SD Curs 4

19 Mar 2025

1. Heap
   - inserție
   - min / max
   - extragerea rădăcinii
   - reprezentarea în memorie
2. Heap Sort
3. Arbori binari de căutare

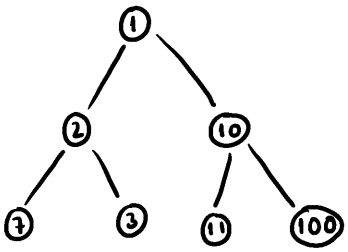
Ce un Heap ?

Două tipuri [min Heap
max Heap

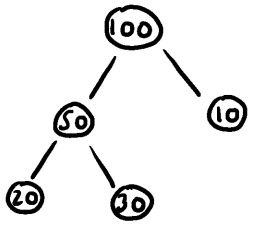
Def

Un min Heap / max - Heap este un arbore binar plin (în care fiecare nod are exact 2 fii cu excepția frunzelor și posibil a unor noduri de pe penultimul nivel) în care fiecare nod are valoarea mai mică (min Heap) / mai mare (max Heap) decât toate nodurile din subarborele său.

Exemple



min Heap



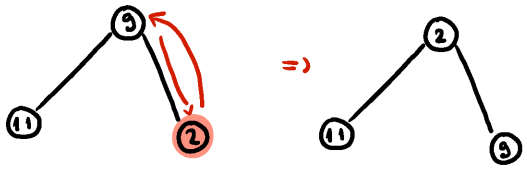
max Heap

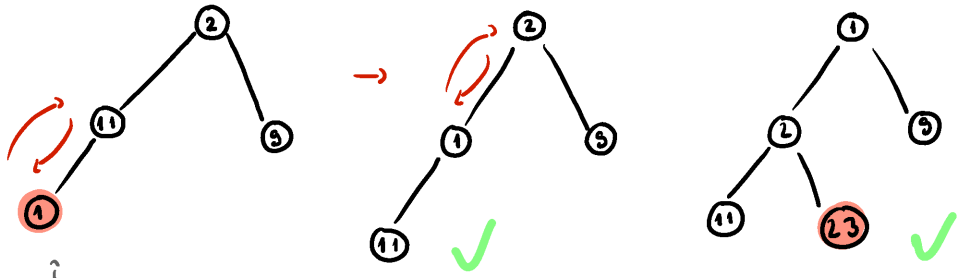
La ce sunt bune Heap-urile ?

- inserție $O(\log n)$ - n = numărul de elemente din Heap
- verificare min / max $O(1)$
- extragerea rădăcinii $O(\log n)$

