Tema 1 Analiza Algoritmilor

Digori Gheorghe, grupa 321CC

21 noiembrie 2016

1 Problema 1

1.1 Clase de complexitate

1.1.1

Exercitiu 1)

$$n * log^3(n) = o(n^2)$$

Rezolvare:

$$f(n) = o(g(n))$$

$$(g(n)) = f : N \to R_+^* | \forall c \in R_+^*, c > 0, \exists n(c) \in N \ a.i. 0 \leq f(n) < c * g(n), \forall n > n(c)$$

$$f(n) = n * log^3(n)$$

$$0 \leq n * log^3(n) < c * n^2$$

$$n * log^3(n) < c * n^2 (impartim \ la \ n) \Rightarrow log^3(n) < c * n, \forall c \in R_+$$

$$Fie : n_0 = 1 \to n \geq 1$$

$$[1] Pentru \ n = 1 \Rightarrow atunci \ avem \ 0 < c, \exists c \in R_+^*$$

$$[2] Pentru \ n \to \infty \quad \lim_{x \to \infty} \ c * n - log^3(n) = \infty$$

$$Din \ [1] \ si \ [2] \Rightarrow \ log^3(n) < c * n, \forall \ c \in R_+^*, n > 1$$

Exercitiu 2)

$$log(n!) = \theta(n * log(n))$$

Rezolvare:

$$f(n) = \theta(g(n))$$

$$(g(n)) = f : N \to R_+^* | \exists \ c1, c2 \in R_+^*, c1 > 0, c2 > 0, \ n_0 \in N \ a.i. \ 0 \leq c1 * g(n) \leq f(n) \leq c2 * g(n), \forall n \geq n_0$$

$$log(n!) = \theta(n * logn) \Rightarrow \{ \ log(n!) = O(n * log(n)) \ (1) log(n!) = \Omega(n * log(n)) \ (2)$$

Demonstram (1)

$$\left. \begin{array}{l} n * log(n) = log(n) + log(n) + ... + log(n) \quad (de \ nori) \\ log(n!) = log(1) + log(2) + ... + log(n) \end{array} \right\} \Rightarrow c2 * n * log(n) \geq log(n!), \forall n_0 \geq 1$$

$$\Rightarrow c2 = 1 \quad Astfel \ am \ demonstrat \ ca \ log(n!) = O(n * log(n))(1)$$

Demonstram (2)

$Pentru \ a \ demonstra \ ca \ log(n!) = \Omega(n*log(n)) \ \ adica \ c1*n*log(n) \leq log(n!) \ Ne \ vom \ folosi \ de \ aproximarea \ lui \ Stirling$

$$\left. \begin{array}{l} n/2 * log(n/2) = log(n/2) + log(n/2 + 1) + ... + log(n) \quad (jumatate) \\ log(n!) = log(1) + log(2) + ... + log(n/2) + ... + log(n) \end{array} \right\} \Rightarrow log(n!) \geq n/2*log(n/2) \Rightarrow$$

$n/2 * log(n/2) = n/2 * log(n-1) - n/2 \Rightarrow n/2 * log(n) - 1/2 \geq 1/4 * n * log(n)$

$\Rightarrow log(n!) \geq n/4 * log(n) \Rightarrow log(n!) \geq 1/4 * n * log(n),\ \forall n_0 \geq 1, \forall n \geq n_0 \Rightarrow$

$\exists\ const\ c \in R_+, c1 > 0,\ si\ n_0 \in N\ a.i.\ 0 \leq c1 * g(n) \leq f(n),\ pentru\ orice\ n \geq n_0$

$Deci\ c1 = 1/4\ \ Astfel\ am\ demonstrat\ ca\ log(n!) = \Omega(n * log(n))\ (2)$

$Din\ (1)\ si\ (2) \Rightarrow c1{*}n{*}log(n) \leq log(n!) \leq n{*}log(n),\quad Pentru\ c1 = 1/4\ si\ c2 = 1\ , c1 > 0,\ c2 > 0,\ si\ c1, c2 \in R$

$\Rightarrow log(n!) = \theta(n * log(n))$

Exercitiu 3)

$n! = \Omega(5^{log(n)})$

Rezolvare:

$f(n) = n!\ si\ g(n) = 5^{log(n)}$

$\Omega(g(n)) = f : N \rightarrow R_+ | \exists const\ c \in R_+, c1 > 0, si n_0 \in N a.i. 0 \leq c1{*}g(n) \leq f(n),\ pentru\ orice\ n \geq n_0 \Rightarrow$

