Algoritmi şi structuri de date (I). Seminar 11: Aplicaţii ale tehnicii greedy

Problema 1 *Problema planificării activităţilor (1).* Se consideră un set de n prelucrări ce trebuie executate de către un procesor. Durata execuţiei prelucrărilor este aceeaşi (se consideră egală cu 1). Fiecare prelucrare i are asociat un termen final de execuţie, $t_i \leq n$ şi un profit, p_i. Profitul unei prelucrări intervine în calculul profitului total doar dacă prelucrarea este executată (dacă prelucrarea nu poate fi planificată înainte de termenul final de execuţie profitul este nul). Se cere să se planifice activităţile astfel încât să fie maximizat profitul total (acesta este corelat cu numărul de prelucrări planificate).

Exemplu: considerăm $n = 4$ activităţi având termenele finale: $(2, 3, 4, 2)$ şi profiturile corespunzătoare $(4, 3, 2, 1)$.

Indicaţie. O soluţie constă în stabilirea unui "orar" de execuţie a prelucrărilor $S = (s_1, s_2, \ldots, s_n)$, $s_i \in \{1, \ldots, n\}$ fiind indicele prelucrării planificate la momentul i. Pentru rezolvarea problemei se poate aplica o tehnică de tip greedy caracterizată prin:

- se sortează activităţile în ordinea descrescătoare a profitului;
- fiecare activitate se planifică într-un interval liber cât mai apropiat de termenul final de execuţie.

Se observă că dacă pentru fiecare $i \in \{1, \ldots, n\}$ numărul activităţilor care au termenul final cel mult i este cel mult egal cu i ($\text{card}\{j | t_j \leq i\} \leq i$) atunci toate activităţile vor fi planificate, altfel vor exista activităţi ce nu pot fi planificate.

Să considerăm $n = 4$ activităţi având termenele finale: $(2, 3, 4, 2)$ şi profiturile corespunzătoare $(4, 3, 2, 1)$. Atunci aplicând tehnica greedy se obţine soluţia $(4, 1, 2, 3)$. Dacă însă termenele de execuţie sunt: $(2, 4, 1, 2)$ şi aceleaşi profituri se obţine soluţia $(3, 1, 0, 2)$ iar activitatea 4 nu este planificată.

Problema 2 *Problema planificării activităţilor (2).* Se consideră un set de n activităţi, fiecare dintre ele fiind caracterizată printr-un interval de desfăşurare (activitatea A_i se desfăşoară în intervalul $[s_i, f_i)$) care trebuie planificate folosind cât mai puţine resurse (activităţile pot fi cursuri iar resursele săli). Propuneţi o strategie de alocare a resurselor la activităţi astfel încât aceeaşi resursă să nu fie alocată unor activităţi care se desfăşoară simultan şi numărul de resurse utilizate să fie cât mai mic (resursele sunt considerate identice şi nu există restricţii privind alocarea acestora la activităţi).

Exemplu: Se consideră 10 activităţi având intervalele de desfăşurare conform tabelului de mai jos:

Activitate	Ora start	Ora final
A	9:00	10:30
B	9:00	12:30
C	9:00	10:30
D	11:00	12:30
E	11:00	14:00
F	13:00	14:30
G	13:00	14:00
H	14:00	16:30
I	15:00	16:30
J	14:30	16:30

Indicaţie. Este o problemă de selecţie de activităţi compatibile. Se poate aplica succesiv strategia de selecţie de activităţi compatibile (în ordinea crescătoare a momentului de finalizare). Pentru exemplul de mai sus s-ar selecta prima dată activităţile (A, D, G, H) după care s-ar aplica aceeaşi strategie pentru activităţile rămase $((B, C, E, F, I, J))$ selectând: (C, E, I). Se mai aplică o dată selecţia pentru setul rămas $((B, F, J))$. Toate cele trei activităţi rămase sunt selectate fiind compatibile, deci numărul total de resurse utilizate este 3.

Problema 3 *Problema ordonării task-urilor.* Se consideră un set de task-uri T_1, T_2,..., T_n care trebuie executate pe un procesor. Fiecare task T_i este caracterizat de o durată de execuţie L_i şi de un timp de aşteptare până la finalizare, $F_i = S_i + L_i$ unde S_i este momentul la care este lansat în execuţie task-ul T_i. Presupunând că primul task este planificat la momentul 0 şi că task-urile trebuie executate secvenţial propuneţi o ordine de executîe a acestora $(i_1, i_2, ..., i_n)$ astfel încât timpul mediu de aşteptare, $\frac{1}{n}(F_{i_1} + F_{i_2} + \ldots F_{i_n})$ este minim, în ipoteza că task-urile sunt planificate fără pauze între ele $(S_{i_k} = F_{i_{k-1}})$. Demonstraţi că strategia propusă este optimală.

