Tema1 AA

OLARU GABRIEL IULIAN, 324CC

Problema 1

1.1.

1) 0.01* n * log(n) – 2000 * n + 6 = O(n log(n))

Notam : $\begin{cases} f(n) = 0.01 * n * \log(n) - 2000 * n + 6 \\ g(n) = n * \log(n) \end{cases}$

f(n) = O(g(n)) <=> ∃ c € R_{+}^{*} si n0€ N^{*} , a.i. f(n) ≤ c * g(n), ∀ n ≥ n0

0.01 f(n)) ≤ c * g(n) <=> 0.01 * n * log(n) – 2000 * n + 6 ≤ c * n * log(n)

pt c = 0.01 € R_{+}^{*}

0.01 * n * log(n) – 2000 * n + 6 ≤ 0.01 *n * log(n) <=>

-2000 * n +6 ≤ 0 <=>

6 ≤ 2000 *n, adevarat ∀ n ≥ 1 => n0 = 1 € N^{*}

deci ∃ c € R_{+}^{*} si n0€ N^{*} , a.i. f(n) ≤ c * g(n), ∀ n ≥ n0 =>(din definitie)

f(n) = O(g(n)), deci propozitia este adevarata

2) Θ(n * log(n)) U o(n * log (n)) ≠ O(n * log(n))

Θ(n * log(n)) U o(n * log (n)) ≠ O(n * log(n)) <=>

$\begin{cases} \exists f(n) a.i. f(n) = \Theta(n * \log(n)) \ \boldsymbol{si} \ f(n) \neq O(n * \log(n)) (1) \\ \boldsymbol{sau} \\ \exists f(n) \ a.i. f(n) = o(n * \log(n)) \ \boldsymbol{si} \ f(n) \neq O(n * \log(n)) \ (2) \end{cases}$

(1): f(n)=$\Theta(n * \log(n))$ =>(din definitie) ∃c1, c2 € R_{+}^{*} si n0€ N^{*} a.i. c1 * n * log(n) ≤ f(n) ≤ c2 * n * log(n), ∀ n ≥n0

f(n) ≤ c2 * n * log(n), ∀ n ≥n0 => ∃ c3 = c2 € R_{+}^{*} si n1 = n0 € N^{*} a.i. f(n) ≤ c3*n*log(n), ∀ n ≥ n1 =>(din definitie)

=> f(n) = O(n*log(n)) => $\nexists f(n) a.i. f(n) = \Theta(n * \log(n)) \ \boldsymbol{si} \ f(n) \neq O(n * \log(n))$ (1')

(2): f(n) = o(n * log(n)) =>(din definitie) ∀ c4 € R_{+}^{*} si n2 € N^{*} a.i. f(n) < c4 * n * log(n), ∀ n ≥ n2 =>

=> ∃ c5 = c4 € R_{+}^{*} si n3 = n2 € N^{*} a.i. f(n) ≤ c5 * n * log(n) =>(din definitie) f(n) = O(n * log(n)) =>

=> $\nexists f(n) a.i. f(n) = o(n * \log(n)) \ \boldsymbol{si} \ f(n) \neq O(n * \log(n))$ (2')

din (1') si (2') => propozitia este falsa

3) $n^{\frac{1}{2}} \log n! = \Theta(n^{\frac{3}{2}} \log n)$

$n^{\frac{1}{2}} \log n! = \Theta(n^{\frac{3}{2}} \log n)$

$\log n! = \log 1 + \log 2 + \log 3 + \ldots + \log n \leq n * \log n$, ∀ n € N^*

$\log n! \leq n * \log n \quad | * n^{\frac{1}{2}}$ =>

$n^{\frac{1}{2}} \log n! \leq n^{\frac{3}{2}} * \log n$, ∀ n € N^* <=> $n^{\frac{1}{2}} \log n! \leq 1 * n^{\frac{3}{2}} * \log n$, ∀ n ≥ 1 =>

=> ∃ c1 = 1 € R_+^* si n0 = 1 € N^* a.i. $n^{\frac{1}{2}} \log n! \leq c1 * n^{\frac{3}{2}} * \log n$, ∀ n ≥ n0 =>(din definitie)

=>$n^{\frac{1}{2}} \log n! = \Theta(n^{\frac{3}{2}} \log n)$ => propozitia este adevarata

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

$\lim_{n\to\infty} \frac{n!}{5^{\log(n)}} = \infty$ (factorialul converge mult mai repede decat exponentiala) =>(din proprietati)

=> n! = $\omega(5^{\log(n)})$ =>(din definitie) ∀ c € R_+^* si n0 € N^* a.i. n! > c * ($5^{\log(n)}$, ∀ n ≥ n0 =>

=> ∃ c1 = c € R_+^* si n1 = n0 € N^* a.i. n! ≥ c1 * $5^{\log(n)}$, ∀ n ≥ n1 =>(din definitie)

=> n! = $\Omega(5^{\log(n)})$ => propozitia este adevarata

5) log^{2019}n = o(n)

$\lim_{n\to\infty} \frac{log^{2019}\text{n}}{n}$ =(∞/∞, aplicam l'Hopital) = $\lim_{n\to\infty} \frac{2019 * log^{2018}\text{n}}{n}$ = ...(analog aplicam l'Hopital) =

= $\lim_{n\to\infty} \frac{2019*2018*\ldots*1}{n}$ = 0 =>(din proprietati)

=>log^{2019}n = o(n) =>propozitia este adevarata

1.2.

