VERMA DORIAN-ALEXANDRU 324CC
ANALIZA ALGORITMILOR

TEMA 1.

PROBLEMA 1

1.1. CLASE DE COMPLEXITATE

1. • $\Theta(n\log n) \cup o(n\log n) \neq O(n\log n)$ – ADEVĂRAT

Fie f o funcție

$f(n) = \Theta(n\log n) \Rightarrow \exists c_1, c_2 \in \mathbb{R}_+^*, \exists n_0 \in \mathbb{N}^*$ astfel încât
$c_1 n\log n \le f(n) \le c_2 n\log n, \forall n \ge n_0$

$f(n) = o(n\log n) \Rightarrow \forall c_3 \in \mathbb{R}_+^*, \exists n_0' \in \mathbb{N}^*$ astfel încât
$f(n) < c_3 \cdot n\log n, \forall n \ge n_0', c_3 \in \mathbb{R}_+^*$

Din $f(n) < c_3 \cdot n\log n, \forall n \ge n_0', c_3 \in \mathbb{R}_+^* \Rightarrow$
$\Rightarrow f(n) \le c_3 \cdot n\log n, \forall n \ge n_0', c_3 \in \mathbb{R}_+^*$
(deci $o(n\log n) \subseteq O(n\log n)$)

$\Rightarrow$ Putem considera $c_2 = c_3$ și $n_0 = n_0' \Rightarrow$
$\Rightarrow$ reuniunea ar fi $c_1 n\log n \le f(n) \le c_3 n\log n$,
$\forall n \ge n_0' \Rightarrow f(n) \le c_3 n\log n, \forall n \ge n_0', c_3 \in \mathbb{R}_+^* \Rightarrow$

$\Rightarrow f(n) = O(n\log n)$

$\Rightarrow \Theta(n\log n) \cup o(n\log n) \subseteq O(n\log n)$

Se pune problema dacă $O(n\log n) \subseteq \Theta(n\log n) \cup o(n\log n)$ pentru a fi demonstrată dubla incluziune și, deci egalitatea.

Dar $O(n\log n) \not\subseteq \Theta(n\log n)$

, iar $\Theta(n\log n) = O(n\log n) \cap \Omega(n\log n)$

$\Rightarrow$ Dacă $\nexists$ o funcție astfel încât $f \neq \Omega(n\log n)$ și $f = O(n\log n)$, atunci $O(n\log n) \subseteq \Theta(n\log n) \cup o(n\log n)$

O astfel de funcție ar putea fi una care sare la un moment dat la valori mici, iar în rest să fie în $O(n\log n)$. Pt această funcție, afirmația este ADEVĂRATĂ.

• $\sqrt{n}\log n! = \Theta(n^{\frac{3}{2}}\log n)$ - ADEVĂRAT

$\sqrt{n}\log n! = \Theta(n^{\frac{3}{2}}\log n) \Rightarrow$

$\Rightarrow \exists c_1, c_2 \in \mathbb{R}_+^*, \exists n_0 \in \mathbb{N}^*$ astfel încât

$c_1 n^{\frac{3}{2}}\log n \le \sqrt{n}\log n! \le c_2 n^{\frac{3}{2}}\log n \Rightarrow$

$\Rightarrow c_1 n^{\frac{3}{2}}\log n \le n^{\frac{1}{2}}\log n! \le c_2 n^{\frac{3}{2}}\log n \,\Big|: n^{\frac{1}{2}} \Rightarrow$

$\Rightarrow c_1 n\log n \le \log n! \le c_2 n \log n$

$\Rightarrow \log n^{c_1 n} \le \log n! \le \log n^{c_2 n}$ $\Big\} \Rightarrow$
Baza logaritmilor este 2

$\Rightarrow n^{c_1 n} \le n! \le n^{c_2 n}$

Trebuie să demonstrăm că $\exists c_1, c_2 \in \mathbb{R}_+^*$ și $n_0 \in \mathbb{N}^*$ pentru care inegalitățile să fie adevărate pentru $\forall n \ge n_0$

Pentru $c_2 = 1 \Rightarrow n! \le n^n$, deoarece

$n! = \underbrace{1\cdot 2\cdot \ldots \cdot n}_{n \text{ elemente}}$, iar $n^n = \underbrace{n\cdot n\cdot \ldots \cdot n}_{n \text{ elemente}}$, iar $n \ge n-1 \ge \ldots \ge 1 \Rightarrow$

$\Rightarrow n^n \ge n!$, deci $c_2 = 1 \in \mathbb{R}_+^*$, $n \ge 1$

Trebuie să găsim $c_1 \in \mathbb{R}_+^*$ pentru care $n^{c_1 n} \le n!$

Ne întoarcem la relația cu logaritmi

$\Rightarrow c_1 n \log n \le \log n! \Rightarrow c_1 \le \dfrac{\log n!}{n\log n}$

Calculăm $l = \lim\limits_{n\to\infty} \dfrac{\log n!}{n\log n}$

unde $a_n = \log n!$ și $b_n = n\log n$, $n \in \mathbb{N}^*$

(Folosesc lema Cesaro - Stolz)

$b_n \nearrow$ (strict crescător)

$\Rightarrow$ vom calcula limita: $l' = \lim\limits_{n\to\infty} \dfrac{a_{n+1}-a_n}{b_{n+1}-b_n} =$

$= \lim\limits_{n\to\infty} \dfrac{\log(n+1)! - \log n!}{(n+1)\log(n+1) - n\log n} = \lim\limits_{n\to\infty} \dfrac{(\log 1 + \log 2 + \ldots + \log(n+1)) - (\log 1 + \ldots + \log n)}{n\log(n+1) - n\log n + \log(n+1)} =$

Scanned with CamScanner

$$= \lim_{n\to\infty} \frac{\log(n+1)}{n\log\left(\frac{n+1}{n}\right)+\log(n+1)} =$$

$$= \lim_{n\to\infty} \frac{\log(n+1)}{\log\left(1+\frac{1}{n}\right)^n+\log(n+1)}$$

, dar $\left(1+\frac{1}{n}\right)^n \to e$ (limită remarcabilă)

$\Rightarrow \log\left(1+\frac{1}{n}\right)^n \to \log e$

$\Rightarrow l' = 1 \in \overline{\mathbb{R}} \Rightarrow l = 1$

$$\Rightarrow \lim_{n\to\infty} \frac{\log n!}{n\log n} = 1$$

Deci trebuie să găsim un $c_1 < l$

$\Rightarrow$ Putem alege un $c_1 \in (0,1) \Rightarrow$

$\Rightarrow c_1 = \frac{1}{2} = 0,5$, care este valabil pentru $n \geq 2$

(limita tinde la 1 dinspre 0)

(pentru $n=2 \Rightarrow \frac{\log n!}{n\log n} = 0,5$

pentru $n=3 \Rightarrow \frac{\log n!}{n\log n} > 0,5$ și tot așa)

$\Rightarrow \exists\, c_1, c_2 \in \mathbb{R}_+^*$; $c_1 = 0,5$, $c_2 = 1$ și $n_0 = 2 \in \mathbb{N}^*$ astfel încât

$$c_1 n^{\frac{3}{2}}\log n \leq \sqrt{n}\log n! \leq c_2 n^{\frac{3}{2}}\log n$$

$$\Rightarrow \sqrt{n}\log n! = \Theta\left(n^{\frac{3}{2}}\log n\right)$$

(n_0 este 2, deoarece pentru c_2 $n \geq 1$, iar pt. c_1, $n \geq 2$ – mai trebuie să alegem n_0 care să fie adevărat pentru amândouă $\Rightarrow n_0 = 2$)

Scanned with CamScanner

- $n^3 \log^4 n = O(n^4)$ - ADEVĂRAT

$n^3\log^4 n = O(n^4) \Leftrightarrow \exists c \in \mathbb{R}_+^*, \exists n_0 \in \mathbb{N}^*$ astfel încât

$n^3 \log^4 n \leq c \cdot n^4, \forall n \geq n_0$

$n^3 \log^4 n \leq c \cdot n^4 \mid : n^3 \Rightarrow$

$\Rightarrow \log^4 n \leq c \cdot n \Rightarrow \frac{\log^4 n}{n} \leq c, \forall n \geq n_0$

facem $l = \lim\limits_{n\to\infty} \frac{\log^4 n}{n} = l$

$\left.\begin{matrix}\log^4(n) \to \infty \\ n \to \infty\end{matrix}\right\} \Rightarrow$ facem l'Hopital