Inserție

Min Heap



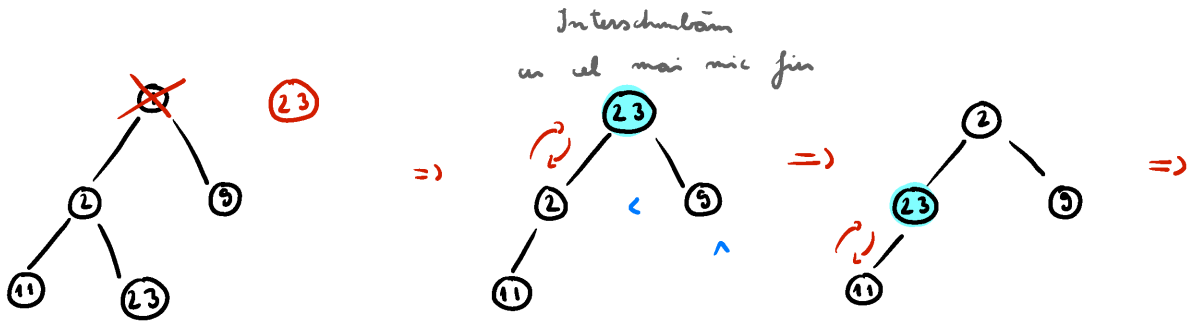


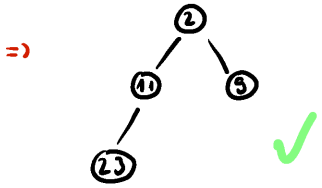
adaugăm 1

=> NU respectă regula de Heap => SWAP

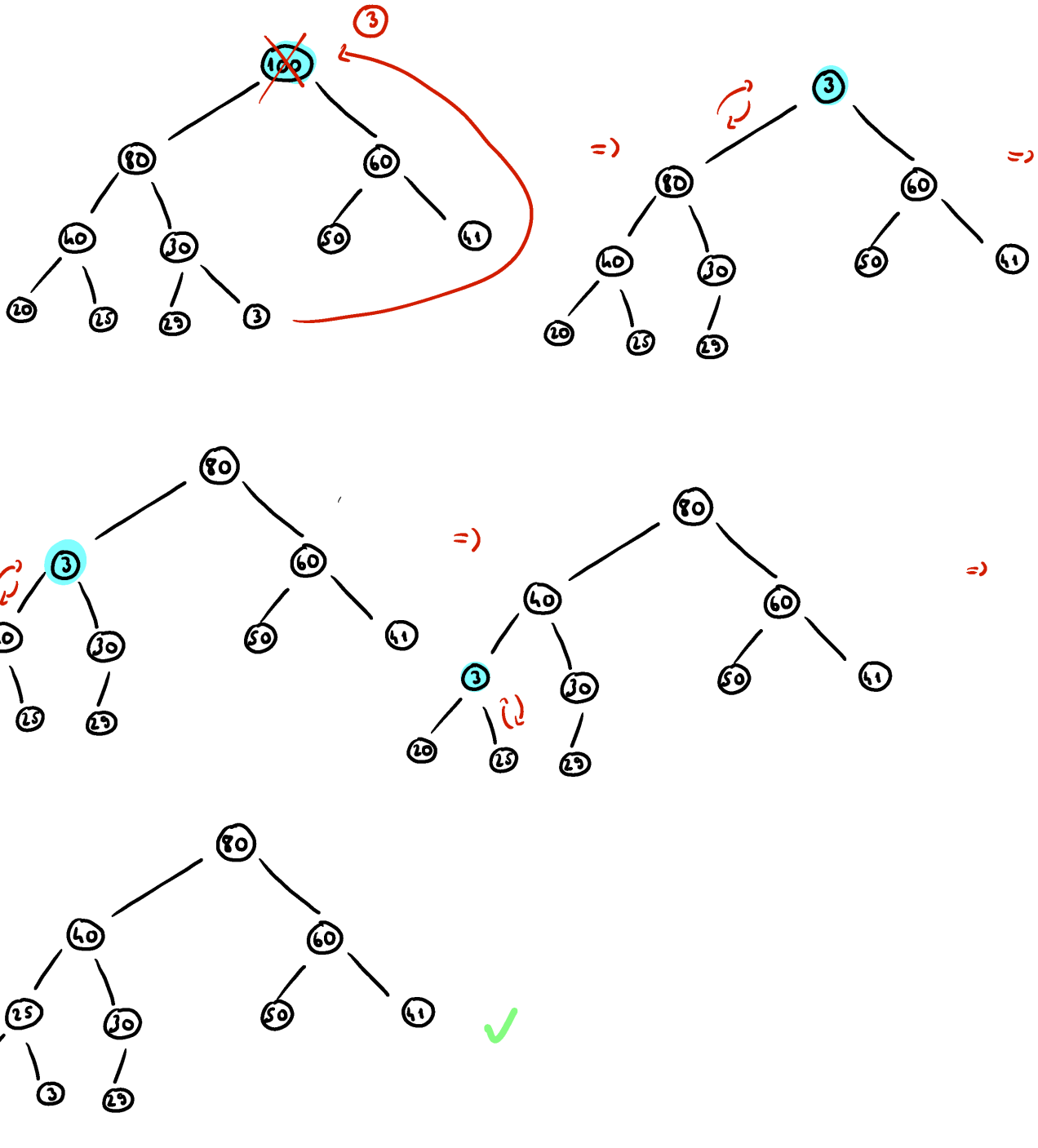
Extragerea rădăcinii

- interschimbăm rădăcina cu ultimul nod adăugat

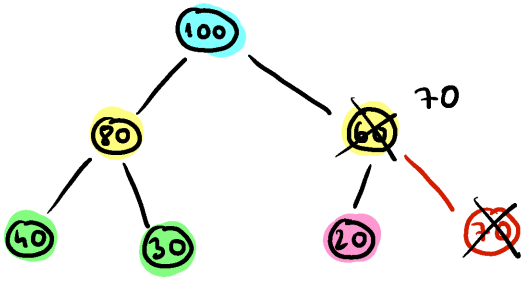


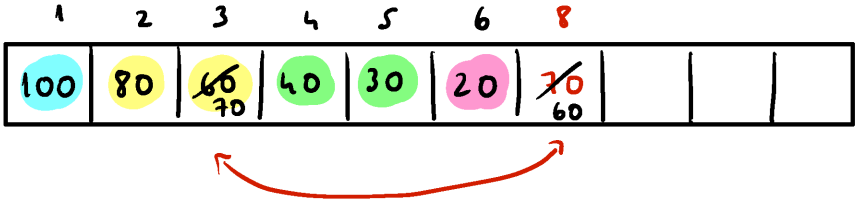


Max Heap


3
100
80
60
40
30
50
41
20
25
29
3
=)
3
80
60
40
30
50
41
20
25
29
=)
=)
80
3
60
40
30
50
41
20
25
29
=)
80
40
60
3
30
50
41
20
25
29
=)
80
40
60
25
30
50
41
20
3
29

Reprezentare în memorie





i → 2 i fiu stâng

2 i + 1 fiu drept

i → $\left[\frac{i}{2}\right]$ părinte

v[7] ? v[3]

v[3] ? v[1]

ok ✓

La ce ne foloseste ?

