi traseul optim.

date.in	**date.out**
3 3 **2 1** 4 1 **3** 2 1 **6 1**	13 1 1 1 2 2 2 3 2 3 3

9. Se consideră un șir de n cuburi colorate (n dat), pentru fiecare cub cunoscându-se lungimea laturii și culoarea sa, codificată cu un număr de la 1 la p (p dat). Să se determine un turn de înălțime maximă în care un cub nu poate fi așezat peste un cub de aceeași culoare sau cu latură mai mică sau egală cu a sa. Afișați și câte astfel de turnuri există. Structura fișierului de intrare: pe prima linie sunt n și p, pe următoarele linii lungimea laturii și culoarea câte unui cub. **O(n^2)**

date.in	**date.out**
7 3 8 3 10 2 9 2 10 1 8 1 5 2 6 2	10 1 9 2 8 1 6 2 2