$\Rightarrow l = \lim\limits_{n\to\infty} \frac{\log^4 n}{n} \overset{l'H}{=\!=} \lim\limits_{n} \frac{4 \cdot \log^3 n \cdot (\log n)'}{1} =$

$= \lim\limits_{n} \frac{4}{\ln 2} \cdot \frac{\log^3 n}{n} \overset{l'H}{=\!=} \lim\limits_{n} \frac{4}{\ln 2} \cdot \frac{3 \cdot \log^2 n}{n \ln 2} =$

$= \lim\limits_{n} \frac{12}{\ln^2 2} \cdot \frac{\log^2 n}{n} \overset{l'H}{=\!=} \lim\limits_{n} \frac{12}{\ln^2 2} \cdot \frac{2}{\ln 2} \cdot \frac{\log n}{n} \overset{l'H}{=\!=}$

$= \lim\limits_{n} \frac{24}{\ln^3 2} \cdot \frac{1}{n \ln 2} = \lim\limits_{n} \frac{24}{\ln^4 2} \cdot \frac{1}{n} = 0$

$\Rightarrow$ pentru $n \to \infty \Rightarrow$ limita este 0.

$\Rightarrow \exists c \in \mathbb{R}_+^*$ astfel încât $\frac{\log^4(n)}{n} \leq c, \forall n \geq n_0$

Trebuie să găsim n_0 și c pentru care $\frac{\log^4(n)}{n} \leq c, \forall n \geq n_0$

Studiem porțiunile pe care funcția este crescătoare și descrescătoare.

Fie $g(x) = \frac{\log^4(x)}{x}$

$g'(x) = \frac{4 \log^3(x) \cdot \frac{1}{x \ln 2} \cdot x - \log^4(x)}{x^2} =$

$= \frac{4x \log^3(x) - x \log^4(x) \ln 2}{x^3 \ln 2}$

$g'(x) = 0 \Rightarrow 4x\log^3 x - x \log^4 x \ln 2 = 0 \Rightarrow$

$\Rightarrow x = 0$ sau $\log^3 x = 0$ sau $\log x = \frac{4}{\ln 2}$

Scanned with CamScanner

$x=0$ nu este bun pentru că avem la numitor x.

$\log^3 x = 0 \Rightarrow x = 1.$

$\log x = \frac{4}{\ln 2} \Rightarrow x = 2^{\frac{4}{\ln 2}} = x_0$

x		1		$2^{\frac{4}{\ln 2}}$		∞
$g'(x)$	$- - -$	0	$+ + + + +$	0	$- - -$	
$g(x)$	$\searrow$	$g(1)$	$\nearrow$	$g(x_0)$	$\searrow$	0

(funcția g descrește către 0 atunci când $x \geq x_0$)

$\Rightarrow$ De la x_0 încolo $\Rightarrow \frac{\log^4 n}{n}$ începe să tindă la 0

$2^{\frac{4}{\ln 2}} < 64 \Rightarrow$ Alegem $n_0 = 64$

calculăm c-ul în funcție de $64 = n_0 \Rightarrow$

$\Rightarrow \frac{\log^4 n_0}{n_0} = \frac{6^4}{64} = \frac{81}{4} = c$

$\Rightarrow \exists\ c = \frac{81}{4} \in \mathbb{R}_+^*,\ \exists\ n_0 = 64$ astfel încât

$n^3 \log^4 n \leq c \cdot n^4,\ \forall n \geq n_0.$

$\Rightarrow n^3 \log^4 n = O(n^4)$

Scanned with CamScanner

• $\log^{2020} n = o(n)$ – ADEVĂRAT

$\log^{2020} n = o(n) \Leftrightarrow \forall c \in \mathbb{R}_+^*, \exists n_0 \in \mathbb{N}^*$ astfel încât

$\log^{2020} n < c \cdot n$

$$l = \lim_{n \to \infty} \frac{\log^{2020} n}{n}$$

$$l \overset{l'H}{=} \lim_{n} \frac{2020}{\ln 2} \cdot \frac{\log^{2019} n}{n} \overset{l'H}{=} \lim_{n} \frac{2020 \cdot 2019}{\ln^2 2} \cdot \frac{\log^{2018} n}{n} =$$

$$= \ldots = \lim_{n} \frac{2020!}{\ln^{2020} 2} \cdot \frac{1}{n} = 0$$

(Am aplicat l'Hopital în mod repetat, în mod similar cu modul în care am rezolvat exercițiul anterior)

$$\lim_{n} \frac{\log^{2020} n}{n} = 0 \Rightarrow \forall c \in \mathbb{R}_+^* \Rightarrow \frac{\log^{2020} n}{n} < c, \forall n \geq n_0$$

$\Rightarrow \log^{2020} n = o(n)$

Scanned with CamScanner

2.

Algoritm 1 (A[0...n], n)	Cost	Repetiții
for i = 1...n	c_1	$n+1$
if(A[i] > 0)	c_2	n
for j = 1...i	c_3	$\sum_{i=1}^{n}(t_i+1)$
if(A[j] mod 2 == 0)	c_4	$\sum_{i=1}^{n} t_i$
for k = 1...j	c_5	$\sum_{i=1}^{n}\sum_{j=1}^{t_i}(t_j+1)$
S = S+i+j+k	c_6	$\sum_{i=1}^{n}\sum_{j=1}^{t_i} t_j$
return S.	c_7	1.

Voi folosi metrica neomogenă pentru a afla complexitatea algoritmului.

Toate operațiile:

$$T(n) = c_1(n+1) + c_2 \cdot n + c_3 \sum_{i=1}^{n}(t_i+1) + c_4 \sum_{i=1}^{n} t_i + c_5 \sum_{i=1}^{n}\sum_{j=1}^{t_i}(t_j+1) + c_6 \sum_{i=1}^{n}\sum_{j=1}^{t_i}(t_j) + c_7 \cdot 1$$

a) cazul favorabil:

În cazul favorabil nu se intră deloc în al doilea for și, în ceea ce e după (în afară de return statement).

Acest lucru se întâmplă atunci când toate elementele din A sunt negative.

$$T(n) = c_1(n+1) + c_2 \cdot n + c_7 \cdot 1 = (c_1+c_2)n + c_1 + c_7 =$$

$$= n(c_1+c_2) + c_1 + c_7 \Rightarrow T(n) = \Theta(n)$$

Scanned with CamScanner

b) cazul defavorabil:

Cazul defavorabil se obține atunci când toate elementele sunt pozitive, iar toate elementele sunt multiplii de 2

Astfel, $t_i = i$, iar $t_j = j$.

(Pentru fiecare element i, se va intra în fiecare for – la fel și pentru j)

$$\Rightarrow T(n) = c_1(n+1) + c_2 \cdot n + c_3 \sum_{i=1}^{n}(i+1) + c_4 \sum_{i=1}^{n} i + c_5 \sum_{i=1}^{n} \sum_{j=1}^{i} (j+1) +$$

$$+ c_6 \sum_{i=1}^{n} \sum_{j=1}^{i} j + c_7 \cdot 1 = c_1(n+1) + c_2 \cdot n + c_3 \cdot \left(\frac{n(n+1)}{2} + n\right) +$$

$$+ c_4 \cdot \frac{n(n+1)}{2} + c_5 \cdot \sum_{i=1}^{n} \left(\frac{i(i+1)}{2} + i\right) + c_6 \cdot \sum_{i=1}^{n} \left(\frac{i(i+1)}{2}\right) + c_7 =$$

$$= c_1(n+1) + c_2 \cdot n + c_3 \cdot \frac{n^2}{2} + c_3 \cdot \frac{n}{2} + c_3 n + c_4 \cdot \frac{n^2}{2} + c_4 \cdot \frac{n}{2} +$$

$$+ c_5 \left(\sum_{i=1}^{n} \left(\frac{i^2}{2} + \frac{i}{2} + i\right)\right) + c_6 \left(\sum_{i=1}^{n} \left(\frac{i^2}{2} + \frac{i}{2}\right)\right) + c_7 =$$