1. Sortare (Heap Sort) → $O(n \log n)$

n inserări + n extrageri

2. Să se interclaseze K vectori sortați care au în total n elemente

Exemplu

v_1: 1 2 7

v_2: 3 9

v_3: 4 6 8 10

$K = 3$

$n = 9$

$\Rightarrow$ 1 2 3 4 6 7 8 9 10

Sol:

1. Sortăm numerele : $O(n \log n)$

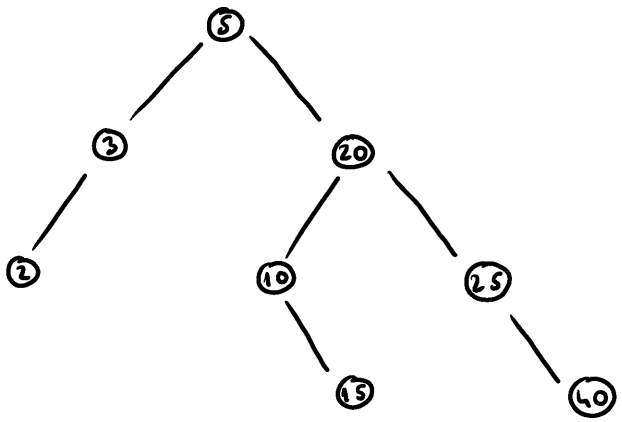
2. Interclasăm vectorii pe rând $O(n K)$

3. Inserăm primele nr. din fiecare vector, apoi extragem minimul și inserăm următorul nr. din vectorul minimului $O(n \log K)$

Arbori binari de căutare

Def Un arbore binar de căutare este un arbore binar în care fiecare nod este mai mare decât toate nodurile din subarborele stâng și mai mic decât toate nodurile din subarborele drept

