Tema 5 - Algoritmi de tip Greedy

1. **Problema continuă a rucsacului:** Avem un rucsac cu care putem transporta o anumită greutate maximă G. Avem n obiecte de transportat şi pentru fiecare obiect se cunoaşte greutatea sa şi beneficiul obţinut de pe urma transportului. Se cere să se determine ce obiecte pot fi transportate odată, astfel încât câştigul să fie maxim.

Se consideră că un obiect poate fi tăiat pentru a umple rucsacul. (2p)

2. Se dau n numere întregi nenule b1, b2, …, bn şi m numere întregi nenule a1, a2, …, am. Presupunând că n ≥ m, Să se determine o submulţime a mulţimii B = {b1, b2, …, bn} care să maximizeze valoarea expresiei

E = a1x1 + a2x2 + … + amxm

unde xi ∈ B, ∀ i∈. Notez cu A = {a1, a2, …, am}. (1p)

3. Într-o sală trebuie planificate într-o zi spectacole. Având n spectacole care, iar pentru fiecare spectacol i se cunoaşte intervalul de desfăşurare [sti, sfi ), se cere să se planifice numărul maxim de spectacole astfel încât să nu se suprapună. (2p)

4. Presupunem că avem n vectori, v1, v2, ..., vn, ordonaţi crescător. Se pune problema interclasării acestora, doi câte doi, până la obţinerea unui şir sortat crescător, care conţine elementele tuturor vectorilor. Această interclasare trebuie realizată cu număr minim de comparaţii. (3p-4p)

***Observaţie****:*

Dacă avem şirurile v1, v2, v3 de lungimi l1 = 30, l2 = 20, l3 = 10, atunci putem interclasa în mai multe moduri:

* Pe v1 cu v2 şi apoi cu v3: numărul de comparaţii / deplasări: (l1 + l2) + (l1 + l2 + l3) = =110
* Pe v2 cu v3 şi apoi cu v1: numărul de comparaţii / deplasări: (l2 + l3) + (l1 + l2 + l3) = 90
* Pe v1 cu v3 şi apoi cu v2: numărul de comparaţii / deplasări: (l1 + l3) + (l1 + l2 + l3) = =100

Se observă că numărul minim se obţine atunci când se interclasează întâi cele mai scurte şiruri.

5. Fie N intervale închise pe axa 0x (de forma [xi, xs]). Să se determine un număr minim de puncte, astfel încât fiecare interval să conţină cel puţin unul dintre acestea. (3p)

6. Trebuie plătită o sumă S. Pentru aceasta se dispune de n tipuri de bancnote, din fiecare bancnotă i, cu valoarea val(i), existând un număr nelimitat de bucăţi la dispoziţie. Cum poate fi plătită suma cu un număr minim de bancnote? (2p)

7. ***Codul Huffmann***

Algoritmul Huffmann este folosit pentru compresia datelor fără pierdere de informaţie. Presupunând că se codifică un text alcătuit din caractere ascii, algoritmul Huffmann utilizează un tablou de frecvenţe, în care se specifică pentru fiecare caracter o valoare care reprezintă frecvenţa de apariţie a acestuia. Lungimea secvenţei de codificare (în biţi) pentru fiecare caracter depinde de frecvenţa acestiua. Cu cât un caracter apare mai frecvent în text, cu atât lungimea codului utilizată pentru el este mai mică.

**Coduri prefix (prefix-codes)**

Un cod prefix este un cod în care nici un element al codului nu reprezintă un prefix pentru al element. Acest lucru permite codificare/decodificare fără ambiguităţi.

**Exemplu:**

Se consideră alfabetul alaătuit din primele 6 litere {a, b, c, d, e, f} şi următorul tabel de codificare:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | a | B | c | d | e | f |
| Frecvenţa (la 1000 caractere în lb. engl.) | 45 | 13 | 12 | 16 | 9 | 5 |
| Codificare cu cod de lungime fixă (3 biţi) – cod1 | 000 | 001 | 010 | 011 | 100 | 101 |
| Codificare cu cod de lungime variabilă – cod2 | 0 | 101 | 100 | 111 | 1101 | 1100 |

Codificarea cu codul 2 00101100110110100 reprezintă 0. 0. 101. 100. 1101. 101.0.0 adică textul aabcebaa

**Arborele Huffman**

Un mod de construcţie a unui astfel de cod prefix de lungime variabilă se realizează cu ajutorului arborelui Huffman. În acest arbore, frunzele reprezintă caracterele de codificat. Arcul de la un nod părinte la descendentul stâng se marchează cu caracterul 0, iar cel către descendentul drept se marchează cu 1.

Codul corespunzător unui caracter dintr-o frunză se obţine prin concatenarea caracterelor care se află pre drumul de la rădăcină către frunza respectivă.

**Observaţie**: un cod optimal se obţine pe baza unui arbore binar în care fiecare nod neterminal are exact doi descendenţi.

**Construcţia arborelui de codificare Huffman:**

Se plasează fiecare caracter într-o frunză şi se plasează într-o listă (vector) de noduri.

Se sortează vectorul de noduri descrescător în funcţie de frecvenţă.

Cât timp în vectorul de noduri se găsesc cel puţin 2 noduri: se extrag ultimele două noduri, se leagă ca descendenţi de către un nou nod părinte, a cărui frecvenţă va fi dată de suma frecvenţelor celor 2 copii. Acest nod nou se plasează în vector pe poziţia corespunzătoare, dată de frecvenţă.

După obţinerea arborelui Huffman se poate construi un look-up table în care fiecărui caracter i se asociază codul obţinut pe baza arborelui. Acest tabel se foloseşte apoi pentru codificare.

***Decodificarea***: pentru a putea decodifica corect documentul, este necesar ca atât cel care codifică, cât şi cel care decodifică să utilizeze acelaşi arbore. Acest lucru se realizează ori prin folosirea de comun acord a aceluiaşi set de caractere cu aceeaşi frecvenţă sau prin transmiterea împreună cu textul codificat şi a arborelui de codificare. În literatură sunt propuse diferite moduri de soluţionare a acestei probleme.

**Problema:** codificarea cu codul Huffmann a unui text. 5p - constructia arborelui, 1p codificarea + 1p decodificarea.