1)

Algoritm1(N[0..n], n)	cost	repetari
for i = 1..n	C1	n
if(A[i] > 0)	C2	n – 1
for j = 1..i	C3	$\sum_{1}^{n} ti$
if(A[j] mod 2 == 0)	C4	$\sum_{1}^{n}(ti-1)$
for k = 1..j	C5	$\sum_{1}^{n}\sum_{1}^{ti} tj$
s = s + i + j + k	C6	$\sum_{1}^{n}\sum_{1}^{ti}(tj-1)$
return s	C7	1

T(n) = c1 * n + c2 * (n - 1) + c3 * $\sum_{1}^{n} ti$ + c4 * $\sum_{1}^{n}(ti-1)$ + c5 * $\sum_{1}^{n}\sum_{1}^{ti} tj$ + c6 * $\sum_{1}^{n}\sum_{1}^{ti}(tj-1)$ + c7

1.Cazul cel mai favorabil: toate numerele sunt negative. ∀ i € [0, n], A[i] < 0

ti = 0, tj =0

T(n) = c1 * n + c2 * (n - 1) + c7 <=>

T(n) = n(c1 + c2) – c2 + c7 =>

T(n) = O (n)

2.Cazul cel mai defavorabil: toate numerele sunt pare si pozitive. ∀ i € [0, n], A[i] > 0 and A[i] mod 2 = 0

ti = i, tj = j

T(n) = c1 * n + c2 * (n - 1) + c3 * $\sum_1^n i$ + c4 * $\sum_1^n (i-1)$ + c5 * $\sum_1^n \sum_1^i j$ + c6 * $\sum_1^n \sum_1^i (j-1)$ + c7

T(n) = c1 * n + c2 * (n - 1) + c3 *$\frac{n(n+1)}{2}$ + c4 * $\frac{n(n-1)}{2}$ + c5 * $\sum_1^n \frac{i(i+1)}{2}$ + c6 * $\sum_1^n \frac{i(i-1)}{2}$ + c7

T(n) = c1 * n + c2 * (n - 1) + c3 * $\frac{n(n+1)}{2}$ + c4 * $\frac{n(n-1)}{2}$ + c5 * ($\frac{n(n+1)}{2} + 1)$ * n + c6 * ($\frac{n(n-1)}{2} + 1)$ * n + c7

T(n) = c1 * n + c2 * (n - 1) + c3 *$\frac{n(n+1)}{2}$ + c4 * $\frac{n(n-1)}{2}$ + c5 *$\frac{n^3 + n^2 + 2*n}{2}$ + c6 * $\frac{n^3 - n^2 + 2*n}{2}$ + c7

T(n) = $n^3 * (\frac{c5}{2} + \frac{c6}{2})$ + $n^2 * (\frac{c3}{2} + \frac{c4}{2} + \frac{c5}{2} - \frac{c6}{2})$ + n * (c1 + c2 + $\frac{c3}{2}$ - $\frac{c4}{2}$ + c5 + c6) – c2 + c7 =>

T(n) = O(n^3)

3.Cazul mediu:

ti = $\frac{i+0}{2}$ = $\frac{i}{2}$; tj = $\frac{j+0}{2}$ = $\frac{j}{2}$

T(n) = c1 * n + c2 * (n - 1) + c3 * $\sum_1^n \frac{i}{2}$ + c4 * $\sum_1^n (\frac{i}{2} - 1)$ + c5 * $\sum_1^n \sum_1^{i/2} \frac{j}{2}$ + c6 * $\sum_1^n \sum_1^{i/2} (\frac{j}{2} - 1)$ + c7

T(n) = c1 * n + c2 * (n - 1) + c3 *$\frac{n(n+1)}{4}$ + c4 * $\frac{n(n-2)}{4}$ + c5 * $\sum_1^n \frac{i(i+1)}{4}$ + c6 * $\sum_1^n \frac{i(i-2)}{4}$ + c7

T(n) = c1 * n + c2 * (n - 1) + c3 * $\frac{n(n+1)}{4}$ + c4 * $\frac{n(n-2)}{4}$ + c5 * ($\frac{n(n+1)}{4} + 1)$ * n + c6 * ($\frac{n(n-2)}{4} + 1)$ * n + c7

T(n) = c1 * n + c2 * (n - 1) + c3 *$\frac{n(n+1)}{4}$ + c4 * $\frac{n(n-2)}{4}$ + c5 *$\frac{n^3 + n^2 + 4*n}{4}$ + c6 * $\frac{n^3 - 2n^2 + 4*n}{4}$ + c7

T(n) = $n^3 * (\frac{c5}{4} + \frac{c6}{4})$ + $n^2 * (\frac{c3}{4} + \frac{c4}{4} + \frac{c5}{4} - \frac{c6}{2})$ + n * (c1 + c2 + $\frac{c3}{4}$ - $\frac{c4}{2}$ + c5 + c6) – c2 + c7 =>

T(n) = O (n^3)

2)

Algoritm2(A[0..n], n)	cost	repetari
s = 0	c1	1
for i = 1 .. n	c2	n
for j = i .. n	c3	$\sum_{1}^{n} ti$
if A[i] > A[j]	c4	$\sum_{1}^{n}(ti-1)$
for k = 1..i	c5	$\sum_{1}^{n}\sum_{ti}^{n} tj$
s = s + A[i] * A[k]	c6	$\sum_{1}^{n}\sum_{ti}^{n}(tj-1)$
return s	c7	1

T(n) = c1 + c2 * n + c3 * $\sum_{1}^{n} ti$ + c4 * $\sum_{1}^{n}(ti-1)$ + c5 *$\sum_{1}^{n}\sum_{ti}^{n} tj$ + c6 * $\sum_{1}^{n}\sum_{ti}^{n}(tj-1)$ + c7

1.Cazul cel mai favorabil: numerele sunt in ordine crescatoare; A[i] < A[j], ∀ i = 1..n si j = i .. n

ti = i, tj = 0

T(n) = c1 + c2 * n + c3 * $\sum_{1}^{n} i$ + c4 * $\sum_{1}^{n}(i-1)$ + c7

T(n) = c1 + c2 * n + c3 *$\frac{n(n+1)}{2}$ + c4 * $\frac{n(n-1)}{2}$ + c7

T(n) = n^2 * ($\frac{c3}{2}+\frac{c4}{2}$) + n * (c2 + $\frac{c3}{2}$ - $\frac{c4}{2}$) + c1 + c7 =>

T(n) = Θ(n^2+n+1) = O(n^2)

