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

```
Subalgoritm S(n, i) este
|       {pre: n:Intreg; i:Intreg}
|       daca n>1 atunci
|       |       i←2*i
|       |       pentru j←1,n executa i←i+1 sfpentru
|       |       m←[n/2]
|       |       daca i mod 2=0 atunci  S(m, i-2)
|       |       |  altfel S(m, i-1)
|       |       sfdaca
|       | altfel
|       |       scrie i
|       sfdaca
sfS
```

$$T(n) = \begin{cases} n + T\left(\frac{n}{2}\right), & n > 1 \\ 1, & n \le 1 \end{cases}$$

$$T(n) = n + T\left(\frac{n}{2}\right) = \frac{n}{2^0} + \frac{n}{2^1} + T\left(\frac{n}{\underset{2^2}{4}}\right)$$

$$T\left(\frac{n}{2}\right) = \frac{n}{2} + T\left(\frac{n}{4}\right)$$

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

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

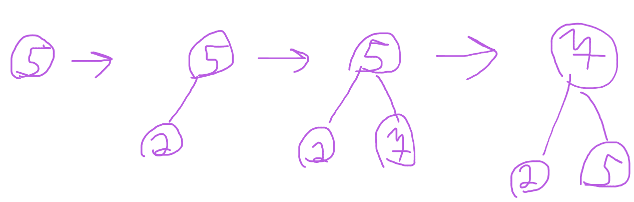
$$= 1 + n\left(1 - \frac{1}{n}\right) \cdot 2 = 1 + 2n - 2 = 2n - 1$$

$\in \Theta(n)$ (CF = CD = CM) din

$$\frac{\frac{1}{2^k} - 1}{-\frac{1}{2}} = \frac{1 - 2^k}{2^k} \cdot \frac{-2}{1} = \frac{2^k - 1}{2^k} \cdot 2 = \left(1 - \frac{1}{2^k}\right) \cdot 2$$

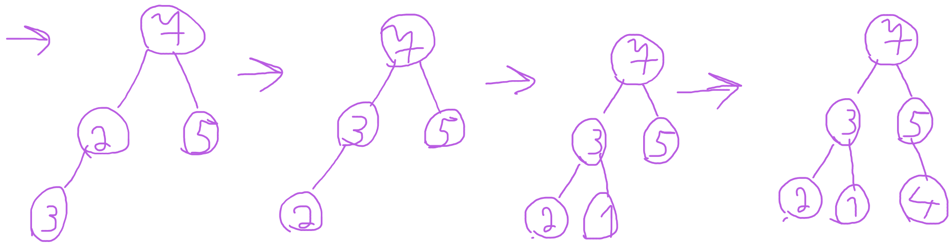
B. Arătați, pas cu pas, aplicarea algoritmului HeapSort pentru ordonarea descrescătoare a vectorului 5, 2, 7, 3, 1, 4. Justificați.

Construim un max-heap cu el.



2. **HEAPSORT.** Sortarea unui vector cu n elemente folosind un ansamblu. Complexitate timp $O(n \log_2 n)$
- se poate *in place*, fără memorarea suplimentară a ansamblului.
  - **Folosind un ansamblu auxiliar** (*out of place*, spațiu suplimentar de memorare $\theta(n)$), ideea este următoarea:
    - Se iau, pe rând, elementele din vector și se adaugă într-un ansamblu $\Rightarrow O(n \log_2 n)$
      - se poate arăta că timpul necesar pentru construcția unui heap cu n elemente este $O(n)$ (a se vedea Observația 2)
    - Se aplică de n ori ștergerea din ansamblul auxiliar și rezultă elementele în ordine $\Rightarrow O(n \log_2 n)$

<u>**Exemplu**</u> Fie vectorul 1, 5, 3, 9, 7. Vrem să îl sortăm descrescător. Construim un **max-heap** cu elementele sale ⇒ ansamblul 9, 7, 3, 1, 5. Apoi scoatem toate elementele din heap și rezultă 9, 7, 5, 3, 1



Pasul 1: Extragi maximul (7)

1. Scoți `7` (rădăcina) → acesta e cel mai mare
2. Pui **ultimul element** (`4`) pe poziția rădăcinii

Obții vectorul:

```python
heap = [4, 3, 5, 2, 1]
```

(Dar acum **nu mai este Max Heap**! Trebuie să îl refaci)

Pasul 2: `heapify` pe rădăcină (index 0)

Structura arborelui e acum:

```markdown
    4
   / \
```

Situația ta:

Ai construit deja un **Max Heap**:

```markdown
      7
     / \
    3   5
   / \   \
  2   1   4
```

Reprezentat ca vector:

```python
heap = [7, 3, 5, 2, 1, 4]
```

C. Una dintre diferențele dintre coadă si stivă este: Justificati

a) cozile necesită liste înlănțuite, iar stivele nu
b) stivele necesita liste înlănțuite, iar cozile nu
c) cozile memorează ambele capete ale structurii, stivele doar un capăt
d) stivele memorează ambele capete ale structurii, cozile doar un capăt

Implementări ale cozilor folosind

- tablouri - vectori (dinamici) - reprezentare circulară (Figura 1, Figura 2).
- liste înlănțuite.
- adăugarea în coadă, ștergerea din caoadă să se facă eficient ($\theta(1)$)

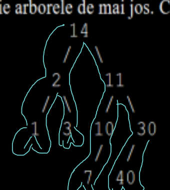
Implementări ale stivelor folosind

- tablouri - vectori (dinamici)
- liste înlănțuite.

2. O *stivă* este o structură liniară de tip listă care restricționează adăugările și ștergerile la un singur capăt (listă LIFO - *Last In First Out*).

2. O *coadă* este o structură liniară de tip listă care restricționează adăugările la un capăt și ștergerile la celălalt capăt (lista FIFO - *First In First Out*).

C. Fie arborele de mai jos. Care este postordinea arborelui? Justificati



1 3 2 7 10 40 30 11 14 POR

14 2 1 3 11 10 7

1 2 3 14 7 10 11 40 30 inordine

a) 1 2 3 7 10 11 14 30 40
b) 1 2 3 14 7 10 11 40 30
c) 1 3 2 7 10 40 30 11 14

D. Cunoscând postordinea și inordinea nodurilor unui arbore binar, să se scrie în Pseudocod sublgoritmul care construiește arborele. Arborele se reprezintă înlănțuit, cu alocare dinamică a nodurilor. Se va indica reprezentarea arborelui și se va preciza complexitatea operației. Folosiți comentarii pentru a ușura înțelegerea soluției.

stânga

preordine (RSD) dreapta

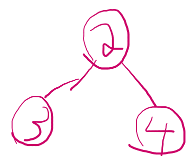
rădăcină

inordine (SRD)

rădăcină

- inordine (SRD)
- postordine (SDR)

D. Cunoscând postordinea și inordinea nodurilor unui arbore binar, să se scrie în Pseudocod sublgoritmul care construiește arborele. Arborele se reprezintă înlănțuit, cu alocare dinamică a nodurilor. Se va indica reprezentarea arborelui și se va preciza complexitatea operației. Folosiți comentarii pentru a ușura înțelegerea soluției.



$2 \leq x$