

## Metoda Programării Dinamice

**Problemele trebuie rezolvate folosind metoda programării dinamice.**

**Pentru fiecare problemă trebuie justificată relația de recurență obținută (pornind de la principiul de optimalitate, evidențiind subproblemele, relațiile de recurență și ordinea de calcul)**

1. 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)  $O(n)$

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

### Indicație:

Subproblema  $s[i]$  = subsecvența de suma maximă care se termina pe poziția  $i$

Recurența  $s[i] = \max\{s[i-1]+v[i], v[i]\}$  (unde  $v$  este șirul dat) – subsecvența care se termina pe poziția  $i$  este formată din elementul  $v[i]$  la care se adaugă eventual subsecvența care se termina pe poziția  $i-1$  (daca astfel se obține o sumă mai mare)

2. Se consideră o tablă de șah  $n \times m$  ( $n, m$  date). Pe fiecare carucel 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 afișa valoarea totală obținută și traseul optim  $O(nm)$

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

3. 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	10 1
8 3	9 2
10 2	8 1
9 2	6 2
10 1	2
8 1	
5 2	
6 2	

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 subșir care verifică cerințele este - carte, temperatura, rar, arbore  $O(n^2) / O(n)$

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

5. 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  $M$   **$O(nM)$**

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

6. 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 moșului? 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 tuturor elementelor)  **$O(nS)$** , unde  $S$ =valoarea totală a cadourilor
7. **Problema monedelor** Având la dispoziție un număr nelimitat de monede de valori date  $\{v_1, v_2, \dots, v_n\}$ , să se determine o modalitate de a plăti o sumă de bani  $S$  dată folosind un număr minim de astfel de monede (dacă suma se poate plăti). Exemplu: pentru monede de valori  $\{7, 5, 1\}$  și  $S = 11$  se vor da 3 monede: două monede de valoare 5, o monedă de valoare 1.  **$O(nS)$**
8. **Generalizarea problemei spectacolelor (planificării activităților) discutată la curs la Greedy.** Se dau  $n$  activități prin timpul de început, timpul de sfârșit și profitul asociat desfășurării activității ( $n$  intervale închise cu extremități numere reale care au asociate ponderi). Să se determine o submulțime de activități compatibile (intervale disjuncte două câte două) care au profitul total maxim. Se vor afișa profitul total și activitățile  **$O(n^2)/O(n \log n)$**

Jon Kleinberg, Éva Tardos, *Algorithm Design*, Addison-Wesley 2005

<https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/06DynamicProgrammingI.pdf>

date.in	date.out
4 1 3 1 2 6 8 4 7 2 10 11 5	13 2 6 10 11

9. (suplimentar) **Generalizarea problemei Maximizarea profitului cu respectarea termenelor limită de la Greedy (scheduling jobs with deadlines profits and durations).** Același enunț, dar pentru o activitate se cunoaște în plus și durata acesteia  $l_i$  (se renunță la ipoteza toate activitățile au aceeași durată și la faptul că  $1 \leq t_i \leq n$ ) -  **$O(nT + n \log(n))$** , unde  $T = \max\{t_i | i=1, n\}$ .

**Exemplu.** Pentru  $n = 4$  și

$p_1 = 3, t_1 = 5, l_1 = 3$

$p_2 = 2, t_2 = 2, l_2 = 1$

$p_3 = 3, t_3 = 2, l_3 = 2$

$p_4 = 5, t_4 = 4, l_4 = 3$

o soluție optimă se obține dacă planificăm activitățile în ordinea 2, 4, profitul fiind 7

**Observație:** Problema discretă a rucsacului poate fi privită ca un caz particular al acestei probleme (obiectele sunt activități de durată  $g_i$ , profit  $c_i$  și termen limită  $G$ )

date.in	date.out
4 3 5 3	7 2 4

2 2 1	
3 2 2	
5 4 3	

10. (Suplimentar) **Generalizarea problemei joc de la Greedy** Pentru jocul descris în problema 10 de la Greedy, renunțând la ipoteza că numărul de elemente  $n$  este par, determinați dacă primul jucător are o strategie de câștig și, în caz afirmativ, **cu cât va câștiga minim** (cu cât va fi mai mare sigur suma lui decât a adversarului). Implementați un joc de două persoane în care pentru primul jucător mută calculatorul conform strategiei optime determinate, iar pentru al doilea joacă utilizatorul. La fiecare pas anunțați-l pe utilizator cu cât sunteți sigur că va fi mai mare suma obținută de calculator față de a sa  $O(n^2)$  (vezi și <http://www.infoarena.ro/problema/joculet>)

11. (suplimentar) Se dă o matrice  $n \times m$  cu elementele în mulțimea  $\{0, 1\}$ .

- a) Afișați dimensiunea și coordonatele colțului din stânga sus ale unui pătrat de arie maximă din matrice care conține doar elemente egale cu 0 (submatrice pătratică având toate elementele egale cu 0, **inclusiv cele din interior**).
- b) Dat un număr  $k$  (pe ultima linie a fișierului de intrare) determinați numărul de pătrate din matrice de latură mai mare sau egală cu  $k$  care conțin doar elemente egale cu 0.  $O(nm)$  (2,5p)

date.in	date.out
4 5 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 2	3 2 3 6 <i>matricea conține 5 pătrate cu elementele 0 de latură 2 și 1 pătrat de latură 3, deci 6 pătrate în total</i>

12. (suplimentar) **Distanța Levenstein** [https://en.wikipedia.org/wiki/Levenshtein\\_distance](https://en.wikipedia.org/wiki/Levenshtein_distance) Se dau două cuvinte a și b. Asupra primului cuvânt putem efectua următoarele 3 operații:

- inserare: se adaugă în cuvânt un caracter pe o poziție (oarecare) - cu costul  $c_1$
- ștergere: se șterge o literă din cuvânt (de pe o poziție, nu toate aparițiile) - cu costul  $c_2$
- înlocuire: se înlocuiește o literă de pe o poziție din cuvânt cu altă literă - cu costul  $c_3$

Costurile  $c_1$ ,  $c_2$  și  $c_3$  sunt date de intrare. Distanța de editare a celor două cuvinte este costul minim al unui șir de operații care trebuie aplicate asupra primului cuvânt pentru a îl transforma în cel de-al doilea (dacă  $c_1=c_2=c_3$ , atunci distanța de editare este chiar numărul minim de operații care trebuie aplicate asupra primului cuvânt pentru a îl transforma în cel de-al doilea). Același tip de operație poate fi aplicat de mai multe ori. Să se determine distanța de editare a celor două cuvinte; **se vor afișa și operațiile care trebuie efectuate asupra primului cuvânt pentru a îl obține pe al doilea. Exemplu:** pentru cuvintele carte și antet, dacă  $c_1=c_2=c_3=1$ , distanța de editare este 3, operațiile efectuate asupra primului cuvânt fiind: ștergem litera c (de pe poziția 1), înlocuim litera r (de pe poziția 3) cu n și adăugăm la sfârșit litera t (v. <http://www.infoarena.ro/problema/edist>)  $O(nm)$   $n=\text{lungime}(a)$ ,  $m=\text{lungime}(b)$

date.in	date.out
carte antet 1 1 1	3 stergem c pastram a inlocuim r-n pastram t pastram e inseram t