UNIVERSITATEA DIN BUCURESTI
STUDENT: VLASCENCO DANIEL-MARIAN
GRUPA: 143 – ANUL I

Tema 1
Curs Structuri de Date

1) Demonstrati ca orice algoritm care construieste un arbore binar de cautare cu n numere ruleaza in timp **Ω**(nlogn).

Demonstratie:
Un algoritm de creare a unui arbore binar de cautare cu n numere este echivalent cu un algoritm de sortare al unui vector cu n elemente.
Astfel problema se rezuma la a demonstra ca orice algoritm de sortare are timp de rulare **Ω**(nlogn). Stim ca un astfel de algoritm poate fi vazut sub forma unui arbore de decizie.
In cel mai nefavorabil caz, numarul comparatiilor realizate de arborele de decizie este egal cu inaltimea arborelui (h). Un arbore binar cu inaltimea h nu poate avea mai mult de 2^h frunze.
$n! \leq 2^h \Rightarrow \log_2 n! \leq h$
Prin aproximarea lui Stirling $\left(n! \geq \frac{n^n}{e^n}\right) \Rightarrow h \geq \log\frac{n^n}{e^n} = n\log n - n\log e = \Omega(\text{n log n})$. ■

2) Demonstrati ca daca f(n) = Θ(g(n)) si g(n) = Θ(h(n)) atunci f(n) = Θ(h(n)).

Demonstratie:

Din f(n) = Θ(g(n)) avem:
$\exists\ c_1, \quad c_2, \quad n_0 > 0\ a.i. \forall\ n \geq n_0 \Rightarrow c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n)$

Din g(n) = Θ(h(n)) avem:
$\exists\ c_3, \quad c_4, \quad n_1 > 0\ a.i. \forall\ n \geq n_1 \Rightarrow c_3 \cdot g(n) \leq h(n) \leq c_4 \cdot g(n)$

Vrem sa aratam ca:
$\exists\ c_5, \quad c_6, \quad n_2 > 0\ a.i. \forall\ n \geq n_2 \Rightarrow c_5 \cdot f(n) \leq h(n) \leq c_6 \cdot f(n)$

Din primele doua relatii rezulta:

$\forall\ n \geq n_0\ avem: \ c_1 \cdot f(n) \leq g(n) \leq \frac{h(n)}{c_3} \Rightarrow \boldsymbol{f(n) \leq \frac{h(n)}{c_1 \cdot c_3}}$

$\forall\ n \geq n_1\ avem: \ \frac{h(n)}{c_4} \leq g(n) \leq c_2 \cdot f(n) \Rightarrow \boldsymbol{f(n) \geq \frac{h(n)}{c_2 \cdot c_4}}$

$Alegem \ \ n_2 = \max(n_0, n_1), \quad c_5 = \frac{1}{c_2 \cdot c_4}, \quad c_6 = \frac{1}{c_3 \cdot c_5}, \ \ de\ unde\ concluzia.$ ■

3) Demonstrati ca $\log(n) = o(\sqrt{n})$.

Demonstratie:

Vom considera, pentru simplicitate, logaritmul natural:

$\ln(n) = o(\sqrt{n}) \Leftrightarrow \forall\, c > 0, \exists\, n_0 > 0\ a.i. \forall\, n > n_0\ avem\ \ln(n) < c \cdot \sqrt{n}$
Problema se reduce la urmatoarea inecuatie: $c \cdot \sqrt{n} - \ln(n) > 0$
Notam cu $f(x) = c \cdot \sqrt{x} - \ln(x)$ si incercam sa-i aflam punctele de extrem. Calculam astfel derivata: $f'(x) = \frac{c}{2\sqrt{x}} - \frac{1}{x} = \frac{c\sqrt{x}-2}{2x}$ pe care o egalam cu 0 si obtinem: $x = \frac{4}{c^2}$. A se nota ca $f : (0, \infty) \to \mathbb{R},$ de asemenea, $c > 0$.
Pentru a calcula mai usor monotonia functiei f, vom considera c = 1.

Se observa ca:

$1 < 4\ si\ f'(1) = \frac{-1}{2} < 0$, deci f e descrescatoare pe intervalul $\left(0, \frac{4}{c^2} \right]$.

$16 > 4\ si\ f'(16) = \frac{2}{32} > 0$, deci f e crescatoare pe intervalul $\left(\frac{4}{c^2}, \infty \right)$.

Astfel, rezulta ca punctul $\frac{4}{c^2}$ este punct de minim, asadar pentru orice n > $\frac{4}{c^2}$, $f(x) > 0$.

Alegem, deci $n_0 = \frac{4}{c^2}$, de unde concluzia. ■

4) Se da un sir cu n numere de la 1 la n, cu exceptia unui numar care apare de 2 ori. Determinati numarul care apare de doua ori.

Solutie:
Consideram ca elementele sunt citite intr-un vector indexat de la 1(prima pozitie e v[1]), de asemenea, functia abs(x) reprezinta modulul numarul x.

```
pentru i = 1, n
        daca v[abs(v[i])] < 0
                cout<< v[i] << "se repeta";
                break;
        altfel
                v[i] = v[i] * (-1)
```

In scrierea acestui algoritm este folosita o singura structura repetitiva (exceptant citirea), complexitatea algoritmului este O(n). ■

5) Fie X[1 :: n] si Y [1 :: n] doi vectori, fiecare continand n numere sortate. Prezentati un algoritm care sa gaseasca mediana celor 2n elemente. Mediana unei multimi de n elemente este elementul de pe pozitia [n/2] in sirul sortat. De exemplu, mediana multimii 3,1,7,6,4,9 este 4.