2.Cazul cel mai defavorabil: numerele sunt in ordine descrescatoare: A[i] > A[j], ∀ i = 1..n si j = i .. n

ti = i, tj = j

T(n) = c1 + c2 * n + c3 * $\sum_{1}^{n} i$ + c4 * $\sum_{1}^{n}(i-1)$ + c5 *$\sum_{1}^{n}\sum_{i}^{n} j$ + c6 * $\sum_{1}^{n}\sum_{i}^{n}(j-1)$ + c7

T(n) = c1 + c2 * n + c3 * $\frac{n(n+1)}{2}$ + c4 *$\frac{n(n-1)}{2}$ + c5 *$\sum_{1}^{n}\frac{(n+i)(n-i+1)}{2}$ + c6 * $\sum_{1}^{n}\frac{(n+i-2)(n-i+1)}{2}$ + c7

T(n) = c1 + c2 * n + c3 * $\frac{n(n+1)}{2}$ + c4 *$\frac{n(n-1)}{2}$ + c5 * $\frac{2*n^3 * 3*n^2 + n}{6}$ + c6 * $\frac{2*n^3 - 2*n}{3}$ + c7

T(n) = $n^3 *$ ($\frac{c5}{3}+\frac{c6}{3}$) + $n^2 *$ ($\frac{c3}{2}+\frac{c4}{2}+\frac{c5}{3}$) + n * (c2 + $\frac{c3}{2}$ - $\frac{c4}{2}$ + $\frac{c5}{6}$ - $\frac{2* c6}{3}$) + c1 + c7 =>

T(n) = Θ(n^3+n^2+n+1) = O(n^3)

3.Cazul mediu: ti = i; tj = j/2

T(n) = c1 + c2 * n + c3 * $\sum_1^n i$ + c4 * $\sum_1^n (i-1)$ + c5 *$\sum_1^n \sum_i^n \frac{j}{2}$ + c6 * $\sum_1^n \sum_i^n (\frac{j}{2} - 1)$ + c7

T(n) = c1 + c2 * n + c3 * $\frac{n(n+1)}{2}$ + c4 *$\frac{n(n-1)}{2}$ + c5 *$\sum_1^n \frac{(n+i)(n-i+1)}{4}$ + c6 * $\sum_1^n \frac{(n+i-4)(n-i+1)}{4}$ + c7

T(n) = c1 + c2 * n + c3 * $\frac{n(n+1)}{2}$ + c4 *$\frac{n(n-1)}{2}$ + c5 * $\frac{2*n^3 * 3*n^2 + n}{12}$ + c6 * $\frac{2*n^3 - 3*n^2 - 5*n}{3}$ + c7

T(n) = $n^3 * (\frac{c5}{6} + \frac{2*c6}{3})$ + $n^2 * (\frac{c3}{2} + \frac{c4}{2} + \frac{c5}{4} - c6)$ + n * (c2 + $\frac{c3}{2}$ - $\frac{c4}{2}$ + $\frac{c5}{12}$ - $\frac{5*c6}{3}$) + c1 + c7 =>

T(n) = Θ($n^3 + n^2 + n + 1$) = O(n^3)

3)

Algoritm3(n)	cost	repetitii
s = 0	c1	1
for i = 1 .. n	c2	n
for j = i .. i*i	c3	$\sum_1^n (ti^2 - ti + 1)$
s = s + j	c4	$\sum_1^n (ti^2 - ti)$
return s	c5	1

T(n) = c1 + c2 * n + c3 *$\sum_1^n (ti^2 - ti + 1)$ + c4 * $\sum_1^n (ti^2 - ti)$+ c5

Cazul cel mai favorabil <=> Cazul mediu <=> Cazul cel mai putin favorabil

ti = i

T(n) = c1 + c2 * n + c3 *$\sum_1^n (ti^2 - ti + 1)$ + c4 * $\sum_1^n (ti^2 - ti)$ + c5

T(n) = c1 + c2 * n + c3 *$\sum_1^n (i^2 - i + 1)$+ c4 * $\sum_1^n (i^2 - i)$ + c5

T(n) = c1 + c2 * n + c3 * $\frac{n*(n^2+2)}{3}$ + c4 * $\frac{n*(n^2-1)}{3}$

T(n) = n^3 * ($\frac{c3}{3}$ + $\frac{c4}{3}$) + n * (c2 + $\frac{2*c3}{3}$ - $\frac{c4}{3}$) + c1 + c5 =>

T(n) = O(n^3)

4)

Algoritm4(n)	cost	repetari
s = 0	c1	1
i = n / 2	c2	1
while(i < n)	c3	n / 2
j = i	c4	n / 2 – 1
while(j <= n)	c5	2 (n/2 * 2 -> n deci corpul se executa doar o data)
k = 1	c6	1
while(k <= n)	c7	n * $\log n$
s += 1	c8	n* $\log n - 1$
k = k * 2	c9	n *$\log n - 1$
j = j * 2	c10	1
i = i + 1	c11	n / 2 - 1
return s	c12	1

T(n) = c1 + c2 + c3*$\frac{n}{2}$ + c4*($\frac{n}{2}$ – 1)+ c5*2 + c6 + c7 *n*$\log n$ +

+c8*(n*$\log n$–1) + c9*n*($\log n$–1) + c10 + c11 * ($\frac{n}{2}$ -1)+ c12

Cazul cel mai favorabil <=> Cazul mediu <=> Cazul cel mai putin favorabil

T(n) = n * $\log n$(c7 + c8 + c9) + n * ($\frac{c3}{2}$ + $\frac{c4}{2}$ + $\frac{c11}{2}$) + c1 + c2 -c4 + 2*c5 + c6 – c8 -c9 + c10 – c11 + c12 =>

=> T(n) = O(n * $\log n$)

1.3.

