A. Deduceți timpii mediu si defavorabil pentru următorul subalgoritm. Justificați rezultatul.

```
Functia Operatie(X, n, i) este {:Intreg}
|       {pre: X: vector; n:Intreg; i:Intreg}
|       daca n>1 atunci
|       |       m←[n/2]; S←Operatie(X, m, i-1)
|       |       pentru j=1, n-1 executa
|       |       |       S←S+i
|       |       sfpentru
|       |       Operatie←S+Operatie(x, m, i+1)
|       | altfel
|       |       Operatie←0
|       sfdaca
sfOperatie
```

$$T(n) = \begin{cases} 2T(\frac{n}{2}) + n - 1 & , n > 1 \\ 1, & n \leq 1 \end{cases}$$

$$T(n) = 2T(\tfrac{n}{2}) + n - 1 = 2\left(2T(\tfrac{n}{4}) + \tfrac{n}{2} - 1\right) + n - 1 = 2^2 T(\tfrac{n}{4}) + n - 2 + n - 1$$

$$T(\tfrac{n}{2}) = 2T(\tfrac{n}{4}) + \tfrac{n}{2} - 1$$

$$= 2^2 T(\tfrac{n}{4}) + 2n - (1+2)$$

$$T(\tfrac{n}{4}) = 2T(\tfrac{n}{8}) + \tfrac{n}{4} - 1$$

$$= 2^2 \left[2T(\tfrac{n}{8}) + \tfrac{n}{4} - 1\right] + 2n - (1+2)$$

$$= 2^3 T(\tfrac{n}{8}) + n - 4 + 2n - (1+2)$$

$$= 2^3 T(\tfrac{n}{8}) + 3n - (1+2+4)$$

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + k \cdot n - \left(2^0 + 2^1 + \ldots + 2^{k-1}\right)$$

$$n = 2^k \Rightarrow T(n) = n \cdot T(1) + n \log_2 n - \left(1 \cdot \frac{2^k - 1}{2 - 1}\right)$$

$$\Rightarrow k = \log_2 n$$

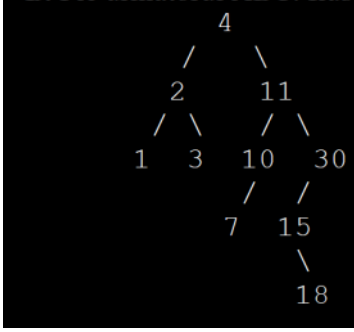
$$= n + n \log_2 n - (n - 1)$$

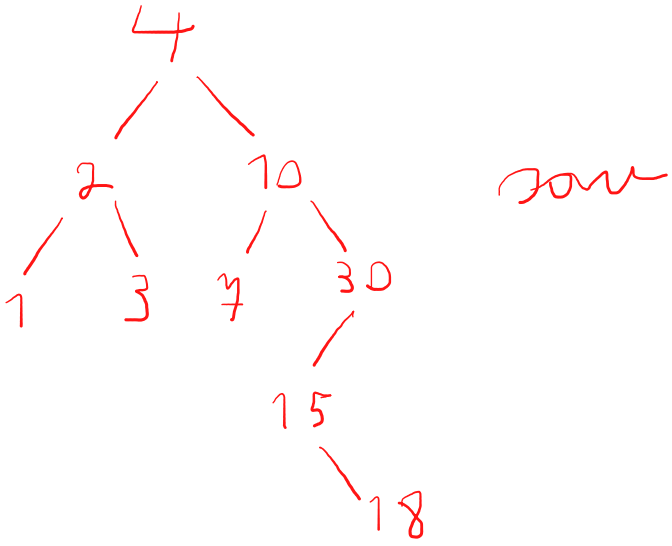
$$= n + n \log_2 n - n + 1$$

$$= n \log_2 n + 1 \in \Theta(n \log n)$$

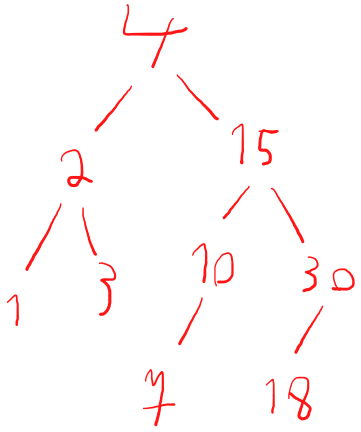
B. Fie următorul ABC. Ilustrați ABC rezultat în urma stergerii cheii 11. Justificați

B. Fie următorul ABC. Ilustrați ABC rezultat în urma ștergerii cheii 11. Justificati





sau



C. Presupunem că se apelează operația de accesare *element* într-o CoadaCuPrioritati având exact 2 elemente cu prioritate egală. Cum se va selecta valoarea returnată? Justificati

a) una aleasa aleator b) cea care a fost inserată prima c) cea care a fost inserată cel mai recent

d) acest lucru nu e posibil (violeaza precondiția)

Se aplică principiul de FIFO

C. Algoritmii pe arbori binari de căutare rulează, în generl, în O(*d*). Cine poate fi *d*? Justificati

a) $\log_2$(numărul de noduri din arbore) b) adâncimea arborelui c) numărul de noduri din arbore d) înălțimea arborelui

d) **Înălțimea arborelui** ✅

- Asta este **drumul cel mai lung de la rădăcină până la o frunză**.
- Este **valoarea maximă posibilă pentru adâncimea unui nod**.
- Într-un arbore binar de căutare, toate operațiile (căutare, inserare, ștergere) merg pe **o ramură**.
- Deci, în cel mai rău caz, se pot face **O(h)** pași, unde **h = înălțimea arborelui**.

🔚 Concluzie:

Algoritmii pe arbori binari de căutare rulează în O(**înălțimea arborelui**), deoarece în cel mai rău caz parcurg un drum complet de la rădăcină până la o frunză.
→ **Răspuns: d) înălțimea arborelui** ✅

Vrei și un exemplu de arbore dezechilibrat și cât ar fi `d`?