
Algoritmi și structuri de date (I). Seminar 11: Aplicații ale tehnicii greedy

Problema 1 *Problema planificării activităților (1)*. Se consideră un set de n prelucrări ce trebuie executate de către un procesor. Durata execuției prelucrărilor este aceeași (se consideră egală cu 1). Fiecare prelucrare i are asociat un termen final de execuție, $t_i \leq n$ și 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 execuție profitul este nul). Se cere să se planifice activitățile astfel încât să fie maximizat profitul total (acesta este corelat cu numărul de prelucrări planificate).

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

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

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

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

Să considerăm $n = 4$ activități având termenele finale: $(2, 3, 4, 2)$ și profiturile corespunzătoare $(4, 3, 2, 1)$. Atunci aplicând tehnica greedy se obține soluția $(4, 1, 2, 3)$. Dacă însă termenele de execuție sunt: $(2, 4, 1, 2)$ și aceleași profituri se obține soluția $(3, 1, 0, 2)$ iar activitatea 4 nu este planificată.

Problema 2 *Problema planificării activităților (2)*. Se consideră un set de n activități, fiecare dintre ele fiind caracterizată printr-un interval de desfășurare (activitatea A_i se desfășoară în intervalul $[s_i, f_i)$) care trebuie planificate folosind cât mai puține resurse (activitățile pot fi cursuri iar resursele săli). Propuneți o strategie de alocare a resurselor la activități astfel încât aceeași resursă să nu fie alocată unor activități care se desfășoară simultan și numărul de resurse utilizate să fie cât mai mic (resursele sunt considerate identice și nu există restricții privind alocarea acestora la activități).

Exemplu: Se consideră 10 activități având intervalele de desfășurare 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 |

Indicație. Este o problemă de selecție de activități compatibile. Se poate aplica succesiv strategia de selecție de activități compatibile (în ordinea crescătoare a momentului de finalizare). Pentru exemplul de mai sus s-ar selecta prima dată activitățile (A, D, G, H) după care s-ar aplica aceeași strategie pentru activitățile rămase $((B, C, E, F, I, J))$ selectând: (C, E, I) . Se mai aplică o dată selecția pentru setul rămas $((B, F, J))$. Toate cele trei activități 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, \dots, T_n care trebuie executate pe un procesor. Fiecare task T_i este caracterizat de o durată de execuție L_i și de un timp de așteptare până la finalizare, $F_i = S_i + L_i$ unde S_i este momentul la care este lansat în execuție task-ul T_i . Presupunând că primul task este planificat la momentul 0 și că task-urile trebuie executate secvențial propuneți o ordine de execuție a acestora (i_1, i_2, \dots, i_n) astfel încât timpul mediu de așteptare, $\frac{1}{n}(F_{i_1} + F_{i_2} + \dots + F_{i_n})$ este minim, în ipoteza că task-urile sunt planificate fără pauze între ele ($S_{i_k} = F_{i_{k-1}}$). Demonstrați că strategia propusă este optimală.

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

Indicație. Timpul mediu de așteptare $(F_{i_1} + F_{i_2} + \dots + F_{i_n})/n$ poate fi rescris ca

$$\begin{aligned} & (L_{i_1} + (L_{i_1} + L_{i_2}) + (L_{i_1} + L_{i_2} + L_{i_3}) + \dots + (L_{i_1} + L_{i_2} + \dots + L_{i_n}))/n = \\ & (nL_{i_1} + (n-1)L_{i_2} + \dots + L_{i_n})/n \end{aligned}$$

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

Problema 4 *Problema alimentării cu energie.* Se consideră un autoturism electric care are o autonomie de L kilometri și un traseu pe care sunt stații de alimentare aflate între ele la distanțele d_1, d_2, \dots, d_n (d_1 reprezintă distanța de la punctul de pornire până la prima stație de alimentare, d_i reprezintă distanța de la stația $i-1$ la stația i , iar d_n reprezintă distanța de la ultima stație până la destinație. Presupunând că distanța dintre oricare două stații este mai mică decât L propuneți o strategie de alimentare (selecție a stațiilor) 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 distanțele sunt $(20, 25, 15, 20, 10, 10, 15, 30, 10)$.

Indicație. Se cumulează succesiv distanțele până când $d_1 + d_2 + \dots + d_{i_1} > L$ (i_1 fiind cel mai mic indice pentru care este îndeplinită condiția) și se decide că prima oprire trebuie efectuată la stația $i_1 - 1$; se continuă procesul calculând suma $d_{i_1} + d_{i_1+1} + \dots + d_{i_2}$ până când se depășește din nou valoarea L , ș.a.m.d.

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

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

Indicație. Este cunoscută și sub numele de "bin-packing problem". Există mai multe variante ce folosesc principiul alegerii greedy însă toate sunt *sub-optimale* (nu garantează obținerea optimului ci doar a unei soluții suficient de apropiate de cea optimă). O variantă acceptabilă este: (a) se ordonează descrescător setul inițial; (b) se parcurge setul ordonat și se transferă elementul curent în primul subset în care "încapă"; dacă nu există "spațiu" în nici unul dintre subseturi atunci se inițiază un nou subset.

Problema 6 *Problema acoperirii.* Se consideră un set de n localități și se pune problema conectării acestora printr-o rețea de drumuri. Presupunând că se cunoaște costul C_{ij} al construirii unui drum între localitățile i și j stabiliți 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. Demonstrați că strategia propusă este optimală.

Exemplu. Se consideră $n = 6$ localități și 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 | – |

Indicație. Se pornește de la oricare dintre localități și se selectează drumul de cost minim care o unește cu altă localitate. Se repetă procesul selectând la fiecare etapă 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.