Pentru a rezolva problema cu o singur aparcurgere a vectorului (timp liniar) retinem la fiecare pas suma curenta si o mapam la index-ul i, intr-un hashmap. Tinem cont ca secventele 3, 2 si 3, 2, -1, 1 genereaza aceiasi suma, deci nu mai este nevoie sa o mapam, ci doar recalculam lungimea secventei si index-ul de start si stop al secventei. Astfel, dupa o parcurgere vom avea atat lungimea maxima cat si index-ul de start si stop.

```
Alg(arr, N, k)
    startIndex = 0
    stopIndex = 0
    sum = 0
    maxLng = 0
    hash_map h (sum -> index)
    pentru i = 0 .. n – 1 executa
        sum = sum + arr[i]
        daca sum = k atunci
            maxLen = i + 1
            startIndex = 0
            stopindex = i
        daca sum nu este presenta in h atunci
            adauga in  h perechea (sum, i)
        daca sum – k este prezenta in h atunci
            index = index-ul leui (sum – k) in h
            idaca maxLen < i - index atunci
                maxLen = i – index
                startIndex = index
                stopIndex = i
    for i = startIndex .. stopIndex
        print arr[i]
```

Asociem fiecarei operatii din interiorul buclei costul 1, iar antetului buclei costul n. In cazul cel mai defavorabil(cand subsecventa de lungime maxima este sirul in sine), se ecexuta toate instructiunile din interiorul structurilor decizonale de n-1 ori. De asemenea, a doua bucla(pentru afisare) se ecexuta de n ori iar instrunctiunea de print de n-1 ori.

```
Alg(arr, N, k)
        op1 //o data
        for i = 1.. n  //de n ori
           op2 // de n-1 ori
        for i = index .. index //de n ori
           op 3 // de n-1 ori
```

O sa avem deci complexitatea O(2 * (n - 1)) = O(n)

1.4.

1) $T(x) = \begin{cases} 3 * T\left(\frac{n-2}{2}\right) + k2 * n^2, & n > 1, k2 \in R_+^* \\ k1, & n = 1, k1 \in R_+^* \end{cases}$

Metoda Iterativa.

S(n) = T(4 * n - 2) <=> S($\frac{n}{2}$) = T(2 * n - 2) (1)

T(4 * n - 2) = 3 * T($\frac{4* n-2-2}{2}$) + k2 * $(4 * n - 2)^2$ <=>

T(4 * n - 2) = 3 * T(2 * n - 2) + k2 * $(4 * n - 2)^2$ =>(din(1))

S(n) = 3 * S($\frac{n}{2}$) + k2 * $(4 * n - 2)^2$

T(1) = Θ (1)

S(n) = 3 * S($\frac{n}{2}$) + Θ (n^2)

S($\frac{n}{2}$) = 3 * S($\frac{n}{2^2}$) + Θ ($\frac{n^2}{4}$) |*3

S($\frac{n}{2^2}$) = 3 * S($\frac{n}{2^3}$) + Θ ($\frac{n^2}{16}$) |*3^2

...

S($\frac{n}{2^k}$) = 3 * S($\frac{n}{2^{k+1}}$) + Θ ($\frac{n^2}{2^{2*k}}$) |*3^k

---

S(n) = 3^{k+1} * Θ (1) + $\sum_0^k \Theta\left(\frac{\mathrm{n}^2}{2^{2*\mathrm{i}}}\right) * 3^i$

$2^{k+1} = n$ => $\log n$ = k + 1

S(n) = $3^{\log n}$ * Θ (1) + Θ (n^2 * (4 - 3^{k+1} * 4^{-k}))

S(n) = Θ($3^{\log n}$) + Θ($n^2 * 4$ - $n^2 *\ 3^{logn} *\ 4^{-logn+1}$)

S(n) = Θ($3^{\log n} + n^2 * 4$ - $n^2 *\ 3^{logn} *\ 4^{-logn+1}$)

$3^{\log n}$ < 3 * $n^{\log 3}$

S(n) = Θ($n^{\log 3} + n^2 * 4\ - n^2 * n^{\log 3} * \frac{1}{4*n^2}$)

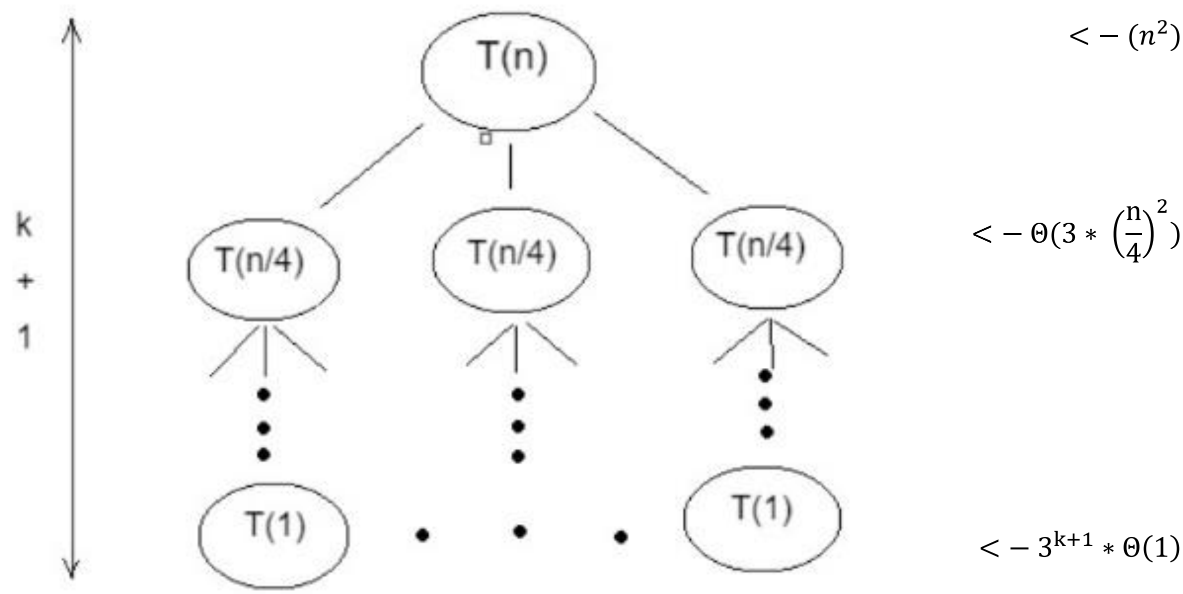
S(n) = Θ($n^{\log 3} +\ n^2 * 4 -\ \frac{n^\wedge log3}{4}$)) => S(n) = Θ($n^2$) = > T(n) = Θ($(4 * n - 2)^2$) =>

T(n) = Θ(n^2)

2) $T(x) = \begin{cases} 3 * T\left(\frac{n}{4}\right) + k2 * n^2, & n > 1, k2 \epsilon R_+^* \\ k1, & n = 1, k1 \epsilon R_+^* \end{cases}$

Metoda arborelui de recurenta.