$c * 5^{log(n)} \leq n!$

$Presupunem\ ca\ c = 1 \Rightarrow 5^{log(n)} \leq n!,\ n_0 = 1 \Rightarrow$

$Pentru\ n = 1 \Rightarrow 5^{log(1)} = 1! \Rightarrow\ 1 = 1 \Rightarrow$

$c * 5^{log(n)} \leq n!,\quad Pentru\ \forall n \geq n_0 \geq 1$

$Astfel\ rezulta\ ca\ \exists\ c \in R,\ c = 1,\ \Rightarrow c * 5^{log(n)} \leq n!$

1.1.2 Algoritmm

Urmatorul algorim in pseudocod verifica daca exista un element x dintr-un vector v1 si un element y dintr-un vector v2 astfel incat x+y= k . Se presupune ca vectorii au n elemente si sunt sortati crescator si k este un numar natural.

Pseudocodul:

```
suma(v1[], v2[], k, n)
{
        int i, j ;
        ok = 0;
        pentru ( i = 0; i < n; i++){
                pentru (j = 0; j < n; j++){
                        x <- v[i];
                        y <- v[j];
                        daca(x + y = k )
                        {
                                ok = 1;
                                stop; (break)
```

```
                        }
                }
        }
        daca(ok = 1){
                Afiseaza("S-a gasit perechea");
                }
        altfel{
                Afiseaza("Nu s-a putut gasi perechea");
        }
}
```

Complexitatea algoritmului gasit este:

$O(n^2)$

Primul "for" se va avea costul de (n+1) iar al 2-lea "for" va avea

$(n+1)^2$

Atribuirea lui x si y a unor elemente in bucla precum si if-ul din interiorul buclei va avea costul de

$n^2 + 2 * n$

Break si ok au cost 1 , deasemenea si if-ul din afara buclei.

1.2 Gasirea si rezolvarea unei recurente

1)Pentru Algoritmul 1 indentificam ca avem 4 apeluri recursive si gasim ca: Functia recurenta pentru acest algoritm :

$$T(n) = \begin{cases} T(1) = \theta(1) \\ T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \end{cases} \Leftrightarrow T(n) = \begin{cases} T(1) = \theta(1) \\ T(n) = 4T(n/2) + n \end{cases}$$

$Aplic\ metoda\ Master:$

$a = 4,\ b = 2 \Rightarrow n^{\log_b a} = n^2$

$f(n) = \theta(1) = \theta(n^0)\ \Rightarrow Deci\ avem\ Cazul\ 1 \Rightarrow \exists\ \varepsilon\ astfel\ incat\ f(n) = O(n^{log_b(a)-\varepsilon})\ Atunci:$

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

$Astfel\ f(n) = \theta(n^{log_b(a)-\varepsilon})\ cu\ \varepsilon = 2,\ \varepsilon > 0\ deci:$

$\Rightarrow T(n) = \theta(n^{log_b(a)}) = \theta(n^2) \Rightarrow$

$Avem\ complexitatea\ primului\ Algoritm:\ T(n) = \theta(n^2)$

2) Pentru Algoritmul 2 am obsevat ca avem 3 apeluri recursive respectiv:

$$T(n) = \begin{cases} T(1) = \theta(1) \\ T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n \end{cases} \Leftrightarrow T(n) = \begin{cases} T(1) = \theta(1) \\ T(n) = 3T(n/2) + n \end{cases}$$

$Aplic\ metoda\ Master:$

$a = 3,\ b = 2 \Rightarrow n^{\log_b a} = n^{log_2(3)}$

$f(n) = \theta(1) = \theta(n^{log_2(3)-\varepsilon})\ \Rightarrow Deci\ avem\ Cazul\ 1 \Rightarrow \exists\ \varepsilon\ astfel\ incat\ f(n) = O(n^{log_b(a)-\varepsilon})\ Atunci:$

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

$Astfel\ f(n) = \theta(n^{log_b(a)-\varepsilon})\ cu\ \varepsilon = log_2(3),\ \varepsilon > 0\ deci:$

$\Rightarrow T(n) = \theta(n^{log_b(a)}) = \theta(n^{log_2(3)}) \Rightarrow$

$Avem\ complexitatea\ primului\ Algoritm:\ T(n) = \theta(n^{log_2(3)}) = \theta(3^{log_2(n)})$

1.3 Rezolvarea unei recurente

4a)

$$T(n) = \begin{cases} 2 * a * T(n-1) & daca\ n > 1,\ a \in N\ si\ a = const \\ 2 & daca\ n = 1, k_1 \in R_+^* \end{cases}$$

