Metoda Greedy

Problemele trebuie însoțite de demonstrarea corectitudinii și justificarea complexității.

0. Implementați problemele discutate la curs

1. **Cuburi**. Se dau n cuburi cu laturile **diferite două câte două**. Fiecare cub are o culoare, codificată cu un număr de la 1 la p (p dat).

a) Să se construiască un turn de înălțime maximă de cuburi în care un cub nu poate fi așezat pe un cub de aceeași culoare sau cu latură mai mică decât a sa – **O(n logn).** Pe prima linie a fișierului de intrare se dau n şi p, iar pe următoarele linii latura şi culoarea fiecărui cub. În fișierul de ieșire se vor afișa înălțimea turnului obținut şi indicele cuburilor folosite (de la bază la vârf)

date.in	date.out
4 2 5 1 10 1 9 1 8 2	23 2 4 1

b) În cazul în care lungimile laturilor cuburilor nu erau diferite mai este valabilă metoda propusă? Justificați.

2. **Problema mulțimii de acoperire**

Fie n intervale închise $I_1 = [a_1, b_1], \dots, I_n = [a_n, b_n]$. Să se determine o mulțime M cu număr minim de elemente astfel încât $\forall k \in \overline{1,n}, \exists x \in M$ astfel încât $x \in I_k = [a_k, b_k]$. Mulțimea M se numește *mulțime de acoperire* a șirului de intervale respectiv.

Problema mai este cunoscută și sub numele de *problema cuielor*: "Se consideră n scânduri, fiecare fiind dată printr-un interval închis de pe axa reală. Să se determine numărul minim de cuie pe care trebuie să le batem astfel încât în fiecare scândură să fie bătut cel puțin un cui. Se consideră faptul că orice cui are o lungime suficient de mare pentru a trece prin oricâte scânduri este nevoie.".

intervale.txt	acoperire.txt
570 670 500 590 600 680 690 840 730 790 700 800 900 930	590 680 790 930

3. **Problema partiționării intervalelor** – Se consideră n intervale închise (interpretare: n cursuri, pentru care se cunosc ora de început şi ora de sfârşit). Se cere să se împartă (partiţioneze) această mulţime de intervale într-un **număr minim de submulţimi** cu proprietatea că oricare două intervale dintr-o submulţime nu se intersectează şi să se afişeze aceste submulţimi (interpretare: să se determine numărul minim de săli necesare pentru a putea programa aceste cursuri în aceeaşi zi şi afişaţi o astfel de programare).

Exemplu: pentru n=3 cursuri, care trebuie să se desfășoare în intervalele: [10, 14], [12, 16], respectiv [17, 18], sunt necesare 2 săli, o programare optimă fiind:

- **Sala 1**: [10, 14] – activitatea 1, [17, 18] – activitatea 3
- **Sala 2**: [12, 16] – activitatea 2

date.in	date.out
3 10 14 12 16 17 18	2 [10, 14] [17,18] [12, 16]

4. **Planificare cu minimizarea întârzierii maxime** – Se consideră o mulţime de n activităţi care trebuie planificate pentru a folosi o aceeaşi resursă. Această resursă poate fi folosită de o singură activitate la un moment dat. Pentru fiecare activitate i se cunosc durata l_i şi termenul limită până la care se poate executa t_i (raportat la ora de început 0). Dorim să planificăm aceste activităţi astfel încât întârzierea fiecărei activităţi să fie cât mai mică. Mai exact, pentru o planificare a acestor activităţi astfel încât activitatea *i* este programată în intervalul de timp $[s_i, f_i)$, definim întârzierea activităţii *i* ca fiind durata cu care a depăşit termenul limită: $\boldsymbol{p_i = \max\{0, f_i - t_i\}}$.

Întârzierea planificării se defineşte ca fiind **maximul întârzierilor activităţilor**:

Exemplu. Pentru n = 3 şi

$l_1 = 1,\ t_1 = 3$

$l_2 = 2,\ t_2 = 2$

$l_3 = 3,\ t_3 = 3$

o soluţie optimă se obţine dacă planificăm activităţile în ordinea 2, 3, 1; astfel:

- activitatea 2 în intervalul [0, 2) – întârziere 0
- activitatea 3 în intervalul [2, 5) – întârziere $5 - t_3 = 5 - 3 = 2$
- activitatea 1 în intervalul [5, 6) – întârziere $6 - t_1 = 6 - 3 = 3$

Întârzierea planificării este max{0, 2, 3}=3 $\boldsymbol{P = \max\{p_1, p_2, ..., p_n\}}$

Să se determine o planificare a activităților date care să aibă întârzierea P minimă. Se vor afişa pentru fiecare activitate intervalul de desfășurare și întârzierea – **O(n log n)**

date.in	**date.out (Solutia nu este unica)**
3 1 3 2 2 3 3	activitatea 2: intervalul [0, 2) intarziere 0 activitatea 3: intervalul [2, 5) intarziere 2 activitatea 1: intervalul [5, 6) intarziere 3 Intarziere planificare 3
3 4 10 5 6 2 4	activitatea 3: intervalul [0, 2) intarziere 0 activitatea 2: intervalul [2, 7) intarziere 1 activitatea 1: intervalul [7, 11) intarziere 1 Intarziere planificare 1

5. Maximizarea profitului cu respectarea termenelor limită Se consideră o mulţime de n activităţi care trebuie planificate pentru a folosi o aceeaşi resursă. Această resursă poate fi folosită de o singură activitate la un moment dat. Toate activităţile au aceeaşi durată (să presupunem 1). Pentru fiecare activitate i se cunosc termenul limită până la care se poate executa t_i (raportat la ora de început 0, $1 \le t_i \le n$) şi profitul p_i care se primeşte dacă activitatea i se execută la timp (cu respectarea termenului limită). Să se determine o submulţime de activităţi care se pot planifica astfel încât profitul total obţinut să fie maxim.