T(n) = 3 * T ($\frac{n}{4}$) + $\Theta(n^2)$



T(n) = $\sum_0^k 3^i * \Theta(\frac{n^2}{4^{2*i}})$ + $\Theta(n^{k+1})$

$4^{k+1} = \mathrm{n} => \mathrm{k} + 1 = \log_4(n)$

T(n) = $\sum_0^k \Theta(\frac{3^i}{16^i} * n^2)$ + $\Theta(3^{k+1})$

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

T(n) = $\Theta(3^{\log_4(n)})$ + $\Theta(\frac{3^{\log_4(n)} - n^2}{13} * 16)$

T(n) = $\Theta(3^{\log_4(n)}(1 + \frac{16}{13})$ - $\frac{16 * n^2}{13})$

T(n) = $\Theta(n^{\log_4(3)}(1 - \frac{16}{13})$ + $\frac{16 * n^2}{13})$

T(n) = $\Theta(-\frac{3}{13} * n^{\log_4(3)} + \frac{16 * n^2}{13})$

T(n) = Θ (n^2)

3) $T(x) = \begin{cases} 3 * T\left(\frac{n}{4}\right) + k2 * n * \log n\,, \;\; n > 1, k2 \in R_+^* \\ \hfill k1, \;\; n = 1, k1 \in R_+^* \end{cases}$

Metoda master.

$\begin{cases} a = 3 \\ b = 4 \end{cases}$ => $n^{\log_b a}$ = $n^{\log_4 3}$

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

cazul 3:

a) f(n) = Ω ($n^{\log_4 3+\epsilon}$)

∃ c1 € R_+^* si n0 € N^* a.i. n * $\log n$ ≥ c1 * $n^{\log_4 3+\epsilon}$, ∀ n > n0

n * $\log n$ ≥ c1 * $n^{\log_4 3+\epsilon}$ | :n => $\log n$ ≥ c1 * $n^{\log_4 3+\epsilon-1}$, ∀ n > n0

$n^{\log_4 3+\epsilon-1}$ -> 0

trebuie ϵ = 1 - $\log_4 3$ > 0

deci $\log n$ ≥ c1, ∀ n > n0

$\begin{cases} n0 = 3 \\ c1 = 1 \end{cases}$ => $\log n$ ≥ 1, ∀ n > 3

b) ∃ c € (0, 1) si n0 € N^* a.i.

f($\frac{n}{b}$) ≤ c * f(n), ∀ n > n0

$\begin{cases} a = 3 \\ b = 4 \end{cases}$

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

3 * $\frac{n}{4}$ * $\log \frac{n}{4}$ ≤ c * n * $\log n$, ∀ n > n0

$\frac{3}{4}$ ($\log n$ - $\log 4$) ≤ c * $\log n$, ∀ n > n0

$\frac{3}{4}$ ($\log n$ - 2) ≤ c * $\log n$, ∀ n > n0

ca sa pot imparti prin $\log n$, pun conditia n0 > 1

$\frac{3*(\log n - 2)}{4*\log n}$ ≤ c, ∀ n > n0

$\begin{cases} c = \frac{3}{4}\,, € \,(0,1) \\ n0 = 2, € \,N^* \end{cases}$

$\begin{cases} (a)\ adevarat \\ (b)\ adevarat \end{cases}$ =>(cazul 3 din teorema master) T(n) = Θ(n * $\log n$)

4) $T(x) = \begin{cases} 3 * T\left(\frac{n}{3}\right) + k2 * \frac{n}{logn}, & n > 1, k2 \epsilon R_+^* \\ k1, & n = 1, k1 \epsilon R_+^* \end{cases}$

Metoda iterativa.

T(1) = Θ(1)

$3^{k+1} = n$ => $\log_3 n$ = k + 1

$T(n) = 3 * T\left(\frac{n}{3}\right) + \Theta(\frac{n}{logn})$

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

...

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

---

$$T(n) = 3^{k+1} * \Theta(1) + \sum_0^k \Theta(\frac{\frac{n}{3^i}}{\log\left(\frac{n}{3^i}\right)} * 3^i)$$

$$T(n) = 3^{k+1} * \Theta(1) + \sum_0^k \Theta(\frac{n}{\log\left(\frac{n}{3^i}\right)})$$

$$T(n) = 3^{k+1} * \Theta(1) + \sum_0^k \Theta(\frac{n}{\log n - \log(3^i)})$$

$$T(n) = \Theta(3^{k+1}) + \Theta(\frac{\mathrm{n}}{logn - \log\frac{3^{k+1}-1}{3-1}})$$

$$T(n) = \Theta(3^{\log_3 n}) + \Theta(\frac{\mathrm{n}}{logn - \log(3^{\log_3 n} - 1) - 1})$$

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

1.5.

1) $T(n) = \begin{cases} 10 * T\left(\frac{n}{3}\right) + k2 * n^2, \ n > 1, k2 \in R_+^* \\ k1, \ n = 1, k1 \in R_+^* \end{cases}$

Gasim clasa de complexitate folosind metoda iterativa:

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

$3^{k+1} = n$ => $\log_3 n$ = k + 1

$T(n) = 10 * T\left(\frac{n}{3}\right) + \Theta(n^2)$

$T\left(\frac{n}{3}\right) = 10 * T\left(\frac{n}{3^2}\right) + \Theta(\frac{n^2}{9})$ | *10

...

