SDA 102

A. $T(n) = \begin{cases} 1, & n = 1 \\ 2 \cdot T(n-1), & \text{altfel} \end{cases}$

$T(1) = 1$

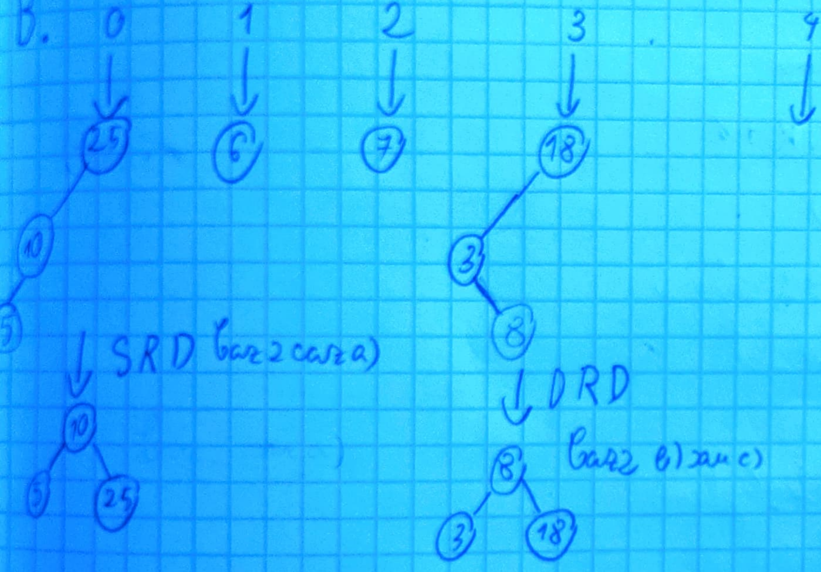
$T(2) = 2$

$T(3) = 4$

$T(4) = 8$

$\Rightarrow T(n) = 2^{n-1} \in \Theta(2^n)$ (pentru ambii timpi)

B.



C. [✓ c] $O(n \cdot \log_2 n)$ deoarece operația adaugă în heap e în cazul defavorabil $\Theta(\log_2 n)$

→ construcția unui heap cu n elemente se execută în $O(n)$

→ extragerea în $O(n \cdot \log_2 n)$ (n ştergeri de $\Theta(\log_2 n)$ caz def.)

C2. [✓ a] → se şterge de pe prima poz $\Rightarrow O(n)$ } trebuie să deplasăm n-1 (n) el. la dreapta

→ se adaugă pe prima poz $\rightarrow O(n)$

D.

Container:

rad: ↑Nod, rădăcina

Nod: st: ↑Nod, vecinul stâng

dr: ↑Nod, vecinul drept

e: TElement, informația utilă

h: Întreg, înălțimea nodului

```
subalgoritm același_nivel(a, e, e')
        Pre: a: arborele binar
             e: TElement
             e': TElement
        Post: același_nivel = { 1, dacă nodurile se află pe același nivel
                              { ¬, dacă nodurile nu se află pe același nivel

creează(c) {c -> coadă}; [a.răd].k = 0
adaugă(c, a.răd); $k_1 \leftarrow -1$; $k_2 \leftarrow -1$
cât timp ¬vidă(c) execută:
        șterge(c, crt)
        Dacă [crt].st != NIL atunci
            [[crt].st].k = [crt].k + 1
            adaugă(c, [crt].st)
            Dacă [crt].e = e
                $k_1 \leftarrow$ [crt].k
            sfârșit dacă
            Dacă [crt].e = e' atunci
                $k_2 \leftarrow$ [crt].k
            sfârșit dacă
        sfârșit dacă
        Dacă [crt].dr != NIL atunci
            [[crt].dr].k = [crt].k + 1
            adaugă(c, [crt].dr)
        sfârșit dacă
sfârșit cât timp
Dacă $k_1 = k_2 \wedge k_1 \neq -1$ atunci
        același_nivel ← Adevărat
altfel
        același_nivel ← Fals
sfârșit dacă
sfârșit subalgoritm
```