A.

Caz favorabil = caz defavorabil = caz mediu $\Rightarrow$

$\Rightarrow$ timpii mediu şi defavorabil vor avea aceeaşi valoare $T(n)$

Functia f are o complexitate de: $\Theta(\log_{10} n) =$

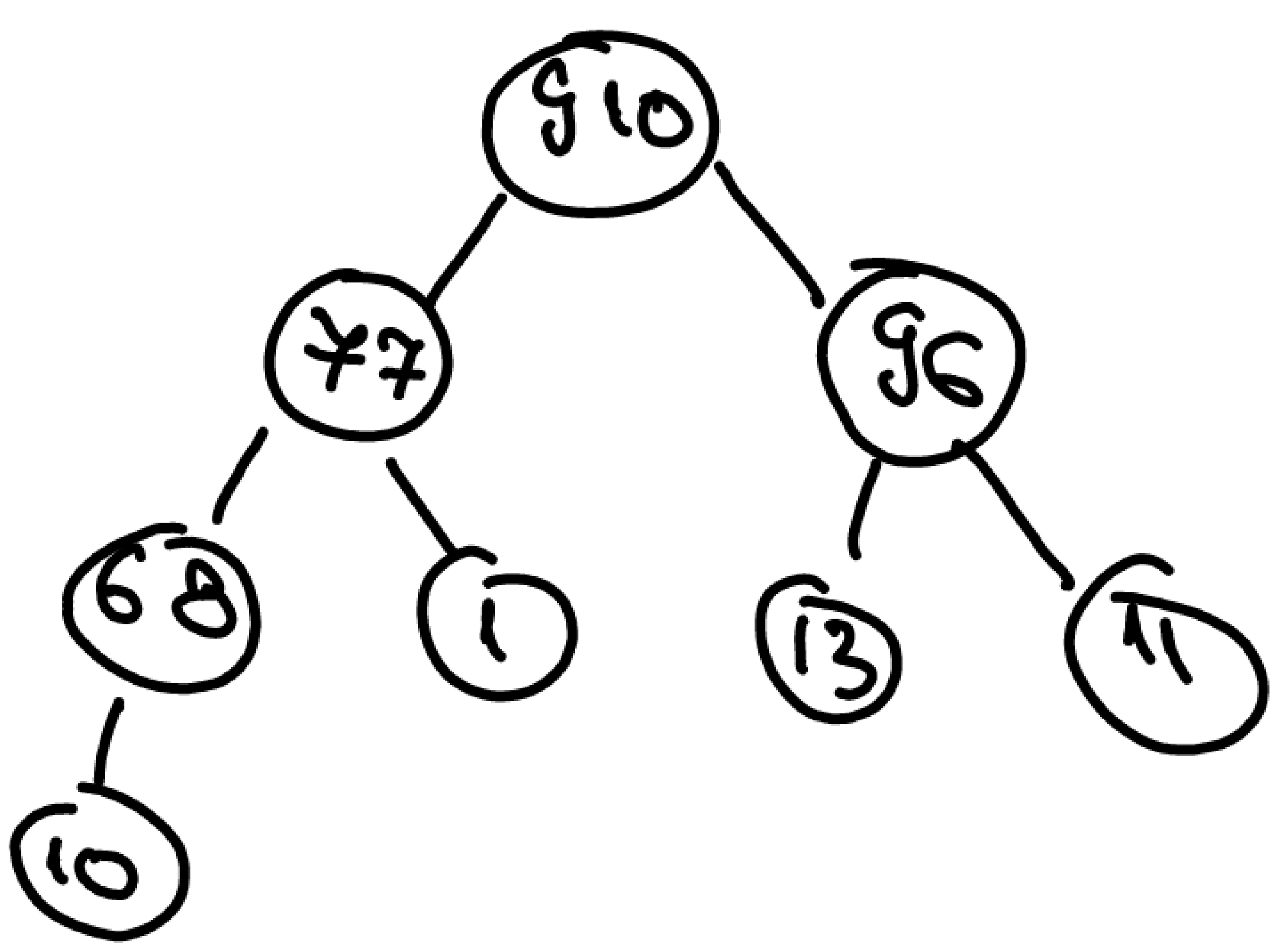
$$= \Theta\left(\frac{\log_2 n}{\log_2 10}\right) = \Theta(\log_2 n)$$