$$= \left(c_1 + c_2 + \frac{c_3}{2} + c_3 + \frac{c_4}{2}\right) n + c_3 \frac{n^2}{2} + c_4 \frac{n^2}{2} + c_1 + c_7 +$$

$$+ c_5 \cdot \left(\frac{1}{2} \cdot \frac{n(n+1)(2n+1)}{6} + \frac{1}{2} \cdot \frac{n(n+1)}{2} + \frac{n(n+1)}{2} \right) +$$

$$+ c_6 \left(\frac{1}{2} \cdot \frac{n(n+1)(2n+1)}{6} + \frac{1}{2} \cdot \frac{n(n+1)}{2} \right) =$$

$$= \left(c_1 + c_2 + \frac{c_3}{2} + 2c_3 + \frac{c_4}{2}\right) n + \left(\frac{c_3}{2} + \frac{c_4}{2}\right) n^2 + c_1 + c_7 +$$

$$+ c_5/2 \cdot \frac{(n^2+n)(2n+1)}{6} + \frac{c_5}{4}(n^2+n) + \frac{c_5 n^2}{2} + \frac{n}{2} c_5 +$$

$$+ \frac{c_6}{12}(n^2+n)(2n+1) + \frac{c_6}{4}(n^2+n) =$$

$$= \left(c_1 + c_2 + \frac{3}{2} c_3 + \frac{c_4}{2}\right) n + \left(\frac{c_3}{2} + \frac{c_4}{2}\right) n^2 + c_1 + c_7 +$$

$$+ \frac{c_5}{12}(2n^3 + 3n^2 + n) + \frac{c_5}{4} n^2 + \frac{c_5}{2} n^2 + \frac{c_5}{4} n + \frac{n}{2} c_5 +$$

$$+ \frac{c_6}{12}(2n^3 + 3n^2 + n) + \frac{c_6}{4} n^2 + n =$$

Scanned with CamScanner

$$= \left(\frac{c_5}{6} + \frac{c_6}{6}\right)n^3 + \left(\frac{c_3}{2} + \frac{c_4}{2} + c_5 + \frac{c_6}{2}\right)n^2 +$$

$$+ \left(c_1 + c_2 + \frac{3}{2}c_3 + \frac{c_4}{2} + \frac{5}{6}c_5 + \frac{13}{12}c_6\right) + c_1 + c_7 \Rightarrow T(n) = \Theta(n^3)$$

c) cazul mediu

În cazul mediu voi considera că jumătate din elementele lui A sunt pozitive, iar jumătate din cele pozitive se împart cu 2

$$\Rightarrow t_i = \frac{i}{2} \text{ , iar } t_j = \frac{j}{2}$$

$$\Rightarrow T(n) = c_1(n+1) + c_2 \cdot n + c_3\sum_{i=1}^{n}{}'\left(\frac{i}{2}+1\right) + c_4\sum_{i=1}^{n}{}'\frac{i}{2} + c_5 \cdot \sum_{i=1}^{n}{}'\sum_{j=1}^{i/2}{}'\left(\frac{j}{2}+1\right) +$$

$$+ c_6\sum_{i=1}^{n}{}'\sum_{j=1}^{i/2}{}' j/2 + c_7 \cdot 1 =$$

$$= c_1(n+1) + c_2 \cdot n + c_3\sum_{i=1}^{n}{}'\frac{i}{2} + c_3\sum_{i=1}^{n}{}' 1 + \frac{c_4}{2}\sum_{i=2}^{n}{}' i +$$

$$+ c_5\sum_{i=1}^{n}\left(\sum_{j=1}^{i/2}{}' j/2 + \sum_{j=1}^{i/2}{}' 1\right) + c_6\sum_{i=1}^{n}\left(\frac{1}{2} \cdot \frac{i/2\,(i/2+1)}{2}\right) + c_7 \cdot 1 =$$

$$= c_1(n+1) + c_2 \cdot n + \frac{c_3}{2} \cdot \frac{n(n+1)}{2} + nc_3 + \frac{c_4}{2} \cdot \frac{n(n+1)}{2} +$$

$$+ c_5\sum_{i=1}^{n}{}'\left(\frac{1}{2} \cdot \frac{i/2\,(i/2+1)}{2} + i/2\right) + c_6 \cdot \frac{1}{2} \cdot \frac{1}{2}\sum_{i=1}^{n}{}'\left(\frac{i^2}{4} + \frac{i}{2}\right) + c_7 \cdot 1 =$$

$$= c_1(n+1) + c_2 \cdot n + \frac{c_3}{4}(n^2+n) + nc_3 + \frac{c_4}{4}(n^2+n) +$$

$$+ c_5\frac{1}{4} \cdot \sum_{i=1}^{n}{}'\left(\frac{i^2}{4} + \frac{i}{2}\right) + \frac{c_5}{2}\sum_{i=1}^{n}{}' i + \frac{c_6}{4} \cdot \frac{1}{4} \cdot \frac{n(n+1)(2n+1)}{6} +$$

$$+ \frac{c_6}{4} \cdot \frac{1}{2} \cdot \frac{n(n+1)}{2} + c_7 = c_1(n+1) + c_2 \cdot n + \frac{c_3}{4}n^2 + \frac{c_3}{4}n + nc_3 + \frac{c_4}{4}n^2 + \frac{c_4}{4}n +$$

$$+ \frac{c_5}{16} \cdot \frac{n(n+1)(2n+1)}{6} + \frac{c_5}{8} \cdot \frac{n(n+1)}{2} + \frac{c_5}{2} \cdot \frac{n(n+1)}{2} + \frac{c_6}{16} \cdot \frac{n(n+1)(2n+1)}{6} +$$

$$+ \frac{c_6}{8} \cdot \frac{n(n+1)}{2} = c_1(n+1) + c_2 \cdot n + \frac{c_3}{4}n^2 + \frac{c_3}{4}n + c_3 \cdot n + \frac{c_4}{4}n^2 + \frac{c_4}{4}n +$$

$$+ \frac{c_5}{96} \cdot (2n^3 + 3n^2 + n) + \frac{c_5}{16}(n^2+n) + \frac{c_5}{4}(n^2+n) + \frac{c_6}{96}(2n^3 + 3n^2 + n) + \frac{c_6}{16}(n^2+n) =$$

Scanned with CamScanner

$$= c_1(n+1) + c_2 \cdot n + \left(\frac{c_3}{4} + \frac{c_4}{4} + \frac{c_5}{32} + \frac{c_5}{16} + \frac{c_5}{4} + \frac{c_6}{32} + \frac{c_6}{16}\right) n^2 + \frac{c_3}{4} n + c_3 \cdot n +$$

$$+ \frac{c_4}{4} n + \left(\frac{c_5}{48} + \frac{c_6}{48}\right) n^3 + \frac{c_5}{96} n + \frac{c_5}{16} n + \frac{c_5}{4} n + \frac{c_6}{96} \cdot n + \frac{c_6}{16} n + c_7 =$$

$$= \left(\frac{c_5}{48} + \frac{c_6}{48}\right) n^3 + \left(\frac{c_3}{4} + \frac{c_4}{4} + \frac{11 c_5}{32} + \frac{3 c_6}{32}\right) n^2 + \left(c_1 + c_2 + \frac{5}{4} c_3 + \frac{c_4}{4} +\right.$$

$$\left. + \frac{31}{96} c_5 + \frac{7}{96} c_6\right) n + c_7 + c_1$$

$$\Rightarrow T(n) = \Theta(n^3)$$

Algoritm2(A[0...n], n)	Cost	Repetiții
s = 0	c_1	1
for i = 1...n	c_2	$n+1$
for j = 1...n	c_3	$\sum_{i=1}^{n}(n+1)$
if(A[i] > A[j])	c_4	$\sum_{i=1}^{n} n$
for k = 1...i	c_5	$\sum_{i=1}^{n}\sum_{k=1}^{t_i}(i+1)$
s = s + A[i]*A[k]	c_6	$\sum_{i=1}^{n}\sum_{k=1}^{t_i} i$
return s	c_7	1

Voi rezolva folosind metrica neomogenă (toate operațiile + operații critice).

Cel de-al doilea for știu sigur că se va executa de $\sum_{i=1}^{n}(n+1)$ ori.

Numărul de repetiții al celui de-al treilea for depinde de numărul de rezultate de true din if. Acest număr este t_i pentru un anume i. Cel de-al treilea for se repetă de i ori.