Solutie:

- Calculam medianele celor doi vectori m_1 (pentru X) si m_2 (pentru Y).
- Daca m_1 si m_2 sunt egale, atunci mediana celor doi vectori este chiar m_1 .
- Daca $m_1 > m_2$, atunci mediana celor doi vectori este prezenta in urmatoarele intervale:
  - X[1 :: m_1]
  - Y[m_2 :: n]
- Daca $m_2 > m_1$, atunci mediana celor doi vectori este prezenta in urmatoarele intervale:
  - X[m_1 :: n]
  - Y[1 :: m_2]
- Aplicam recursiv metoda explicata mai sus pana cand lungimea vectorilor devine 2, caz in care afisam mediana

Se poate observa ca la fiecare recursie se "elimina" n/2 elemente din X, respectiv $\frac{n}{2}$ elemente din Y. Avand in vedere ca timpul folosit de gasirea medianelor m_1 si m_2 este constant(O(1)), timpul de care se foloseste algoritmul poate fi scris sub forma:

T[n] = T[n/2] + 1.

Folosind Teorema Master se poate observa ca:
a = 1, b = 2, f(n) = 1, asadar:

$$f(n) = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^{\log_2 1}\right) = \Theta(n^0) = \Theta(1) = 1.$$
$$Deci\ T(n) = \Theta\left(n^{\log_b a} \cdot \log n\right) = \Theta\left(n^{\log_2 1} \cdot \log n\right) = \Theta(n^0 \cdot \log n) = \Theta(\log n). \blacksquare$$

6) Sa presupunem urmatoarele. Ati castigat la loterie si v-ati cumparat o vila pe care doriti sa o mobilati. Deoarece Ferrari-ul dvs. are capacitate limitata, doriti sa faceti cat mai putine drumuri de la magazin la vila. Mai exact, Ferrari-ul are capacitate n, iar dumneavoastra aveti de cumparat k bunuri de dimensiune $x_1, x_2, \dots, x_k$.
Fie urmatorul algoritm greedy. Parcurgem bunurile in ordinea 1,2,...k si incercam sa le punem in masina. In momentul in care un bun nu mai incape in masina, efectuam un transport si continuam algoritmul.

1. Demonstarti ca algoritmul prezentat mai sus nu este optim. (0.5 puncte)

Demonstratie:
Presupunem ca algoritmul greedy descris mai sus este optim. Acest lucru presupune efectuarea unui numar minim de drumuri. Consideram urmatorul caz:
Capacitate = 20kg.
k = 3 (bunuri)
$x_1 = 5kg;\ x_2 = 18kg;\ x_3 = 7kg.$

Conform algoritmului descris, sunt parcurse 3 drumuri (Pune obiectul de 5 kg in masina, observa ca cel de-al doilea obiect nu incape si face primul drum. Pune apoi obiectul de 18kg in masina, observa ca cel de-al treilea obiect nu incape si face cel de-al doilea drum. In final ia ultimul obiect si face al treilea drum.)
Exista, totusi, o solutie in care pot fi parcurse 2 drumuri: Punem primul si al treilea obiect in masina (12kg in total) si facem un drum, si inca un drum pentru obiectul de 18kg.
Dar am presupus ca algoritmul greedy descris este optim, deci contradicitie. ■

2. Fie OPT, numarul de drumuri in solutia optima. Demonstrati ca algoritmul greedy prezentat mai sus efectueaza cel mult 2OPT drumuri. (1 punct).

Demonstratie:

Fie $s(D_i)$ suma dimensiunilor bunurilor pe care le poate cara Ferrariul nostru pe drumul D_i, in solutia generata de algoritmul descris in cerinta .

Pentru orice doua drumuri consecutive $(D_j \text{ si } D_{j+1})$, stim ca $s(D_j) + s(D_{j+1}) > n$

De notat ca n este capacitatea Ferrariului .

Fie SOL numarul de drumuri parcurse de algoritmul descris in cerinta.

$s(D_1) + s(D_2) > n, s(D_3) + s(D_4) > n, \dots, s(D_{k-1}) + s(D_k) > n.$

Adunand aceste inegalitati, obtinem: $(SUMA\ 1): \sum_{i=1}^{SOL} s(D_i) > \frac{nSOL}{2}$

Analog, pentru algoritmul optim obtinem: $(SUMA\ 2): \sum_{i=1}^{OPT} s(D_i) > \frac{nOPT}{2}$

Stim ca $OPT \leq SOL$. *Astfel,* $(SUMA\ 1) \leq (SUMA\ 2)$. *Presupunem prin absurd ca algoritmul greedy prezentat in cerinta efectueaza mai mult de 2OPT drumuri*

Cu alte cuvinte, am presupus ca $SOL = 2OPT + 1$. *Comparam sumele si inlocuim:*

$(SUMA\ 1)\ de\ comparat\ cu\ (SUMA\ 2)\ \Leftrightarrow$

$\frac{nSOL}{2}\ de\ comparat\ cu\ \frac{nOPT}{2} \Leftrightarrow$

$\frac{(2OPT+1)n}{2}\ de\ comparat\ cu\ \frac{nOPT}{2} \Leftrightarrow$

$2OPT+1\ de\ comparat\ cu\ OPT \Leftrightarrow\ OPT+1\ de\ comparat\ cu\ 0 \Leftrightarrow$
$OPT+1 > 0. Deci\ (SUMA\ 1) > (SUMA\ 2), contradictie. Ramane, deci\ ca$
$SOL \leq 2OPT.$ ■