$$T(n) = \sum_{i=1}^{n} f(2i+1) = \sum_{i=1}^{n} (\log_2 (2i+1)) =$$

$$= \log_{10} 3 + \log_{10} 5 + \log_{10} 7 + \dots + \log_{10}(2n+1) =$$

$$= \log_{10}(3 \cdot 5 \cdot 7 \cdot \dots \cdot (2n+1))$$

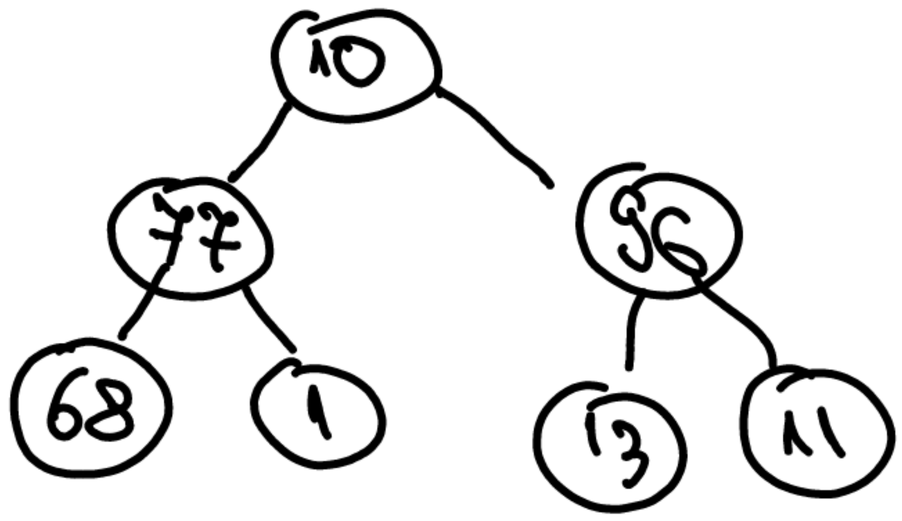
B. Ansamblu inițial



- Se observă că ansamblul dat este un MAX HEAP (elementele cele mai mari au prioritate)
- întotdeauna dintr-un ansamblu se şterge rădă-

cima; aceasta este înlocuită cu ultimul element al ansamblului, respectiv urmând să fie coborât până când se îndeplinesc prop. ansamblului

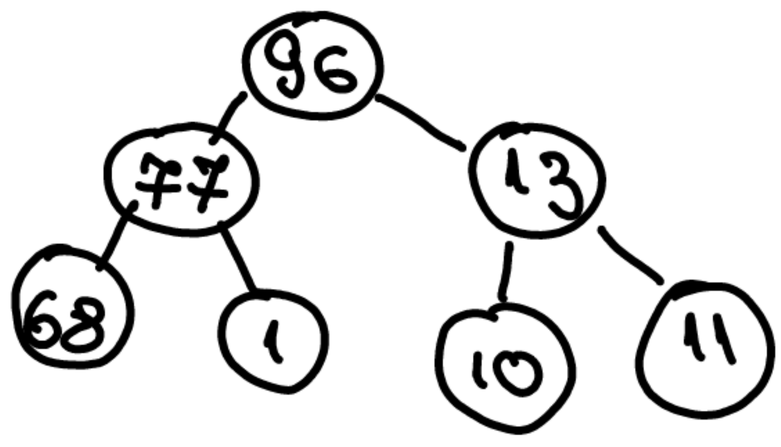
$a_i > a_{2i}$ și $a_i > a_{2i+1}$



$10 < 96$ => coborâm pe 10

$10 < 13$ => coborâm pe 10

=>



c1. a) $O(n)$

- acest lucru se datorează faptului că, deși putem găsi poziția, prin alg. de căutare clasică, în $\log_2 n$ pași, inserarea necesită deplasarea a cel mult n elemente; tocmai pentru că există cazuri defavorabile $\Theta(n)$ și cazuri favorabile $\Theta(1)$ face să avem complexitatea generală de $O(n)$

c2. a) adevărat

Algoritmul va pune întotdeauna elementele din st. primului în cel de egalitate (în timpul interclasării)

D.

Arbore
  e : TElement[ ]
  st : Intreg[ ]
  dr : Intreg[ ]
  r : Intreg (rădăcina)

Subalgoritm înălțime (a, e) este
  {pre: a ∈ Arbore
       e ∈ TElement , e ≠ NULL -TElement
  post : se returnează înălțimea nodului e,
     dacă acesta există în arbore, altfel -1
     @ excepție dacă arborele e vid }

  Dacă a.r = -1 atunci
     @ aruncă excepție "Arbore vid"
  SfDacă
  i ← -1
  Creează (c) { s-a creat o coadă }
  adaugă (c, {a.r, -1}) {se adaugă perechea
                                     indice-nivel}
  CâtTimp ¬ vida(c) execută
     șterge (c, {poz, nivel})
     Dacă a.e[poz] = e atunci
        { se consideră nivelul 0 }
        { începem nr. nivelurilor subgrafului
        cu rădăcina în e }
        nivel ← 0

Dacă nivel > i atunci
  i ← nivel
SfDacă

     SfDacă

```
Dacă a.st[poz] ≠ -1 atunci
    Dacă nivel ≠ -1 atunci
        adaugă (c, {a.st[poz], nivel+1})
    altfel
        adaugă (c, {a.st[poz], -1})
    SfDacă
SfDacă

Dacă a.dr[poz] ≠ -1 atunci
    Dacă nivel ≠ -1 atunci
        adaugă (c, {a.dr[poz], nivel+1})
    altfel
        adaugă (c, {a.dr[poz], -1})
    SfDacă
SfDacă
SfCâtTimp
înălțime ← i
SfSubalgoritm
```

Complexitate:

- de timp: $\Theta(n)$, unde n reprezintă nr. de noduri

presupunând că operațiile cozii: vidă, creează, șterge, adaugă au o complexitate $\Theta(1)$ (astfel, se poate reprezenta prin dublă înlănțuire dinamică: pt. a putea adăuga și șterge în timp constant)

- de spațiu: $\Theta(2^{h-1})$ (spațiu suplimentar, utilizat de coadă), unde h este înălțimea arborelui