Toate operațiile:

$$T(n) = c_1 + c_2(n+1) + c_3\cdot\sum_{i=1}^{n}(n+1) + c_4\sum_{i=1}^{n} n +$$

$$+ c_5\sum_{i=1}^{n}\sum_{k=1}^{t_i}(i+1) + c_6\sum_{i=1}^{n}\sum_{k=1}^{t_i} i + c_7$$

a) caz favorabil

În cazul favorabil, if-ul va fi întotdeauna false. În acest caz nu se va intra deloc în cel de-al treilea for.

$\Rightarrow t_i$ va fi 0

$$\Rightarrow T(n) = c_1 + c_2(n+1) + c_3\cdot n(n+1) + c_4\cdot n^2 + c_7 =$$

$$= n^2(c_3+c_4) + n(c_2+c_3) + c_1 + c_2 + c_7$$

$$\Rightarrow T(n) = \Theta(n^2)$$

b) caz defavorabil

În cazul defavorabil, if-ul va fi true pentru toate j-urile din cel de-al doilea for.

$\Rightarrow t_i = n$

$$\Rightarrow T(n) = c_1 + c_2(n+1) + c_3 n(n+1) + c_4 n^2 + c_5\sum_{i=1}^{n}\sum_{k=1}^{n}(i+1) +$$

$$+ c_6\sum_{i=1}^{n}\sum_{k=1}^{n} i + c_7 = c_1 + c_2 n + c_2 + c_3 n^2 + c_3 n + c_4 n^2 +$$

$$+ c_5\sum_{i=1}^{n}((i+1)\cdot n) + c_6\sum_{i=1}^{n} ni + c_7 = c_1 + c_2(n+1) + c_3 n(n+1) +$$

$$+ c_4 n^2 + n c_5\cdot\left(\frac{n(n+1)}{2} + n\right) + c_6 n\frac{n(n+1)}{2} + c_7 =$$

$$= c_1 + c_2 n + c_2 + c_3 n^2 + c_3 n + c_4 n^2 + \frac{c_5}{2}n^3 + \frac{c_5}{2}\cdot n^2 + c_5 n^2 +$$

$$+ \frac{c_6}{2}n^3 + \frac{c_6}{2}n^2 + c_7 =$$

$$= \left(\frac{c_5}{2} + \frac{c_6}{2}\right)n^3 + \left(c_3 + c_4 + \frac{3c_5}{2} + \frac{c_6}{2}\right)n^2 + (c_2+c_3)n + c_1 + c_7$$

$$\Rightarrow T(n) = \Theta(n^3)$$

Scanned with CamScanner

c) cazul mediu:

În cazul mediu, doar jumătate din condițiile din if vor fi true

$\Rightarrow t_i = \frac{n}{2}$

$$\Rightarrow T(n) = c_1 + c_2(n+1) + c_3\sum_{i=1}^{n} 1(n+1) + c_4\sum_{i=1}^{n} 1 n + c_5\sum_{i=1}^{n}\sum_{k=1}^{n/2} 1(i+1) +$$

$$+ c_6\sum_{i=1}^{n}\sum_{k=1}^{n/2} 1 i + c_7 = c_1 + c_2(n+1) + c_3(n+1)n + c_4 n^2 + c_5\sum_{i=1}^{n} 1\left((i+1)\frac{n}{2}\right) +$$

$$+ c_6\sum_{i=1}^{n} 1\left(i\frac{n}{2}\right) + c_7 = c_1 + c_2(n+1) + c_3(n^2+n) + c_4 n^2 + \frac{c_5}{2} n\cdot\left(\frac{n(n+1)}{2} + n\right) +$$

$$+ c_6\cdot\frac{n(n+1)}{2}\cdot\frac{n}{2} + c_7 = c_1 + c_2 n + c_2 + c_3 n^2 + c_3 n + c_4 n^2 + \frac{c_5}{4}(n^3+n^2) +$$

$$+ \frac{c_5}{2} n^2 + \frac{c_6}{4}(n^3+n^2) + c_7 =$$

$$= \left(\frac{c_5}{4} + \frac{c_6}{4}\right)n^3 + \left(c_3 + c_4 + \frac{3c_5}{4} + \frac{c_6}{4}\right)n^2 + (c_2 + c_3)n + c_1 + c_2 + c_7$$

$$\Rightarrow T(n) = \Theta(n^3)$$

Operații critice:

Operațiile critice sunt operațiile executate cel mai frecvent sau cele mai complicate din punct de vedere al complexității:

a) Caz favorabil $\rightarrow$ În cazul favorabil, operațiile critice sunt c_3 și c_4 (cele cu costurile c_3 și c_4)

$$\Rightarrow T(n) = c_3\sum_{i=1}^{n} 1(n+1) + c_4\sum_{i=1}^{n} 1 n =$$

$$= c_3 n(n+1) + c_4 n^2 = (c_3 + c_4)n^2 + c_3 n$$

$$\Rightarrow T(n) = \Theta(n^2)$$

b) cazul defavorabil

În cazul defavorabil, operațiile critice sunt c5 și c6. (cele cu costul c5 și c6), iar $t_i = n$

$$\Rightarrow T(n) = c_5 \sum_{i=1}^{n} \sum_{k=1}^{n} (i+1) + c_6 \sum_{i=1}^{n} \sum_{k=1}^{n} i =$$

$$= c_5 \sum_{i=1}^{n} n(i+1) + c_6 \sum_{i=1}^{n} i\, n = n c_5\left(\frac{n(n+1)}{2} + n\right) +$$

$$+ n c_6 \frac{n(n+1)}{2} = \frac{c_5}{2}\cdot(n^3+n^2) + n^2 c_5 + \frac{c_6}{2}(n^3+n^2) =$$

$$= \left(\frac{c_5}{2} + \frac{c_6}{2}\right)n^3 + \left(\frac{3c_5}{2} + \frac{c_6}{2}\right)n^2$$

$$\Rightarrow T(n) = \Theta(n^3)$$

c) cazul mediu.

În cazul mediu, operațiile critice sunt tot 5 și 6., $t_i = \frac{n}{2}$ (doar jumătate din cazurile din if vor fi true).

$$\Rightarrow T(n) = c_5 \sum_{i=1}^{n} \sum_{k=1}^{n/2} (i+1) + c_6 \sum_{i=1}^{n} \sum_{k=1}^{n} i =$$

$$= c_5 \sum_{i=1}^{n} (i+1)\frac{n}{2} + c_6 \sum_{i=1}^{n} n i = n\frac{c_5}{2}\left(\frac{n(n+1)}{2} + n\right) + n\cdot c_6 \frac{n(n+1)}{2} =$$

$$= \frac{c_5}{4}(n^3+n^2) + c_5 \frac{n^2}{2} + \frac{c_6}{2}(n^3+n^2) =$$

$$= \left(\frac{c_5}{4} + \frac{c_6}{2}\right)n^3 + \left(\frac{3c_5}{4} + \frac{c_6}{2}\right)n^2$$

$$\Rightarrow T(n) = \Theta(n^3)$$

Scanned with CamScanner

PROBLEMA 1.

1.2.

3.

- $T(n) = \begin{cases} 3T\left(\frac{n-2}{2}\right) + k_2 n^2 & , n > 1, k_2 \in \mathbb{R}_+^* \\ k_1 & , n = 1, k_1 \in \mathbb{R}_+^* \end{cases}$

$T(n) = 3\,T\left(\frac{n-2}{2}\right) + \Theta(n^2) \quad , T(1) = \Theta(1)$

Notăm $n = 4m-2$

$\Rightarrow T(4m-2) = 3\,T\left(\frac{4m-2-2}{2}\right) + \Theta((4m-2)^2)$

$T(4m-2) = 3T\left(4 \cdot \frac{m}{2} - 2\right) + \Theta(16m^2 - 16m + 4)$

unde $\Theta(16m^2 - 16m + 4) = \Theta(m^2)$

$\Rightarrow T(4m-2) = 3\,T\left(4 \cdot \frac{m}{2} - 2\right) + \Theta(m^2)$

Notăm $S(m) = T(4m-2)$, $S(1) = \Theta(1)$

$\Rightarrow S(m) = 3\,S(m/2) + \Theta(m^2)$

$S(m) = 3\,S(m/2) + \Theta(m^2) \,|\cdot 3^0$

$S(m/2^1) = 3\,S(m/2^2) + \Theta((m/2^1)^2) \,|\cdot 3^1$

. . . . . .