$T\left(\frac{n}{3^k}\right) = 10 * T\left(\frac{n}{3^{k+1}}\right) + \Theta(\frac{n^2}{3^{2*k}})$ | *10^k

____

$T(n) = 10^{k+1} * \Theta(1) + \sum_0^k \Theta(\left(\frac{n^2}{3^{2*i}}\right) * 10^i)$

$T(n) = \Theta(10^{k+1}) + \sum_0^k \Theta(\left(\frac{10^i}{9^i}\right) * n^2)$

$T(n) = \Theta(10^{k+1}) + \Theta(\frac{1-\frac{10^{k+1}}{9^{k+1}}}{1-\frac{10}{9}} * \mathrm{n}^2)$

$T(n) = \Theta(10^{k+1}) + \Theta(\frac{10^{\mathrm{k}+1}}{9^{\mathrm{k}+1}} * 9 * \mathrm{n}^2 - 9 * \mathrm{n}^2)$

$T(n) = \Theta(10^{\log_3 n}) + \Theta(\frac{10^{\log_3 n}}{9^{\log_3 n}} * 9 * \mathrm{n}^2 - 9 * \mathrm{n}^2)$

$T(n) = \Theta(10^{\log_3 n}) + \Theta(10^{\log_3 n} * 9 - 9 * \mathrm{n}^2)$

$T(n) = \Theta(10^{\log_3 n} * 10 - 9 * \mathrm{n}^2)$

T(n) = $\Theta(\mathrm{n}^{\log_3 10} * 10 - 9 * \mathrm{n}^2)$

Aleg clasa de complexitate $\Theta(\mathrm{n}^{\log_3 10} * 10 - 9 * \mathrm{n}^2)$

$\mathrm{T(n)} = \Theta(\mathrm{n}^{\log_3 10} * 10 - 9 * \mathrm{n}^2)$ =>(din definitie)

∃c1, c2 € R_+^* si n0€ N^* a.i. c1 * ($\mathrm{n}^{\log_3 10} * 10 - 9 * \mathrm{n}^2$) ≤ T(n) ≤ c2 * ($\mathrm{n}^{\log_3 10} * 10 - 9 * \mathrm{n}^2$), ∀ n ≥n0

Cazul de baza: T(1) = $\Theta(1)$

c1 ≤ k1 ≤ c2, adevarat ∀ n ≥ 1 (n0 = 1)

Pasul de inductie: n/3 -> n

Ipoteza de inductie: c1 * $((\frac{n}{3})^{\log_3 10} * 10 - 9 * (\frac{n}{3})^2)$ ≤ T($(\frac{n}{3})$ ≤ c2 * $((\frac{n}{3})^{\log_3 10} * 10 - 9 * (\frac{n}{3})^2)$

Arat ca: c1 * $(n^{\log_3 10} * 10 - 9 * n^2)$ ≤ T(n) ≤ c2 * $(n^{\log_3 10} * 10 - 9 * n^2)$

c1 * $((\frac{n}{3})^{\log_3 10} * 10 - 9 * (\frac{n}{3})^2)$ ≤ T($(\frac{n}{3})$ ≤ c2 * $((\frac{n}{3})^{\log_3 10} * 10 - 9 * (\frac{n}{3})^2)$ | * 10 =>

c1 * 10 *$(n^{\log_3 10} - n^2)$ ≤ 10 *T($(\frac{n}{3})$ ≤ c2 * 10 *$(n^{\log_3 10} - n^2)$ | + k2 * n^2 =>

c1 * 10 *$(n^{\log_3 10} - n^2)$ + k2 * n^2 ≤ 10 *T($(\frac{n}{3})$ + k2 * n^2 ≤ c2 * 10 *$(n^{\log_3 10} - n^2)$ + k2 * n^2 =>(din recurenta)

c1 * 10 *$(n^{\log_3 10} - n^2)$ + k2 * n^2 ≤ T(n) ≤ c2 * 10 *$(n^{\log_3 10} - n^2)$ + k2 * n^2

c1 * $(n^{\log_3 10} * 10 - 9 * n^2)$ ≤ c1 * 10 *$(n^{\log_3 10} - n^2)$ + k2 * n^2 <=>

-9 * c1 * n^2 ≤ -10 * c1 * n^2 + k2 * n^2 <=>

c1 * n^2 ≤ k2 * n^2 <=> c1 ≤ k2

c2 * 10 *$(n^{\log_3 10} - n^2)$ + k2 * n^2 ≤ c2 * $(n^{\log_3 10} * 10 - 9 * n^2)$ <=>

-10 * c2 * n^2 + k2 * n^2 ≤ -9 * c2 * n^2 <=>

k2 * n^2 ≤ c2 * n^2 <=> k2 <= c2

$\begin{cases} c1 \le k1 \le c2 \\ c1 \le k2 \\ k2 \le c2 \end{cases}$ c1 = c2, k2 = k2 = c1

=> T(n) = $\Theta(n^{\log_3 10} * 10 - 9 * n^2)$ cctd.

2) $T(n) = \begin{cases} 2 * T\left(\frac{n}{3}\right) + k2 * n^4, \; n > 1, k2 \in R_+^* \\ k1, \; n = 1, k1 \in R_+^* \end{cases}$

Gasim clasa de complexitate folosind metoda iterativa:

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

$3^{k+1} = n$ => $\log_3 n$ = k + 1

$T(n) = 2 * T\left(\frac{n}{3}\right) + \Theta(n^4)$

$T\left(\frac{n}{3}\right) = 2 * T\left(\frac{n}{3^2}\right) + \Theta(\frac{n^4}{81})$ | *2

...

