Metoda Greedy - probleme

1. Se dau mulțimile de numere întregi nenule $A=\{a_1, a_2, \ldots, a_m\}$ și $B=\{b_1,b_2, \ldots, b_n\}$, unde $m \leq n$. Să se determine $B'=\{x_1, x_2, \ldots, x_m\}$ o submulțime a lui B astfel încât suma $a_1 x_1 + a_2 x_2 + \ldots + a_m x_m$ să fie maximă. **Exemplu**: Pentru A ={2, 4, 3} și B = {5, -3, 8, -1, 2} suma maximă este 51 și se obține pentru B' = {2, 8, 5} **(1p)**

2. Se dau n cuburi cu laturile diferite două câte două. Fiecare cub are culoarea albă sau neagră. 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. În cazul în care cuburile pot avea mai multe culori (p culori, codificate {1,2,..,p}) mai este valabilă metoda propusă? Dar dacă lungimile laturilor cuburilor nu erau diferite? **(1p)**

3. **Monede –** Având la dispoziție un număr nelimitat de monede de valori date $\{v_1,v_2,...,v_n\}$, care verifică proprietățile: $v_1 \geq v_2 \geq .. . \geq v_n = 1$ și v_{i-1} este multiplu al lui v_i, pentru orice $i \in \{2,3,...,n\}$, să se determine o modalitate de a plăti o sumă de bani S dată folosind un număr minim de astfel de monede. **Exemplu**: pentru monede de valori {10, 5, 1} şi S = 27 se vor da 5 monede: două monede de valoare 10, o monedă de valoare 5 şi două monede de valoare 1. **(1p)**

4. **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) (2p)**

5. **Problema partiționării intervalelor** – Se consideră n cursuri, pentru care se cunosc ora de început și ora de sfârșit. Determinați numărul minim de săli necesare pentru a putea programa aceste cursuri în aceeași zi și afișați o astfel de programare. Exemplu: pentru n=4 cursuri, care trebuie să se desfășoare în intervalele: [10, 14], [10, 12], [15, 16], respectiv [13, 18], sunt necesare 2 săli, o programare optimă fiind:
   - **Sala 1**: [10, 14] – activitatea 1, [15, 16] – activitatea 3
   - **Sala 2**: [10, 12] – activitatea 2, [13, 18] – activitatea 4 - **O(n logn) (2,5p)**

6. **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, $1 \leq t_i \leq n$) şi profitul p_i care se primeşte dacă activitatea i se execută la timp (cu respectarea termenului limită). Se cere 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 $p_3 + p_4 + p_1 = 2 + 5 + 4 = 11$. Întârzierea planificării este 3. **O(n logn) (3p)**

7. **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**:

$$\boldsymbol{P = \max\{p_1, p_2, ..., p_n\}}$$

Se cere să se determine o planificare a activităţilor date care să aibă întârzierea P minimă.

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. – **O(n log n) (2p)**

8. Dat un arbore cu n vârfuri, să se determine o mulțime de vârfuri neadiacente de cardinal maxim (o submulțime independentă maximă a mulțimii vârfurilor). Pentru un graf oarecare mai este valabilă metoda? – **O(n) (2p)**

9. **Clustering** – vezi curs **(2p)**

10. Considerăm următorul **joc pentru două persoane**. Tabla de joc este o secvență 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 secvenței. Numărul selectat este șters de pe tablă. Jocul se termină când toate numerele au fost selectate. Primul jucător câștigă dacă suma numerelor pe care le-a selectat este **cel puțin 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. Să se scrie un program astfel încât, indiferent cum va juca al doilea jucător, primul jucător câștigă. Scrieți programul astfel încât primul jucător să mute cu ajutorul programului, iar calculatorul să mute aleator de la stânga sau de la dreapta. La ieșire se va scrie suma obținută de primul jucător, suma obținută de cea de al doilea și secvențele de mutări sub forma unor șiruri cu caracterele S pentru stânga și 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 (respective 3+1), deci primul jucător a câştigat - **O(n) (1,5p)**

11. **Memorarea textelor pe banda - cu frecvenţe.** n texte cu lungimile L(1),...,L(n) urmează a fi aşezate pe o bandă. Pentru a citi textul de pe poziţia k, trebuie citite textele de pe poziţiile 1,2,...,k. Pentru fiecare text i se cunoaşte şi frecvenţa f(i) cu care acesta este citit (f(i) = de câte ori este citit textul i). Să se determine o modalitate de aşezare a textelor pe bandă astfel încât timpul total mediu de acces să fie minimizat. **(1p)**

Observaţii

- **Se vor prezenta maxim 4 probleme**
- **Pentru fiecare problemă se acordă punctaj suplimentar pentru demonstrarea corectitudinii (dacă nu a fost făcută integral la curs sau seminar) astfel**
  - **1p pentru problemele 1,2,10**
  - **1,5p pentru restul problemelor**
- **pentru sortări se vor folosi Arrays.sort sau Collections.sort**

Bibliografie

1. **Jon Kleinberg, Éva Tardos** - Algorithm Design**,** 2005 Addison-Wesley Professional

   Slideu-rile oficiale pentru această carte se pot găsi la

   http://www.cs.princeton.edu/~wayne/kleinberg-tardos/pearson/

   o versiune îmbogăţită a acestora este accesibilă la adresa

   http://www.cs.princeton.edu/~wayne/kleinberg-tardos/

   (curs la Univ. Princeton)

2. Pentru problema 6

http://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec10.pdf