


# Programarea Algoritmilor

– LABORATOR NR. 7 –

## Programare Dinamică

1. Se consideră un șir de  $n$  piese de domino. O piesă de domino are formă dreptunghiulară și are înscrisă pe ea două numere.  Conform regulilor la domino, un lanț de piese este un subșir al șirului de piese inițial constituit din piese care respectă următoarea condiție: pentru oricare două piese consecutive din lanț, al doilea număr înscris pe prima din cele două piese coincide cu primul număr înscris pe cea de a doua piesă (piesele nu se pot roti). Se citesc din fișierul **date.in** un număr natural  $n$  și un șir de  $n$  perechi ordonate de numere reprezentând piese de domino.

a) Să se determine un lanț de lungime maximă care se poate obține cu piesele din șirul dat (adică un cel mai lung subșir de perechi cu proprietatea că, pentru oricare două perechi consecutive din subșir, ultimul număr din prima pereche coincide cu primul număr din cea de a doua pereche).

b) Determinați câte astfel de subșiruri de lungime maximă există.  $O(n^2)$  (\*\*)

date.in	date.out (un exemplu, soluția nu este unică)
7	1 5
1 8	5 2
1 5	2 4
5 3	2
5 2	
4 8	
2 4	
2 3	

(un alt subșir de lungime maximă este (1,5), (5,2), (2,3) )

2. Se consideră o tablă de șah  $n \times m$  ( $n, m$  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 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. Plata unei sume folosind un număr minim de monede cu valori date.

**Exemplu:** suma = 14, monede = [5, 3, 2]

=> răspuns: 4 monede

$$14 = 5 + 5 + 2 + 2$$

**Indicație de rezolvare:** pentru fiecare valoare de la 0 la suma, calculăm care este numărul minim de monede necesar și reținem care este valoarea ultimei (cele mai mari) monede folosite.

4. Având un șir de numere naturale, să se găsească un subșir crescător de lungime maximă.

**Exemplu:**  $V = (5, 1, 7, 3, 7, 8, 4, 9, 2) \Rightarrow \text{sol} = (1, 7, 7, 8, 9)$  (nu e unică)

5. Având două șiruri de litere, să se găsească subșirul comun de lungime maximă.

**Exemplu:**  $s = \text{SUBSIR}, t = \text{RUSTICE} \Rightarrow \text{sol} = \text{USI}$

6. Problema rucsacului varianta discretă 0/1

Se dau  $n$  obiecte, fiecare obiect  $i$  având asociat un câștig  $c_i$  și o greutate  $g_i$  număr natural. Știind  $G$  capacitatea maximă a rucsacului, să se găsească o variantă de a încărca rucsacul cu obiecte întregi astfel încât câștigul acelor obiecte să fie maxim.

**Indicație de rezolvare:**

Se folosește o matrice  $c_{\max}[i][j]$  = câștigul maxim care se poate obține din primele  $i$  obiecte în limita a  $j$  kg.

1. **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	3
antet	stergem c
1	pastram a
1	inlocuim r-n
1	pastram t
	pastram e
	inseram t

2. Se dau  $n$  vectori de numere naturale nenule și un număr natural  $k$ . Să se construiască un șir de  $n$  numere cu următoarele proprietăți:

- Al  $i$ -lea element al șirului este ales din vectorul  $i$ .
- Suma elementelor șirului este egală cu  $k$ .

Datele de intrare se citesc dintr-un fișier. Prima linie va conține numerele  $n$  și  $k$ . Pe fiecare din următoarele  $n$  linii sunt scrise elementele câte unui vector, separate prin spații.

Se vor afișa elementele unui șir construit cu restricțiile de mai sus. Dacă nu există un șir cu proprietățile cerute se va afișa 0. **O(mk)**,  $m=\text{numărul total de elemente din șiruri}$  – licență 2014 (\*\*\*)

date.in	date.out
3 11	5 4 2
3 5 10 8	
4 3 7	
6 8 2 9	

3. **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

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

4. Se citesc din fișierul **date.in**  $m$  șiruri binare (care conțin doar 0 și 1), la care se adaugă șirul "0" și șirul "1". Dat un fișier **cod.in** cu o singură linie care conține doar 0 și 1, să se descompună conținutul fișierului (șirul binar din fișier) într-un număr **minim** de șiruri binare dintre cele  $m+2$  (astfel, pentru a memora șirul se pot memora doar indicii șirurilor binare în care se descompune).  $O(n^2m)$  unde  $n$ =numărul de caractere din fișierul **cod.in** (\*\*\*)

date.in	cod.in	date.out
3	010010011	010+010+01+1
01		
010		
1001		

5. **Alinierea secvențelor** (v. și curs) Se citesc două cuvinte (secvențe) de lungimi  $n$ , respectiv  $m$ , peste un alfabet (spre exemplu, cuvinte peste alfabetul  $\{A,C,G,T\} \Rightarrow$  secvențe ADN). Alinierea a două secvențe reprezintă punerea pozițiilor (caracterelor) din cele două secvențe în corespondență 1 la 1, cu posibilitatea de a insera spații în ambele cuvinte. Astfel, vom alinia secvențele inserând în ele caracterul " " pentru ca secvențele să devină de aceeași lungime și penalizând pozițiile pe care diferă secvențele obținute. Scorul alinierii este dat de suma dintre penalizarea alinierii unui caracter cu un spațiu și penalizările pentru alinieri de litere diferite. Date două cuvinte, penalizarea inserării unui spațiu (= penalizarea alinierii unui caracter cu un spațiu) și, pentru fiecare pereche de litere  $X$  și  $Y$  din alfabet, penalizarea pentru potrivirea (alinierii) literei  $X$  cu  $Y$ , să se afișeze alinierea cu scor minim a celor două cuvinte și scorul acestei penalizări.  $O(nm)$  (\*\*\*)

**Exemplu:** pentru secvențele GATC și TCAG, dacă penalizarea pentru spațiu este 2, penalizarea pentru alinierea A-C sau G-T este 1, iar pentru celelalte alinieri este 3, scorul minim este 6 pentru alinierea

G-ATC  
TCAG-

date.in	date.out
GATC	6
TCAG	G-ATC
	TCAG-

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

6. **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	7
3 5 3	2 4
2 2 1	
3 2 2	
5 4 3	

7. Se dă un cuvânt formate numai cu litere.

a) Să se determine câte palindromuri (subsecvențe egale cu inversele lor) conține cuvântul  **$O(n^2)$**

b) Să se descompună șirul în număr minim de palindromuri. Exemplu: Pentru abcbaabc – se obțin 3 palindromuri: a, b, cbaabc; pentru aaacaaba se obțin 3 palindromuri: aa, aca, aba.

<https://leetcode.com/problems/palindrome-partitioning-ii/>  **$O(n^2)$**  (\*\*\*\*)

8. **Generalizarea problemei 10 (joc pentru două persoane) 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>) (\*\*\*\*).