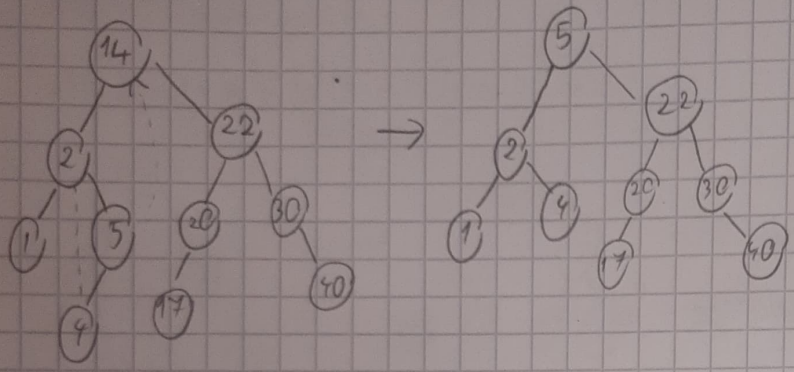
A. $T(n) = \begin{cases} 1, & n = 1 \\ \#T(n-2), & \text{altfel} \end{cases}$

$\Rightarrow T(n) = \underbrace{2 + 2 + \ldots + 2}_{[n/2] \text{ ori}} + 1 = n + 1 \in \Theta(n)$

→ Timpul va fi $\Theta(n)$ în ambele cazuri

B. → o înlocuim cu el maxim din subarborele stâng (de care știm sigur că nu are descendenți) (apoi ștergem respectivul nod)



C. (a) col

C2. (b) → distanța maximă până la o frunză (3 e nivelul, 2 înălțimea)

D. subalgoritm acelasi_nivel(a, e, c)
$R_1 \leftarrow -1$ $R_2 \leftarrow -1$
creează(c)
adaugă(c, {0, 0})
cât timp ¬vidă(c) execută
șterge(c, {i, nivel})
Dacă a.e[i] = e atunci
$R \leftarrow$ nivel
sf.dacă
Dacă a.e[i] = e' atunci
$R_2 \leftarrow$ nivel
sf.dacă
Dacă a.st[i] ≠ -1 atunci
adaugă(c, {a.st[i], nivel+1})
sf.dacă
Dacă a.dr[i] ≠ -1 atunci
adaugă(c, {a.dr[i], nivel+1})
sf.dacă
sf.cât timp
acelasi_nivel $\leftarrow R_1 = R_2 \wedge R_1 \neq -1$

Container:
e: TElement[]
st: Intreg[]
dr: Intreg[]