$T\left(\frac{n}{3^k}\right) = 2 * T\left(\frac{n}{3^{k+1}}\right) + \Theta(\frac{n^4}{3^{4*k}})$ | $*2^k$

$T(n) = 2^{k+1} * \Theta(1) + \sum_0^k \Theta(\left(\frac{n^4}{3^{4*i}}\right) * 2^i)$

$T(n) = \Theta\left(2^{k+1}\right) + \sum_0^k \Theta(\left(\frac{2^i}{81^i}\right) * n^4)$

$T(n) = \Theta\left(2^{k+1}\right) + \Theta(\frac{1-\frac{2^{k+1}}{81^{k+1}}}{1-\frac{2}{81}} * \mathrm{n}^4)$

$T(n) = \Theta\left(2^{k+1}\right) + \Theta((1 - \frac{2^{k+1}}{81^{k+1}}) * \frac{81}{79} * \mathrm{n}^4)$

$T(n) = \Theta\left(2^{k+1}\right) + \Theta((1 - \frac{2^{k+1}}{81^{k+1}}) * \frac{81}{79} * \mathrm{n}^4)$

$T(n) = \Theta\left(2^{\log_3 n}\right) + \Theta(\frac{81}{79} * \mathrm{n}^4 - \frac{2^{\log_3 n}}{81^{\log_3 n}} * \frac{81}{79} * \mathrm{n}^4)$

$T(n) = \Theta(2^{\log_3 n} + \frac{81}{79} * \mathrm{n}^4 - 2^{\log_3 n} * \frac{81}{79})$

$T(n) = \Theta(\frac{81}{79} * \mathrm{n}^4 - 2^{\log_3 n} * \frac{2}{79})$

Aleg clasa de complexitate $\Theta(\frac{81}{79} * \mathrm{n}^4 - 2^{\log_3 n} * \frac{2}{79})$

$\mathrm{T(n)} = \Theta(\frac{81}{79} * \mathrm{n}^4 - 2^{\log_3 n} * \frac{2}{79})$ =>(din definitie)

∃c1, c2 € R_+^* si n0€ N^* a.i. c1 * ($\frac{81}{79} * \mathrm{n}^4 - 2^{\log_3 n} * \frac{2}{79}$) ≤ T(n) ≤ c2 * ($\frac{81}{79} * \mathrm{n}^4 - 2^{\log_3 n} * \frac{2}{79}$), ∀ n ≥n0

Cazul de baza: T(1) = $\Theta(1)$

c1 $\leq$ k1 $\leq$ c2, adevarat ∀ n ≥ 1 (n0 = 1)

Pasul de inductie: n/3 -> n

Ipoteza de inductie: c1 * ($\frac{81}{79} * \left(\frac{n}{3}\right)^4 - 2^{\log_3 \frac{n}{3}} * \frac{2}{79}$) ≤ T($\frac{n}{3}$) ≤ c2 * ($\frac{81}{79} * \left(\frac{n}{3}\right)^4 - 2^{\log_3 \frac{n}{3}} * \frac{2}{79}$)

Arat ca: c1 * ($\frac{81}{79} * \mathrm{n}^4 - 2^{\log_3 n} * \frac{2}{79}$) ≤ T(n) ≤ c2 * ($\frac{81}{79} * \mathrm{n}^4 - 2^{\log_3 n} * \frac{2}{79}$

c1 * ($\frac{81}{79} * \left(\frac{n}{3}\right)^4 - 2^{\log_3 \frac{n}{3}} * \frac{2}{79}$) ≤ T($\frac{n}{3}$) ≤ c2 * ($\frac{81}{79} * \left(\frac{n}{3}\right)^4 - 2^{\log_3 \frac{n}{3}} * \frac{2}{79}$) | *2 =>

2 *c1 * ($\frac{81}{79} * \left(\frac{n}{3}\right)^4 - 2^{\log_3 \frac{n}{3}} * \frac{2}{79}$) ≤ 2 * T($\frac{n}{3}$) ≤ 2 *c2 * ($\frac{81}{79} * \left(\frac{n}{3}\right)^4 - 2^{\log_3 \frac{n}{3}} * \frac{2}{79}$) | +k2 *$n^4$ =>

2 *c1 * ($\frac{81}{79} * \left(\frac{n}{3}\right)^4 - 2^{\log_3 \frac{n}{3}} * \frac{2}{79}$) +k2 *$n^4$≤ 2 * T($\frac{n}{3}$) +k2 *$n^4$≤ 2 *c2 * ($\frac{81}{79} * \left(\frac{n}{3}\right)^4 - 2^{\log_3 \frac{n}{3}} * \frac{2}{79}$) +k2 *$n^4$

2 *c1 * ($\frac{\mathrm{n}^4}{79} - \frac{n^{\log_3 2}}{79}$) +k2 *$n^4$≤ 2 * T($\frac{n}{3}$) +k2 *$n^4$≤ 2 *c2 * ($\frac{\mathrm{n}^4}{79} - \frac{n^{\log_3 2}}{79}$)) +k2 *$n^4$

=>(din recurena) 2 *c1 * ($\frac{\mathrm{n}^4}{79} - \frac{n^{\log_3 2}}{79}$) +k2 *$n^4$≤ T(n) ≤ 2 *c2 * ($\frac{\mathrm{n}^4}{79} - \frac{n^{\log_3 2}}{79}$)) +k2 *$n^4$

c1 * ($\frac{81}{79} * n^4 - \frac{2 * n^{\log_3 2}}{79}$) ≤ 2 *c1 * ($\frac{\mathrm{n}^4}{79} - \frac{n^{\log_3 2}}{79}$) + k2 *$n^4$ <=>

$\frac{81}{79} * c1 \;\leq\; 2 * \mathrm{c1} * \frac{1}{79} + k2$ <=>

c1 ≤ k2

2 *c1 * ($\frac{\mathrm{n}^4}{79} - \frac{n^{\log_3 2}}{79}$) + k2 *$n^4$ ≤ c2 * ($\frac{81}{79} * n^4 - \frac{2 * n^{\log_3 2}}{79}$) <=>

$2 * \mathrm{c1} * \frac{1}{79} + k2 \;\leq\; \frac{81}{79} * c2$ <=>

k2 ≤ c2

$\begin{cases} \mathrm{c1} \;\leq\; \mathrm{k1} \;\leq\; \mathrm{c2} \\ \quad \mathrm{c1} \;\leq\; \mathrm{k2} \\ \quad \mathrm{k2} \;\leq\; \mathrm{c2} \end{cases}$ c1 = c2, k2 = k2 = c1

=> T(n) = Θ(($\frac{81}{79} * \mathrm{n}^4 - 2^{\log_3 n} * \frac{2}{79}$) cctd.