$S(m/2^k) = 3\,S(m/2^{k+1}) + \Theta((m/2^k)^2) \,|\cdot 3^k$

unde $\frac{m}{2^{k+1}} = 1 \Rightarrow m = 2^{k+1} \Rightarrow k+1 = \lg m \Rightarrow k = \lg m - 1$

$\Rightarrow S(m) = 3^{k+1}\,\Theta(1) + \sum_{i=0}^{k} 3^i \cdot \frac{1}{4^{2i}} \cdot m^2 \cdot \Theta(1)$

$S(m) = 3^{\lg m}\,\Theta(1) + \Theta(m^2)\,\frac{\left(\frac{3}{16}\right)^{k+1} - 1}{\frac{3}{16} - 1} =$

$= m^{\lg 3}\,\Theta(1) + \Theta(m^2) \cdot \frac{\left(\frac{3}{16}\right)^{\lg m} - 1}{-\frac{13}{16}} =$

Scanned with CamScanner

$$= \Theta(n^{\lg 3}) + \Theta(n^2)\cdot\left(-\frac{16}{13}\right)\left(\left(\frac{3}{16}\right)^{\lg n} - 1\right) =$$

$$= \Theta(n^{\lg 3}) + \Theta(n^2)\cdot\left(-\frac{16}{13}\right)\left(n^{\lg \frac{3}{16}} - 1\right) =$$

$$= \Theta(n^{\lg 3}) + \Theta(n^2)\left(\frac{16}{13} - n^{\lg \frac{3}{16}}\right) \Bigg\} \Rightarrow$$

$\lg \frac{3}{16} < 0$

$\Rightarrow n^{2+\lg \frac{3}{16}} < n^2$

$\Rightarrow \Theta(n^2)\left(\frac{16}{13} - n^{\lg \frac{3}{16}}\right) = \Theta(n^2)$

De asemenea $n^{\lg 3} < n^2$ pt. că $\lg 3 < 2$

$\Rightarrow S(n) = \Theta(n^2)$

$T(4n-2) = S(n) = T(4n-2) = \Theta(n^2)$

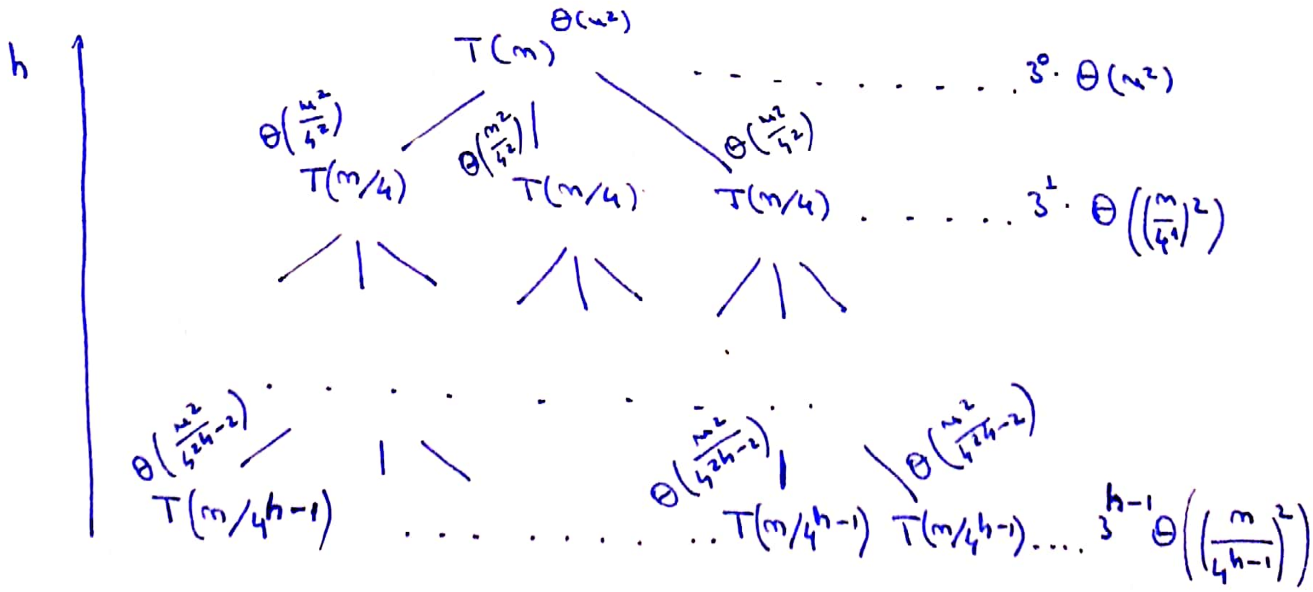
$n = \frac{m+2}{4}$

$\Rightarrow T(n) = \Theta(n^2)$

- $T(m) = \begin{cases} 3T\left(\frac{n}{4}\right) + k_2 n^2 & , n > 1,\ k_2 \in \mathbb{R}_+^* \\ k_1 & , n = 1,\ k_1 \in \mathbb{R}_+^* \end{cases}$

$T(n) = 3T(n/4) + \Theta(n^2)$, $T(1) = \Theta(1)$

Folosesc metoda arborelui de recurență



h - înălțimea, iar $\frac{n}{4^{h-1}} = 1 \Rightarrow h = \log_4 m + 1$

Scanned with CamScanner

$$\Rightarrow \quad T(n) = \sum_{i=0}^{h-1} \left(3^i \cdot \Theta\left(\frac{n^2}{4^{2i}}\right)\right) =$$

$$= \sum_{i=0}^{h-1} \left(\Theta(n^2) \cdot \frac{3^i}{4^{2i}}\right) = \Theta(n^2) \cdot \sum_{i=0}^{h-1} \left(\frac{3}{4^2}\right)^i = \Theta(n^2) \cdot \sum_{i=0}^{h-1} \left(\frac{3}{16}\right)^i =$$

$$= \Theta(n^2) \cdot \frac{\left(\frac{3}{16}\right)^h - 1}{\frac{3}{16} - 1} = \Theta(n^2) \cdot \left(-\frac{16}{13}\right)\left(\left(\frac{3}{16}\right)^{\log_4 n + 1} - 1\right) =$$

$$= \Theta(n^2) \cdot \left(1 \cdot \frac{16}{13} - \frac{16}{13} \cdot \frac{3}{16} \cdot \left(\frac{3}{16}\right)^{\log_4 n}\right) =$$

$$= \Theta(n^2) \cdot \left(\frac{16}{13} - \frac{3}{13} \cdot n^{\log_4 \frac{3}{16}}\right)$$

, dar $\log_4 \frac{3}{16} < 0 \Rightarrow n^2 > n^{2 + \log_4 \frac{3}{16}}$ $\Bigg\} \Rightarrow$

$$\Rightarrow \Theta(n^2)\left(\frac{16}{13} - \frac{3}{13} \cdot n^{\log_4 \frac{3}{16}}\right) = \Theta(n^2)$$

$$\Rightarrow T(n) = \Theta(n^2)$$

- $T(n) = \begin{cases} 3T\left(\frac{n}{4}\right) + n \log n \, k_2, & n > 1, k_2 \in \mathbb{R}_+^* \\ k_1, & n = 1, k_1 \in \mathbb{R}_+^* \end{cases}$

$T(n) = 3T\left(\frac{n}{4}\right) + \Theta(n \log n)$, $T(1) = \Theta(1)$

Rezolv cu metoda MASTER.

Încerc cazul 2:

$T(n) = aT(n/b) + f(n)$

Dacă $f(n) = \Theta(n^{\log_b a})$ atunci $T(n) = \Theta(n^{\log_b a} \cdot \lg n)$
unde lg - logaritm în baza 2 din n.

Dar $a = 3$, $b = 4$

$\Rightarrow n^{\log_b a} = n^{\log_4 3}$

$f(n) = k_2 n \log n = \Theta(n \log n)$, $\log_4 3 < 1$ $\Bigg\} \Rightarrow f(n) \neq \Theta(n^{\log_b a}) \Rightarrow$

Scanned with CamScanner

$\Rightarrow$ nu se poate aplica cazul 2.

