SDA 101

A. $T(n) = \begin{cases} 1 & n=0 \\ \sum_{i=1}^{n-1} T(i), & \text{altfel} \end{cases}$

f(n) apelează f(n) (ciclu infinit) {presupun că for merge doar până la n-1}

$T(0) = 1$

$T(1) = 1$

$T(2) = 2$

$T(3) = 4$

$T(4) = 8$

→ observăm că $T(n) = 2^{n-1}$ $\forall n \geq 1$ (putem demostra prin inducție)

⇒ cei doi timpi vor fi egali cu $\Theta(2^n)$

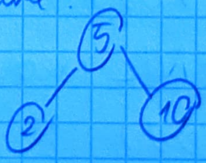
$2^{n-1} \in \Theta(2^n)$ deoarece $\lim_{n\to\infty} \frac{2^n}{2^{n-1}} = 2$ (constantă, val de dispersie

B.

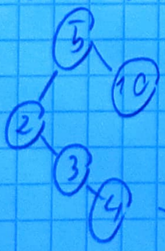
↳0 Celule	1	2	3	4	5	6	7	8	9
↳5		2	3		35	6	10	18	8

→ verificarea liniară introduce elementele în ordine.

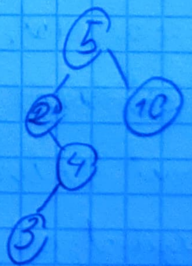
C. ⓑ Contra exemplu: fie arborele



Adăugăm pe 3, 4 :



Adăugăm pe 4, 3 :



→ Cei 2 sunt alti arbori

C₂. $10 \cdot n^2 < 5 \cdot 2^{n-1}$ $/ :5$

$2 \cdot n^2 < 2^{n-1}$ Testăm: $n = 1 \Rightarrow 2 < 1$ $n = 6 \Rightarrow 72 < 32$

$n = 2 \Rightarrow 8 < 2$ $n = 7 \Rightarrow 98 < 64$

$n = 3 \Rightarrow 18 < 4$ $n = 8 \Rightarrow 128 < 128$ → egal

$n = 4 \Rightarrow 32 < 8$ $n = 9 \Rightarrow 162 < 256$ ⇒ ⓒ

$n = 5 \Rightarrow 50 < 16$

D. Inordine -> se parcurge Subarbore stâng, Rădăcină, Subarbore drept

Complexitate: $O(n)$

Container:

    răd : ↑ Nod, rădăcina arborelui

Nod : e: ↑Element, informația utilă

    st: ↑Nod, nodul stâng

    dr: ↑Nod, nodul drept

subalgoritm cnt($\overset{i}{a}$, $\overset{i}{e}$, $\overset{o}{nr}$)

PreCond: a: arbore binar

    e: ↑Element, elementul căutat

Post. Cond nr: Întreg, indicele elementului în inordine

    @aruncă excepție dacă nu există elementul

```
creează(st) {Creează o stivă}
crt ← a.răd
nr ← 0; ok ← Fals
cât timp ¬vidă(st) ∨ crt ≠ NIL execută
    cât timp crt ≠ NIL execută
        adaugă(st, crt)
        crt ← [crt].st
    sfârșit cât timp
    șterge(st, crt)
    Dacă [crt].e = e atunci
        nr ← cnt
        ok ← Adevărat
        STOP
    sfârșit dacă
    crt ← [crt].dr {am parcurs toată partea stângă}
    cnt ← cnt + 1
sfârșit cât timp; Dacă ok = Fals atunci; @Nu există; sfârșit dacă; sfârșit subalgoritm
```