Aplic metoda iterativa

$$\left.\begin{array}{l} T(n) = 2 * a * T(n-1), \quad T(1) = \Theta(1)| * (2*a)^0 \\ T(n-1) = 2 * a * T(n-2), \quad T(1) = \Theta(1)| * (2*a)^1 \\ . \quad . \quad . \quad . \\ T(n-n+2) = 2 * a * T(n-n+1), \quad T(1) = \Theta(1)| * (2*a)^n - 2 \end{array}\right\} " + "$$

Recursivitatea se opreste la pasul n-2. Prin urmare adunand membru cu membru egalitatile de mai sus si reducand termenii identici obtin:

$$T(n) = (2*a)^n - 1 * \theta(1) \Rightarrow T(n) = \theta((2*a)^n - 1) = \theta((2*a)^n), \quad cu\ a \in N,\ a = ct$$

4b)

$$T(n) = \begin{cases} 3 * T(n^{1/3}) & daca\ n > 1 \\ 1 & daca\ n = 1 \end{cases} \quad (n^2)$$

$T(1) = \Theta(1)$

$Substituimn = 3^k \Rightarrow k = log_3(n)$

$T(3^k) = 3 * T(3^{k/3}) + k * log_2(3) \Rightarrow$

$Presupunem\ ca\ avem\ S(k) = T(3^k) \Rightarrow S(k) = 3 * S(k/3) + k * log_2(3)$

$Aplic\,metoda\ Master\ pe\ S(k):$

$a = 3,\ b = 3 \Rightarrow n^{\log_b a} = n^{log_3(3)} = 1$

$f(k) = k * log_2(3)$

Prin urmare se aplica Cazul 2 al Metodei Master:

$f(k) = \theta(k) \Rightarrow k * log_2(3) = \theta(k) \Rightarrow T(k) = \theta(k * log(k) \Rightarrow$

$$k = log_3(n) = ct * log(n), unde\ ct = constanta$$

$$Complexitatea\ este\ : T(n) = \theta(log(n) * log(log(n))$$

5)

$$T(n) = \begin{cases} 3 * T(n/5) + k_2(n^2) & daca\ n > 1 \\ k_2 \in R_+^* k_1 & daca\ n = 1, \quad k_1 \in R_+^* \end{cases}$$

$$Aplic\ metoda\ Master\ pe\ pentru\ a\ afla\ complexitatea :$$

$$a = 3,\ b = 5 \Rightarrow n^{\log_b a} = n^{log_5(3)}$$

Aplcam Cazul 3 al metodei Master si obtinem complexitatea este:

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

Folosim Metoda Substitutiei :

$$\exists\ c_1, c_2 \in R_+^*, \exists\ n_0 \in N_*\ astfel\ incat\ c_1 * n^2 \leq T(n) \leq c_2 * n^2, \forall n > n_0$$

Demonstram Inductia matematica dupa n: Caz de baza : n =1

$$\Rightarrow c_1 \leq k_1 \leq c_2 \Rightarrow n_0 = 1,\ n = 1$$

Ipoteza de inductie:

$$c_1(n^2/5) \leq T(n/5) \leq c_2(n^2/5)$$

Pasul de inductie:

$$n/5 \rightarrow n$$

Aratam ca :

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

$$\Rightarrow c_1(n^2/5) \leq T(n/5) \leq c_2(n^2/5)| * 3 + k_2 * n^2 \Rightarrow\ 3/5 * c_1 * n_2 + k_2 * n_2 \leq T(n) \leq 3/5 * c_2 * n_2 + k_2 * n_2 \Rightarrow$$

$$3/5 * c_1 * n_2 + n_2 * (k_2 - c_1) \leq T(n) \leq 3/5 * c_2 * n_2 + n_2 * (k_2 - c_2) \Rightarrow$$

$$3/5 * c_1 * n_2 \leq 3/5 * c_1 * n_2 + n_2 * (k_2 - c_1) \leq T(n) \leq 3/5 * c_2 * n_2 + n_2 * (k_2 - c_2) \leq 3/5 * c_2 * n_2$$

$$Expresia\ c_1 \leq k_2 \leq c_2\ valida\ si\ la\ cazul\ de\ baza :$$

$$c_1 = min(k_1, k_2)\ si\ c_2 = max(k_1),\ c_1, c_2 \in R_2^*,\ \forall n_0 = 1,\ n_0 \in Na.i\ c_1 * n^2 \leq T(n) \leq c_2 * n^2, \forall n \geq n_0 \Rightarrow$$

$$\Rightarrow T(n) = \theta(n)$$