Încerc să rezolv folosind cazul 1:

$T(n) = a\,T(n/b) + f(n)$

caz 1: Dacă $\exists\, \varepsilon > 0$ astfel încât $f(n) = O(n^{\log_b a - \varepsilon})$

atunci $T(n) = \Theta(n^{\log_b a})$

$f(n) = k_2 n \log n$

$O(n^{\log_b a - \varepsilon}) = O(n^{\log_4 3 - \varepsilon})$

Încerc să demonstrez că $f(n) = O(n^{\log_4 3 - \varepsilon})$

$\Rightarrow \exists\, c \in \mathbb{R}_+^*,\ \exists\, n_0 \in \mathbb{N}^*$ astfel încât

$c\, n^{\log_4 3 - \varepsilon} \geq k_2 n \log n$

, dar $\log_4 3 < 1 \mid -\varepsilon \Rightarrow \log_4 3 - \varepsilon < 1 - \varepsilon < 1 \Rightarrow$

$\Rightarrow n \log n$ crește mai repede decât $n^{\log_4 3 - \varepsilon}$, unde $\varepsilon > 0$

$\Rightarrow \nexists\, \varepsilon > 0$ astfel încât $f(n) = O(n^{\log_4 3 - \varepsilon})$

$\Rightarrow$ nu putem aplica cazul 1.

Încerc să rezolv folosind cazul 3:

$T(n) = a\,T(n/b) + f(n)$

caz 3: Dacă $\exists\, \varepsilon > 0$ astfel încât $f(n) = \Omega(n^{\log_b a + \varepsilon})$ și

$\exists\, c \in (0,1)$, $n_0 \in \mathbb{N}^*$ astfel încât

$a\, f(n/b) \leq c \cdot f(n),\ \forall n \geq n_0$

atunci $T(n) = \Theta(f(n))$

Am 2 etape în demonstrație:

Prima oară arăt că $\exists\, \varepsilon$.

$\exists\, \varepsilon > 0$ astfel încât $f(n) = \Omega(n^{\log_b a + \varepsilon})$

$\Rightarrow \exists\, c_1 \in \mathbb{R}_+^*,\ \exists\, n_0' \in \mathbb{N}^*$ astfel încât

Scanned with CamScanner

$c_1 n^{\log_4 3+\varepsilon} \leq f(n), \forall n \geq n_0'$

$\Rightarrow c_1 n^{\log_4 3+\varepsilon} \leq k_2 n \log n \mid : n, \forall n \geq n_0' \Rightarrow$

$\Rightarrow c_1 n^{\log_4 3+\varepsilon-1} \leq k_2 \log n$

$\Rightarrow$ Alegem un $\varepsilon > 0$ pentru care $\exists c'$ și n_0' pentru care

$f(n) = \Omega(n^{\log_b a+\varepsilon})$

$\Rightarrow$ Putem alege astfel încât $\log_4 3+\varepsilon-1 = 0 \Rightarrow \varepsilon = 1-\log_4 3 > 0$

$\Rightarrow c_1 n^{\log_4 3+\varepsilon-1} = c_1 n^0 = c_1$

$\Rightarrow k_2 \log n \geq c_1$ unde $n_0' = 2 \in \mathbb{N}^*$, iar $c_1 = k_2 \in \mathbb{R}_+^*$

$\Rightarrow \varepsilon = 1-\log_4 3$ (1)

Apoi arătăm că $\exists c \in (0,1), \exists n_0 \in \mathbb{N}^*$ astfel încât

$a f(n/b) \leq c \cdot f(n), \forall n \geq n_0$

$\Rightarrow 3 \frac{n}{4} \lg \frac{n}{4} \leq c \cdot n \lg n, \forall n \geq n_0$

$\overset{(:n)}{\Longrightarrow} \frac{3}{4} \lg \frac{n}{4} \leq c \lg n$

$\frac{3}{4}(\lg n - \lg 4) \leq c \lg n, \forall n \geq n_0$

$\Rightarrow \frac{3}{4} \cdot \frac{\lg n - \lg 2}{\lg n} \leq c$

$\lg n - \lg 2 \leq \lg n \Rightarrow \frac{\lg n - \lg 2}{\lg n} \leq 1.$

$\Rightarrow c = \frac{3}{4} \in (0,1)$

, iar $n_0 = 2 \in \mathbb{N}^*$, deoarece $n \neq 1$ din moment ce avem logaritm la numitor ($\lg$ este logaritm în baza 2)

$\Rightarrow \exists c = \frac{3}{4} \in (0,1), \exists n_0 \in \mathbb{N}^*$ unde $n_0 = 2$ astfel încât

$a f(n/b) \leq c \cdot f(n), \forall n \geq n_0$ (2)

Din (1) și (2) $\Rightarrow$ putem aplica cazul 3 $\Rightarrow$

$\Rightarrow T(n) = \Theta(n \log n)$

$$\bullet \quad T(n) = \begin{cases} 3T\left(\frac{n}{3}\right) + \frac{n}{\log n} k_2, & n > 1, k_2 \in \mathbb{R}_+^* \\ k_1 & , n = 1, \end{cases}$$

$$T(n) = 3T(n/3) + \Theta(n/\log n), \quad T(1) = \Theta(1)$$

Fac cu metoda iterativă

$$T(n) = 3T(n/3) + \Theta(n/\log n) \quad | \cdot 3^0$$

$$T(n/3^1) = 3T(n/3^2) + \Theta\left(\frac{n}{3}/\log\frac{n}{3}\right) | \cdot 3^1$$

. . . .

$$T(n/3^k) = 3T(n/3^{k+1}) + \Theta\left(\frac{n}{3^k}/\log\frac{n}{3^k}\right) | \cdot 3^k$$

unde $\frac{n}{3^{k+1}} = 1 \Rightarrow n = 3^{k+1} \Rightarrow k = \log_3 n - 1$

$$T(n) = 3^{k+1}\Theta(1) + \sum_{i=0}^{k} 3^i \cdot \frac{n}{3^i \log\frac{n}{3^i}} \Theta(1) =$$

$$= \Theta(n) + \Theta(n)\left(\frac{1}{\log\frac{n}{3^0}} + \frac{1}{\log\frac{n}{3^1}} + \dots + \frac{1}{\log\frac{n}{3^k}}\right) =$$

$$= \Theta(n) + \Theta(n)\left(\frac{1}{\log\frac{n}{3^{\log_3 n - \log_3 n}}} + \dots + \frac{1}{\log\frac{n}{3^{\log_3 n - 2}}} + \frac{1}{\log\frac{n}{3^{\log_3 n - 1}}}\right) =$$

$$= \Theta(n) + \Theta(n)\left(\frac{1}{\log\frac{3n}{n}} + \frac{1}{\log\frac{3^2 n}{n}} + \dots + \frac{1}{\log\frac{3^{\log_3 n} n}{n}}\right) =$$

$$= \Theta(n) + \Theta(n)\left(\frac{1}{\log 3} + \frac{1}{\log 3^2} + \dots + \frac{1}{\log n}\right) =$$

$$= \Theta(n) + \Theta(n)\left(\frac{1}{\log 3} + \frac{1}{2\log 3} + \dots + \frac{1}{\log_3 n \cdot \log 3}\right) =$$

$$= \Theta(n) + \Theta(n) \cdot \frac{1}{\log 3}\left(\frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{\log_3 n}\right)$$

Scanned with CamScanner

$$= \Theta(n) + \Theta(n)\left(\frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{\log_3 n}\right)$$

De la analiză știu că

$$\lim_{n \to \infty} \frac{1 + \frac{1}{2} + \dots + \frac{1}{n}}{\ln n} = 1$$ (demonstrat cu criteriul clește-lui)

$$\Rightarrow 1 + \frac{1}{2} + \dots + \frac{1}{n} \sim \ln n$$

pentru numere mari o să consider

$$\frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{\log_3 n} \sim \ln(\log_3 n) \sim \log(\log n)$$

$$\Rightarrow T(n) = \Theta(n) + \Theta(n)\log(\log n)$$

Consider $\ln n \sim \log n$

$$\Rightarrow T(n) = \Theta(n) + \Theta(n)\log(\log n)$$

$$\Rightarrow T(n) = \Theta(n \log(\log n))$$

Nr. Recurență	Metoda folosită	Complexitatea
Prima recurență	Metoda iterativă	$\Theta(n^2)$
A doua recurență	Metoda arborelui de recurență	$\Theta(n^2)$
A treia recurență	Metoda Master	$\Theta(n \log n)$
A patra recurență	Metoda iterativă	$\Theta(n \log(\log n))$