1.6.

1) $T(x) = \begin{cases} 4 * T\left(\frac{n}{2}\right) + k2 * n,\ n > 1, k2 \epsilon R_+^* \\ k1,\ n = 1, k1 \epsilon R_+^* \end{cases}$

Rezolvam folosind metoda iterativa.

T(1) = Θ(1)

T(n) = 4 * S($\frac{n}{2}$) + Θ(n)

T($\frac{n}{2}$) = 4 * S($\frac{n}{2^2}$) + Θ($\frac{n}{2}$) |*4

T($\frac{n}{2^2}$) = 4 * S($\frac{n}{2^3}$) + Θ($\frac{n}{2^2}$) |*4^2

...

T($\frac{n}{2^k}$) = 4 * S($\frac{n}{2^{k+1}}$) + Θ($\frac{n}{2^k}$) |*4^k

____

T(n) = 4^{k+1} * Θ(1) + $\sum_0^k \Theta(\frac{\mathrm{n}}{2^{\mathrm{i}}})\ 4^i$

$2^{k+1} = n$ => $\log n$ = k + 1

T(n) = $4^{\log n}$ * Θ(1) + Θ(n *($2^{\mathrm{logn}} - 1$))

T(n) = Θ($4^{\log n}$) + Θ(n^2 - n))

T(n) = Θ(2 *n^2 - n))

T(n) = Θ (n^2)

2) $T(x) = \begin{cases} 3 * T\left(\frac{n}{2}\right) + k2 * n, & n > 1, k2 \in R_+^* \\ k1, & n = 1, k1 \in R_+^* \end{cases}$

Rezolvam folosind metoda iterativa.

T(1) = Θ(1)

T(n) = 3 * S($\frac{n}{2}$) + Θ(n)

T($\frac{n}{2}$) = 3 * S($\frac{n}{2^2}$) + Θ($\frac{n}{2}$) |*3

T($\frac{n}{2^2}$) = 3 * S($\frac{n}{2^3}$) + Θ($\frac{n}{2^2}$) |*3^2

...

T($\frac{n}{2^k}$) = 3 * S($\frac{n}{2^{k+1}}$) + Θ($\frac{n}{2^k}$) |*3^k

____

T(n) = 3^{k+1} * Θ(1) + $\sum_0^k \Theta(\frac{\mathrm{n}}{2^{\mathrm{i}}}) * 3^i$

$2^{k+1} = n$ => $\log n$ = k + 1

T(n) = $3^{\log n}$ * Θ(1) + Θ(n *($2^{-\mathrm{logn}+1} * \ 3^{\mathrm{logn}}$ - 2))

T(n) = Θ($3^{\log n}$) + Θ($\frac{3^{\mathrm{logn}}}{2}$ - 2n))

T(n) = Θ($3^{\log n}$ + $\frac{3^{\mathrm{logn}}}{2}$ - 2n)

T(n) = Θ($\frac{3 * 3^{\mathrm{logn}}}{2}$ - 2n)

$3^{\log n}$ < 3 * $n^{\log 3}$

deci: T(n) = Θ($n^{\log 3}$)

Problema 2

Pentru a putea implementa o structura de date care sa permita cat mai multe operatii in timp logaritmic pornim de la o structura de date care permite deja operatiile de baza precum accesarea(lookup), inserarea stergerea. Pentru aceasta, ne putem folosi de orice tip de arbore balansat. In implementarea structurii am ales avl-ul.

Daca lucram doar cu date(fara index, atunci un avl ar fi fost suficient), dar deoarece fiecarei valori ii corespunde un index va trebui sa modificam proprietatile avl-ului pentru a ne putea permite sa pastram ordinea in care au fost introduse elementele. Astfel, index-ul devine cheia.

Insert, delete, set, lookup

Pentru a putea insera ne folosim de dimensiunea(numarul de noduri) subarborelui stang respectiv a celui drept. Daca indexul valorii pe care dorim sa o inseram este mai mare decat dimensiunea subarborelui stang, inseamna ca va trebui sa il inseram in subarborele drept. Cautam recursiv in subarborele drept, sacazand din index numarul de noduri pe care "le-am vazut deja", adica dimensiunea subarborelui stang. Astfel, arborele isi pastreaza proprietatea de AVL, avand inaltimea maxima de logn, iar la parcurgerea sa in inordine se vor afisa elementele in ordinea in care au fost introduse. La delete se va proceda similar.

Deorarece arborele creat este inca un AVL => are inaltimea maxima de logn => inserarea se va efectua, in cel mai rau caz, in O(nlogn). Analog operatia de delete.

Operatia de lookup este aceeasi cu cautarea intr-un arbore binar de cautare. Deoarece AVL-ul este balansat(are inaltimea maxima de logn) garanam lookup-ul in timp logaritmic.

Operatia de set include un lookup(O(logn)) si modificarea valorii de la un index(O(1)), deci se va efectua in timp logaritmic.

Split si Concat

Operatiile de split si concat nu se pot efectua in timp logaritmic. Pentru doua structuri de dimensiuni m si n, s-a demonstrat ca cea mai buna complexitate este O(m * log(1 + $\frac{n}{m}$)).

O metoda pentru a concatena cele doua structuri este de a parcurge una din ele si a insera treptat elementele in a doua, rezultand o complexitate de n*logn(daca au aceeasi dimensiune n)

Pentru split se va face o cautare(log n) si apoi se vor insera toate nodurile elementele care preced intr-o noua structura(n *logn), rezultand o complexitate de n* $(\log^2 n)$.

Din lipsa de timp, structura implementata in fisierul avl.c este doar un avl normal, adaugand valori(deci nu pastreaza indecsii) si efectuand operatiile de insert, set si lookup, in timp logaritmic.

Referinte: https://en.wikipedia.org/wiki/Join-based_tree_algorithms

http://goto.ucsd.edu/~ucsdpl-blog/datastructures/2015/12/09/avl-trees/