Exemplu. Pentru n = 4 şi

$p_1 = 4,\ t_1 = 3$

$p_2 = 1,\ t_2 = 1$

$p_3 = 2,\ t_3 = 1$

$p_4 = 5,\ t_4 = 3$

o soluţie optimă se obţine dacă planificăm activităţile în ordinea 3, 4, 1, profitul obţinut fiind $p_3 + p_4 + p_1 = 2 + 5 + 4 = 11$. **O(n logn)**

date.in	**date.out**
4 4 3 1 1 2 1 5 3	11 3 4 1

6. Interclasarea optimă a n șiruri ordonate – Se dau lungimile a n șiruri ordonate $L_1, L_2, \ldots, L_n$. Dorim să obținem un șir ordonat crescător care conține toate elementele celor n șiruri inițiale, interclasând succesiv perechi de șiruri. Știind că interclasarea a două șiruri de lungimi A respectiv B necesită A+B deplasări, să se determine o ordine în care trebuie să se realizeze interclasările astfel încât numărul total de deplasări să fie minim – **O(nlogn)**

Exemplu: Pentru șirurile de lungimi $L_1 = 20$, $L_2 = 30$, $L_3 = 20$, $L_4 = 35$

Pentru ordinea de interclasare:

(sirul 1, sirul 2) => sirul 5 de lungime $L_5 = L_1 + L_2 = 50$ (50 de deplasari)

(sirul 5, sirul 3) => sirul 6 de lungime $L_6 = L_5 + L_3 = 70$ (70 de deplasari)

(sirul 6, sirul 4) => sirul 7 de lungime $L_7 = L_6 + L_4 = 105$ (105 de deplasari)

numărul total de deplasări va fi 50+70+105 = 225

(=strategia interclaseaza(interclaseaza(interclaseaza(1,2), 3), 4))

O ordine de interclasare cu număr minim de deplasări ar fi:

(sirul 1, sirul 3) => sirul 5 de lungime $L_5 = L_1 + L_3 = 20+20=40$ (40 de deplasări)

(sirul 2, sirul 4) => sirul 6 de lungime $L_6 = L_2 + L_4 = 30+35=65$ (65 de deplasări)

(sirul 5, sirul 6) => sirul 7 de lungime $L_7 = 105$ cu $L_5 + L_6 = 40+65 = 105$ (105 deplasari)

Numărul total de deplasări va fi: 40 + 65 + 105= 210

(=strategia interclaseaza(interclaseaza(1,3), interclaseaza(2,4)))

7. (suplimentar) Considerăm următorul **joc pentru două persoane**. Tabla de joc este o secvenţă de n numere întregi pozitive, iar cei doi jucători mută alternativ. Când un jucător mută, el selectează un număr ori de la stânga ori de la dreapta secvenţei. Numărul selectat este şters de pe tablă. Jocul se termină când toate numerele au fost selectate. Primul jucător câştigă dacă suma numerelor pe care le-a selectat este **cel puţin egală** cu suma selectată de al doilea jucător. Al doilea jucător joacă cât de bine poate. Primul jucător începe jocul. Ştim că tablă se află la început un număr **par** de elemente n.

a) Să se scrie un program astfel încât, indiferent cum va juca al doilea jucător, primul jucător câştigă. Scrieţi programul astfel încât primul jucător să mute cu ajutorul programului, iar calculatorul să mute aleator de la stânga sau de la dreapta. La ieşire se va scrie suma obţinută de primul jucător, suma obţinută de cea de al doilea şi secvenţele de mutări sub forma unor şiruri cu caracterele S pentru stânga şi D pentru dreapta. **Exemplu**: pentru tabla cu numerele 2 1 4 3 o soluţie câştigătoare este următoarea:

Pasul 1 - primul jucător alege S (valoarea 2); rămân pe tablă 1 4 3

Pasul 2 - calculatorul are două posibilităţi: S (valoarea 1) sau D (valoarea 3).

Pasul 3 – dacă la pasul 2 calculatorul a ales S, atunci primul jucător alege S (valoarea 4);
dacă la pasul 2 calculatorul a ales D, atunci primul jucător alege D (valoarea 4);

Pasul 4 – pe tablă a mai rămas doar o valoare (3 respectiv 1, în funcţie de alegerea de la pasul 2), pe care o alege calculatorul

Astfel, primul jucător a adunat suma 2+4, iar calculatorul suma 1+3 (respectiv 3+1), deci primul jucător a câştigat

b) În cazul în care urma strategiei implementate la a) pentru o secvenţă de numere primul jucător obţine o sumă egală cu cel de al doilea, este posibil ca prin altă strategie totuşi primul jucător să câştige cu o sumă strict mai mare pentru aceeaşi secvenţă? Justificaţi