Scanned with CamScanner

4.

$$\bullet\, T(n) = \begin{cases} 10T\left(\frac{n}{3}\right) + K_2 n^3 & , n > 1, K_2 \in \mathbb{R}_+^* \\ K_1 & , n = 1, K_1 \in \mathbb{R}_+^* \end{cases}$$

Din calculul făcut cu metoda iterativă a rezultat

$T(n) = \Theta(n^3)$

$\Rightarrow$ Luăm $T(n) = \Theta(n^3) \Leftrightarrow$

$\Leftrightarrow \exists c_1, c_2 \in \mathbb{R}_+^*, \exists n_0 \in \mathbb{N}^*$ astfel încât

$c_1 n^3 \leq T(n) \leq c_2 n^3, \forall n \geq n_0$

Caz de bază:

$n = 1: \quad c_1 \leq K_1 \leq c_2$

$\Rightarrow n_0 = 1$ - caz de bază

Pas de inducție: $n/3 \to n$

Ipoteza de inducție

$$c_1\left(\frac{n}{3}\right)^3 \leq T\left(\frac{n}{3}\right) \leq c_2\left(\frac{n}{3}\right)^3$$

Arăt că

$$c_1 n^3 \leq T(n) \leq c_2 n^3$$

$$c_1\left(\frac{n}{3}\right)^3 \leq T\left(\frac{n}{3}\right) \leq c_2\left(\frac{n}{3}\right)^3 \Big| \cdot 10$$

$$\Rightarrow 10c_1\left(\frac{n}{3}\right)^3 \leq 10T\left(\frac{n}{3}\right) \leq 10c_2\left(\frac{n}{3}\right)^3 \Big| + K_2 n^3 \Rightarrow$$

$$\Rightarrow 10c_1\left(\frac{n}{3}\right)^3 + K_2 n^3 \leq 10T\left(\frac{n}{3}\right) + K_2 n^3 \leq 10c_2\left(\frac{n}{3}\right)^3 + K_2 n^3$$

Este îndeajuns să demonstrăm că

$$c_1 n^3 \leq 10c_1\left(\frac{n}{3}\right)^3 + K_2 n^3 \text{ și}$$

$$10c_2\left(\frac{n}{3}\right)^3 + K_2 n^3 \leq c_2 n^3$$

$$c_1 n^3 \leq 10c_1\left(\frac{n}{3}\right)^3 + K_2 n^3 \Rightarrow c_1 \leq 10 \cdot \frac{1}{27} c_1 + K_2 \Rightarrow$$

$$\Rightarrow \frac{17}{27} c_1 \leq K_2 \Rightarrow c_1 \leq \frac{27}{17} K_2 \quad (1)$$

Scanned with CamScanner

$$10 c_2 \left(\frac{n}{3}\right)^3 + K_2 n^3 \le c_2 n^3 \quad |:n^3$$

$$\frac{10}{27} c_2 + K_2 \le c_2 \Rightarrow K_2 \le \frac{17}{27} c_2 \Rightarrow \frac{27}{17} K_2 \le c_2 \ (2)$$

Din cazul de bază avem:

$$c_1 \le K_1 \le c_2$$

Din pasul de inducție avem (1) și (2): $c_1 \le \frac{27}{17} K_2 \le c_2$

Trebuie satisfăcute condițiile de mai sus pentru ca inducția să fie adevărată:

$\Rightarrow \exists\ c_1 = \min\left(K_1, \frac{27}{17} K_2\right)$ și

$\exists\ c_2 = \max\left(K_1, \frac{27}{17} K_2\right)$,

$\exists\ n_0 = 1 \in \mathbb{N}_0^*$ (caz de bază),

astfel încât

$$T(n) = \Theta(n^3)$$

Scanned with CamScanner

$$\cdot\ T(n) = \begin{cases} 7T\left(\frac{n}{3}\right) + k_2 n^2 & , \text{dacă } n > 1,\ k_2 \in \mathbb{R}_+^* \\ k_1 & , \text{dacă } n = 1,\ k_2 \in \mathbb{R}_+^* \end{cases}$$

Din calculul făcut cu metoda iterativă a rezultat

$$T(n) = \Theta(n^2)$$

$\Rightarrow$ Luăm $T(n) = \Theta(n^2) \Leftrightarrow$

$\Leftrightarrow \exists\, c_1, c_2 \in \mathbb{R}_+^*, \exists\, n_0 \in \mathbb{N}^*$ astfel încât

$$c_1 n^2 \le T(n) \le c_2 n^2,\ \forall n \ge n_0$$

Caz de bază:

$n = 1$: $c_1 \le k_1 \le c_2$

$\Rightarrow n_0 = 1$ - caz de bază

Pas de inducție: $n/3 \to n$

Ipoteza de inducție:

$$c_1\left(\frac{n}{3}\right)^2 \le T\left(\frac{n}{3}\right) \le c_2\left(\frac{n}{3}\right)^2$$

Arăt că:

$$c_1 n^2 \le T(n) \le c_2 n^2$$

$$c_1\left(\frac{n}{3}\right)^2 \le T\left(\frac{n}{3}\right) \le c_2\left(\frac{n}{3}\right)^2 \ \Big|\cdot 7 \Rightarrow$$

$$\Rightarrow 7c_1\left(\frac{n}{3}\right)^2 \le 7T\left(\frac{n}{3}\right) \le 7c_2\left(\frac{n}{3}\right)^2 \ \Big| + k_2 n^2 \Rightarrow$$

$$\Rightarrow 7c_1\left(\frac{n}{3}\right)^2 + k_2 n^2 \le 7T\left(\frac{n}{3}\right) + k_2 n^2 \le 7c_2\left(\frac{n}{3}\right)^2 + k_2 n^2 \Rightarrow$$

$$\Rightarrow 7c_1\left(\frac{n}{3}\right)^2 + k_2 n^2 \le T(n) \le 7c_2\left(\frac{n}{3}\right)^2 + k_2 n^2$$

Este îndeajuns să demonstrăm că:

$$c_1 n^2 \le 7c_1\left(\frac{n}{3}\right)^2 + k_2 n^2 \text{ și}$$

$$7c_2\left(\frac{n}{3}\right)^2 + k_2 n^2 \le c_2 n^2$$

Scanned with CamScanner

$$c_1 n^2 \le 7 c_1 \left(\frac{n}{3}\right)^2 + K_2 n^2 \,\Big|\, : n^2 \Rightarrow$$

$$\Rightarrow c_1 \le 7 c_1 \cdot \frac{1}{9} + K_2 \Rightarrow \frac{2}{9} c_1 \le K_2 \quad (1)$$

$$7 c_2 \left(\frac{n}{3}\right)^2 + K_2 n^2 \le c_2 n^2 \,\Big|\, : n^2 \Rightarrow$$

$$\Rightarrow 7 c_2 \cdot \frac{1}{9} + K_2 \le c_2 \Rightarrow K_2 \le \frac{2}{9} c_2 \quad (2)$$

Din cazul de bază avem

$$c_1 \le K_1 \le c_2$$

Din pasul de inducție avem (1) și (2): $\frac{2}{9} c_1 \le K_2 \le \frac{2}{9} c_2 \Rightarrow$

$$\Rightarrow c_1 \le \frac{9}{2} K_2 \le c_2$$

Trebuie satisfăcute condițiile de mai sus pentru ca inducția să fie adevărată:

$$\Rightarrow \exists\, c_1 = \min\left(K_1, \frac{9}{2} K_2\right),$$

$$\exists\, c_2 = \max\left(K_1, \frac{9}{2} K_2\right),$$

$$\exists\, n_0 = 1 \in \mathbb{N}_0^* \text{ (caz de bază)},$$

astfel încât

$$T(n) = \Theta(n^2)$$

Scanned with CamScanner

1.3.

