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 şir: Se dă un şir de numere (în fişier, separate prin spaţii). Să se afişeze o subsecvenţă de sumă maximă a şirului (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] = subsecventa de suma maxima care se termina pe pozitia i
Recurenta s[i] = max{s[i-1]+v[i], v[i]} (unde v este sirul dat) – subsecventa care se termina pe pozitia i este formata din elemental v[i] la care se adauga eventual subseventa care se termina pe pozitia i-1 (daca astfel se obtine o suma mai mare)

2. Se consideră o tablă de șah nxm (n,m date). Pe fiecare careul 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 **O(nm) – v. seminar**

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

3. Se consideră un şir de n cuburi colorate (n dat), pentru fiecare cub cunoscându-se lungimea laturii şi 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) – v. seminar**

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

4. Dat un şir de cuvinte formate cu litere mici, să se determine cel mai lung subşir al său astfel încât pentru orice două cuvinte consecutive din subşir ultimele două litere din primul să coincidă cu primele două litere din cel de al doilea. **Exemplu**: Pentru şirul: seara, carte, teorema, temperatura, rar, mare, arbore
cel mai lung subşir care verifică cerinţele 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. **Distanţa Levenstein** https://en.wikipedia.org/wiki/Levenshtein_distance Se dau două cuvinte a şi b. Asupra primului cuvânt putem efectua următoarele 3 operaţii:
   - inserare: se adaugă în cuvânt un caracter pe o poziţie (oarecare) - cu costul c1
   - ştergere: se şterge o literă din cuvânt (de pe o poziţie, nu toate apariţiile) - cu costul c2
   - înlocuire: se înlocuieşte o literă de pe o poziţie din cuvânt cu altă literă - cu costul c3

Costurile c1, c2 şi c3 sunt date de intrare. Distanţa de editare a celor două cuvinte este costul minim al unui şir de operaţii care trebuie aplicate asupra primului cuvânt pentru a îl transforma în cel de-al doilea (dacă c1=c2=c3, atunci distanţa de editare este chiar numărul minim de operaţii care trebuie aplicate asupra primului cuvânt pentru a îl transforma în cel de-al doilea). Acelaşi tip de operaţie poate fi aplicat de mai multe ori. Să se determine distanţa de editare a celor două cuvinte; **se vor afişa şi operaţiile care trebuie efectuate asupra primului cuvânt pentru a îl obţine pe al doilea**. **Exemplu:** pentru cuvintele carte şi antet, dacă c1=c2=c3=1, distanţa de editare este 3, operaţiile efectuate asupra primului cuvânt fiind: ştergem litera c (de pe poziţia 1), înlocuim litera r (de pe poziţia 3) cu n şi adăugam la sfârşit litera t (v. http://www.infoarena.ro/problema/edist) **O(nm)** n=lungime(a), m=lungime(b) (**2,5p**)

`date.in`	`date.out`
carte antet 1 1 1	3 stergem c pastram a inlocuim r-n pastram t pastram e inseram 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 **O(nM)**

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

7. Moş 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

8. **Problema monedelor** Având la dispoziție un număr nelimitat de monede de valori date {v1,v2,...,vn}, 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)**

9. **Generalizarea problemei spectacolelor (planificării activităților) discutată la curs la Greedy.** Se dau n activități prin timpul de început, timpul de sfârşit şi profitul asociat desfăşurării 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 afişa profitul total şi activităţile **O(n^2)/O(nlog 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

10. **(suplimentar) Generalizarea problemei *Maximizarea profitului cu respectarea termenelor limită* de la Greedy (*scheduling jobs with deadlines profits and durations*).** Acelaşi enunţ, dar pentru o activitate se cunoaşte în plus şi durata acesteia l_i (se renunţă la ipoteza toate activităţile au aceeaşi durată şi la faptul că $1 \le t_i \le n$) - **O(nT+nlog(n)),** unde T=max{ti|i=1,n}.

Exemplu. Pentru n = 4 şi

p1 = 3, t1 = 5, l_1 = 3

p2 = 2, t2 = 2, l_2 = 1

p3 = 3, t3 = 2, l_3 = 2

p4 = 5, t4 = 4, l_4 = 3

o soluţie optimă se obţine dacă planificăm activităţile în ordinea 2, 4, profitul fiind 7

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

date.in	date.out
4 3 5 3 2 2 1 3 2 2 5 4 3	7 2 4

11. **(Suplimentar) Generalizarea problemei joc de la Greedy** Pentru jocul descris în problema 10 de la Greedy, renunţând la ipoteza că numărul de elemente n este par, determinaţi dacă primul jucător are o strategie de câştig şi, în caz afirmativ, **cu cât va câştiga minim** (cu cât va fi mai mare sigur suma lui decât a adversarului). Implementaţi 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 anunţaţi-l pe utilizator cu cât sunteţi sigur că va fi mai mare suma obţinută de calculator faţă de a sa **O(n^2)** (vezi şi http://www.infoarena.ro/problema/joculet)

12. (**suplimentar**) Se dă o matrice nxm cu elementele în mulţimea {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 fişierului de intrare) determinaţi numărul de pătrate din matrice de latură mai mare sau egală cu k care conţin 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 *matricea conţine 5 pătrare cu elementele 0 de latură 2 şi 1 pătrat de latură 3, deci 6 pătrate în total*