Exemplu: considerăm un set cu $n = 5$ task-uri având duratele: $(5, 10, 3, 8, 2)$.

Indicaţie. Timpul mediu de aşteptare $(F_{i_1} + F_{i_2} + \ldots F_{i_n})/n$ poate fi rescris ca

$$(L_{i_1} + (L_{i_1} + L_{i_2}) + (L_{i_1} + L_{i_2} + L_{i_3}) + \ldots + (L_{i_1} + L_{i_2} + \ldots + L_{i_n}))/n =$$

$$(nL_{i_1} + (n-1)L_{i_2} + ... + L_{i_n})/n$$

de unde rezultă că valoarea este minimă dacă task-urile se planifică în ordinea crescătoare a duratelor de execuţie.

Problema 4 *Problema alimentării cu energie.* Se consideră un autoturism electric care are o autonomie de L kilometri şi un traseu pe care sunt staţii de alimentare aflate între ele la distanţele d_1, d_2,..., d_n (d_1 reprezintă distanţa de la punctul de pornire până la prima staţie de alimentare, d_i reprezintă distanţa de la staţia $i-1$ la staţia i, iar d_n reprezintă distanţa de la ultima staţie până la destinaţie. Presupunând că distanţa dintre oricare două staţii este mai mică decât L propuneţi o strategie de alimentare (selecţie a staţiilor) astfel încât numărul de opriri să fie cât mai mic. Se presupune că după fiecare alimentare autonomia este de L kilometri.

Exemplu. $L = 50$ iar distanţele sunt $(20, 25, 15, 20, 10, 10, 15, 30, 10)$.

Indicaţie. Se cumulează succesiv distanţele până când $d_1 + d_2 + \ldots + d_{i_1} > L$ (i_1 fiind cel mai mic indice pentru care este îndeplinită condiţia) şi se decide că prima oprire trebuie efectuată la staţia $i_1 - 1$; se continuă procesul calculând suma $d_{i_1} + d_{i_1+1} + \ldots + d_{i_2}$ până când se depăşeşte din nou valoarea L, ş.a.m.d.

Problema 5 *Problema impachetarii.* Se consideră un set de numere $a_1, a_2, \ldots, a_n$ cu proprietatea că $a_i \in (0, C]$. Propuneţi o strategie de grupare în cât mai puţine subseturi (k) astfel încât suma elementelor din fiecare subset să nu depăşească valoarea C.

Exemplu. $C = 10$, $a = (7, 5, 1, 4, 3, 8, 9, 6, 2)$.

Indicaţie. Este cunoscută şi sub numele de "bin-packing problem". Există mai multe variante ce folosesc principiul alegerii greedy însă toate sunt *sub-optimale* (nu garantează obţinerea optimului ci doar a unei soluţii suficient de apropiate de cea optimă). O variantă acceptabilă este: (a) se ordonează descrescător setul iniţial; (b) se parcurge setul ordonat şi se transferă elementul curent în primul subset în care "încape"; dacă nu exista "spaţiu" în nici unul dintre subseturi atunci se iniţiază un nou subset.

Problema 6 *Problema acoperirii.* Se consideră un set de n localităţi şi se pune problema conectării acestora printr-o reţea de drumuri. Presupunând că se cunoaşte costul C_{ij} al construirii unui drum între localităţile i şi j stabiliţi care drumuri ar trebui construite astfel încât să se poată ajunge de la o localitate la oricare alta (eventual trecând prin altă localitate) iar costul total să fie cât mai mic. Demonstraţi că strategia propusă este optimală.

Exemplu. Se consideră $n = 6$ localităţi şi costurile construirii drumurilor dintre ele (∞ înseamnă că nu este posibilă construirea nici unui drum).

	A	B	C	D	E	F
A	$-$	2	∞	∞	6	3
B	2	$-$	6	∞	3	∞
C	∞	6	$-$	5	1	∞
D	∞	∞	5	$-$	∞	4
E	6	3	1	∞	$-$	2
F	3	∞	∞	4	2	$-$

Indicaţie. Se porneşte de la oricare dintre localităţi şi se selectează drumul de cost minim care o uneşte cu altă localitate. Se repetă procesul selectând la fiecare etape drumul de cost minim dintre toate drumurile care conectează o localitate deja selectată cu una care nu a fost încă selectată. Ideea de rezolvare este cea de la algoritmul lui Prim pentru construirea arborilor de acoperire de cost minim.