5. reverse (s)

```
    n = size (s)
    if n <= 1                                                          c1
        return s                                                       c2
    else
        return concat (reverse (substring(s, n/2, n)),                 c3
                       reverse (substring(s, 0, n/2)));
```

Voi considera formula de recurență:

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + k_2 n & , k_2 \in \mathbb{R}_+^*, n > 1 \\ k_1 & , k_1 \in \mathbb{R}_+^*, n = 1 \end{cases}$$

$k_1 = c_1 + c_2$ – caz de bază

$n k_2 = c_1 + c_3$ – operația c3 depinde de apelul funcției substring (deci nu va fi în timp constant) ($\frac{n}{2}$ pentru primul apel și $\frac{n}{2}$ pentru al doilea apel al substring din return statement). Am ales partea inferioară și partea superioară a $\frac{n}{2}$ (inferior pt. primul apel de substring și reverse și superior pt. celălalt.

Voi calcula complexitatea folosind metoda iterativă

$$T(n) = 2T(n/2) + \Theta(n) \quad , T(1) = \Theta(1)$$

$$T(n) = 2T(n/2) + \Theta(n) \quad |\cdot 2^0$$

$$T(n/2^1) = 2T(n/2^2) + \Theta(\tfrac{n}{2}) \quad |\cdot 2^1$$

. . . .

$$T(n/2^k) = 2T(n/2^{k+1}) + \Theta\left(\tfrac{n}{2^k}\right) \quad |\cdot 2^k$$

unde $n/2^{k+1} = 1 \Rightarrow k+1 = \log n \Rightarrow k = \log n - 1$

$$\Rightarrow T(n) = 2^{k+1}\Theta(1) + \sum_{i=0}^{k} 2^i \cdot \frac{n}{2^i} \cdot \Theta(1) =$$

$$= 2^{\log n}\Theta(1) + \log n \, \Theta(n) = \Theta(n) + \log n \, \Theta(n) = \Theta(n \log n).$$

Scanned with CamScanner

PROBLEMA 2 - EXPLICAȚII COMPLEXITATE

La problema 2 am ales să implementez un treap, deoarece am vrut să combin funcționalitățile unui arbore binar de căutare cu un heap (am folosit min-heap).

Structura folosită arată așa:

```
typedef struct treapnode {
        Type value;
        int priority;
        struct treapnode *parent;
        struct treapnode *left;
        struct treapnode *right;
} *Node;

typedef struct sequence {
        struct treapnode *node;
        int size;
} *Sequence;
```

Am folosit tipul de date int pt. Type (typedef int Type)

Pentru simplitate și o mai bună organizare, am ales să folosesc două structuri (treapnode desemnând un nod în treap, iar sequence toată structura). Indexul îl folosesc pe post de prioritate a fiecărui nod. Astfel, conform proprietăților unui min-heap, rădăcina arborelui treap va fi cea cu prioritatea cea mai mică (indexul 0). Pentru a face rotațiile mai ușor și a simplifica acest proces de adăugare a unui element și de ștergere, am introdus și un pointer spre părintele fiecărui nod (NULL pt. rădăcină). Structura sequence are în componență un nod treapnode (rădăcina - care la rândul ei se leagă de alte noduri de

Scanned with CamScanner

tip treapnode) și size-ul care reprezintă numărul de elemente din treap.

Fișierul sequence.c conține mai multe funcții ajutătoare pe lângă cele a căror antet a fost dat în enunțul temei. Ele sunt folosite pentru a nu modifica antetele și pentru a simplifica codul. Fișierul main.c conține un demo. Am scris și comentarii în sequence.c și sequence.h pentru a fi de înțeles codul. Comanda „make" compilează fișierele. Comanda „make run" rulează main-ul. „make clean" șterge executabilul.

Mai jos voi explica complexitățile

- Sequence init(); → $O(1)$
  Funcția returnează structura vidă. (data_structure→size==0)
  Conține un malloc pentru structura Sequence.
  De asemenea, nodul rădăcină este inițializat cu NULL.
- Sequence insert(Sequence data_structure, Type item, int index)
  Funcția are complexitatea $O(n)$
  Este compusă din cinci părți:
  1. Este alocată memorie pentru un nou nod (treapnode) - $O(1)$
  2. Se adaugă nodul după valoare (fiind respectate regulile de adăugare într-un arbore binar de căutare) - $O(\log n)$
  3. Se fac rotațiile necesare pentru a duce elementul înapoi spre rădăcină în funcție de prioritate (index) - $O(\log n)$
  4. Se trece prin toate nodurile și se măresc indexurile celor care au index-urile mai mari sau egale cu cel adăugat (astfel încât prioritatea, indexul unui element să fie unic) - $O(n)$
  5. Se incrementează size-ul structurii.

Din cei 5 pași (la 2 și 3 fiind considerat cazul defavorabil) ⇒ ⇒ complexitatea este $O(n)$

Scanned with CamScanner

• Type lookup (Sequence data_structure, int index)

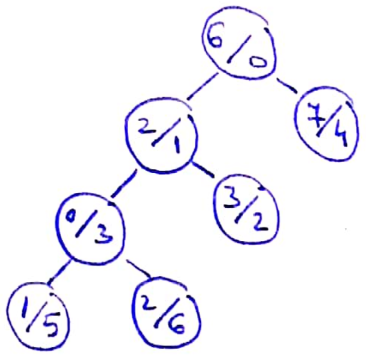
Operația de lookup are în cazul defavorabil complexitatea O(n).

Funcția apelează o funcție ajutătoare (lookup_helper) care returnează referința spre nodul căutat.

Funcția lookup_helper este recursivă și parcurge tot arborele în inordine până când găsește elementul cu prioritatea index. Dacă se caută un nod cu un index < data_structure -> size, atunci, dacă în momentul parcurgerii se ajunge la un element cu prioritatea mai mare, atunci nu se mai continuă în jos pe fiii acestui nod.

Astfel, complexitatea nu este în medie O(n), dar în majoritatea cazurilor este comparabilă cu această complexitate.

Un caz defavorabil ar fi, spre exemplu, acesta:



Dacă se caută elementul de pe poziția 4 (adică 7), atunci se va trece prin toate nodurile.

Astfel, pot spune că este O(n) complexitatea.

• Sequence delete (Sequence data_structure, int index)

Această operație are complexitatea O(n)

1. Se caută elementul care trebuie șters -> O(n)

2. Se șterge nodul -> O(1) dacă vorbim de cazul în care nodul are 1 fiu sau este frunză (caz favorabil) sau dacă succesorul în inordine nu este găsit în O(n).

Funcția „get_succesor" caută succesorul în inordine. Dacă fiul din stânga al fiului din dreapta al nodului de șters are un număr de fii comparabil cu n, atunci complexitatea este O(n). Dacă se consideră acest caz, atunci pasul 2 are complexitatea O(n).

3. Se face reindexarea elementelor cu indexul mai mare decât al elementului care a fost șters. În cazul defavorabil este $O(n)$

4. Se decrementează size-ul structurii. $- O(1)$

=> Complexitatea acestei operații este $O(n)$

- Sequence set (Sequence data - structure, Type item, int index)

Funcția caută în mod similar cu cea de lookup elementul al cărui valoare trebuie înlocuit.

Apelează funcția „set_helper" care se realizează în $O(n)$

=> Complexitatea este $O(n)$

- int size (Sequence data_structure)

Funcția returnează membrul size al structurii Sequence

Are complexitatea $O(1)$

- Sequence *split (Sequence data_structure, int index)

Funcția este compusă din două părți:

1. Se caută rând pe rând, fiecare element începând de la index+1 în data_structure (se folosește lookup) $- O(n^2)$.

Tot aici se inserează în data_structure2 elementul $- O(n)$

2. Se șterg elementele puse în data_structure2 din data_structure $- O(n^2)$

După aceasta se construiește vectorul returnat

=> Complexitate: $O(n^2)$

- Sequence concat (Sequence data_structure1, Sequence data_structure2)

Funcția are 2 părți:

1. Reindexarea în data_structure2 (fiecare element își adaugă indexul cu data_structure1->size) $\rightarrow O(m)$ unde m - nr. de elemente din data_structure2.

2. Apelarea funcției concat_helper care adaugă treap-ul 2 în același mod în care se inserează un nod într-un binary search tree $- O(\log n)$

=> Complexitate: $O(n)$.

Scanned with CamScanner