Exemplu



Operații

1. Căutare — O(n) - Worst Case
2. Inserare — O(n) - Worst Case
3. Min / Max
4. Succesor / Predecesor
5. Ștergere

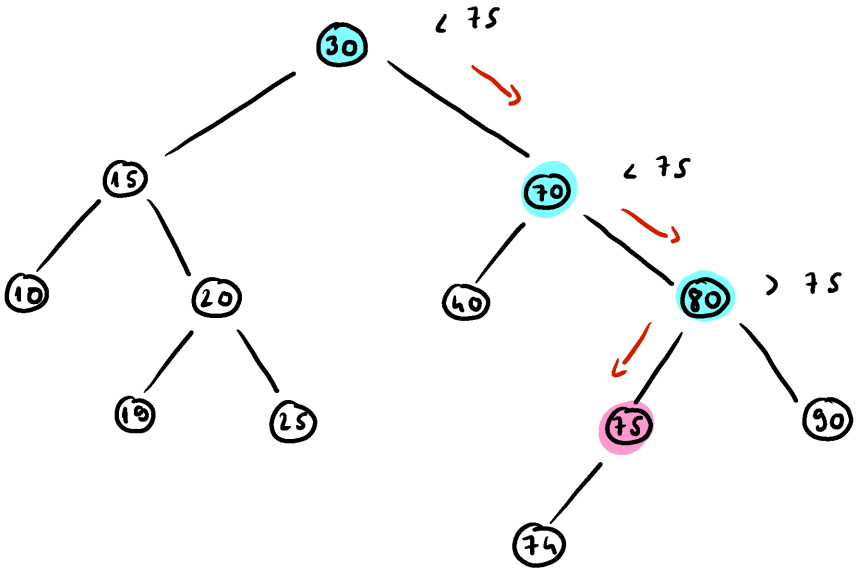
Căutarea

Caut 75

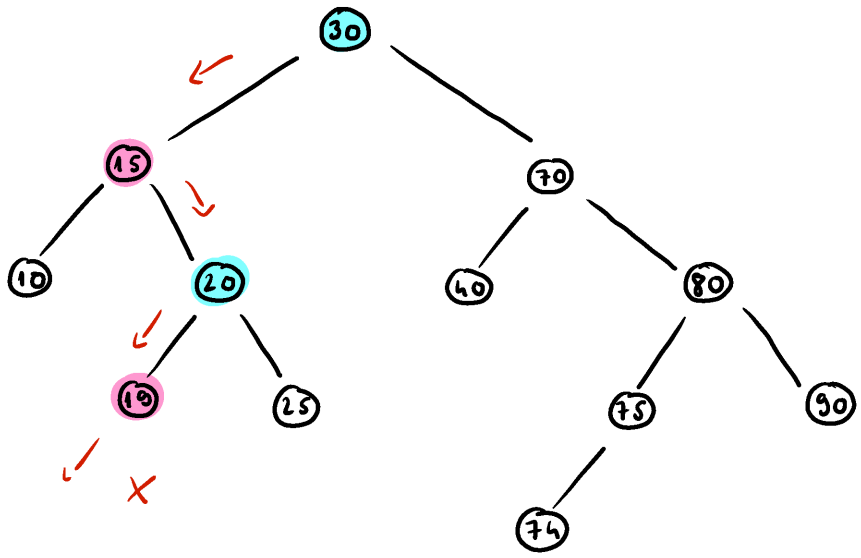
- dacă x este mai **mare** decât nodul

merg în **dreapta**

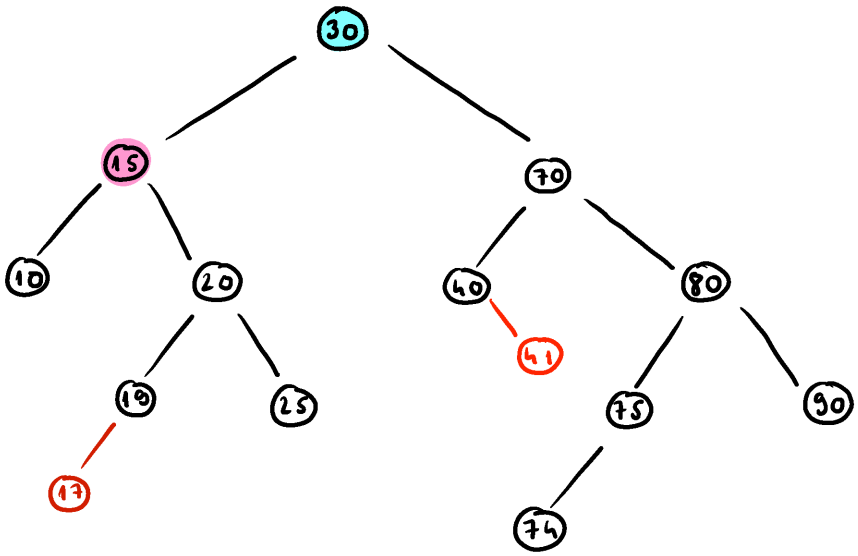
- **dacă stânga**



Caut 17



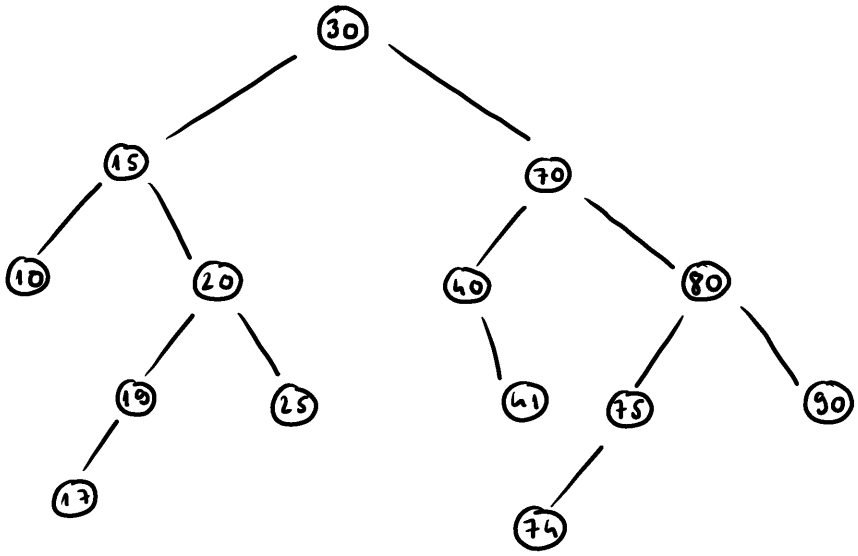
Inserția



Predecesor / Succesor

Predecesor (x) = cel mai mare număr mai mic decât x

Succesor (x) = cel mai mic număr mai mare decât x



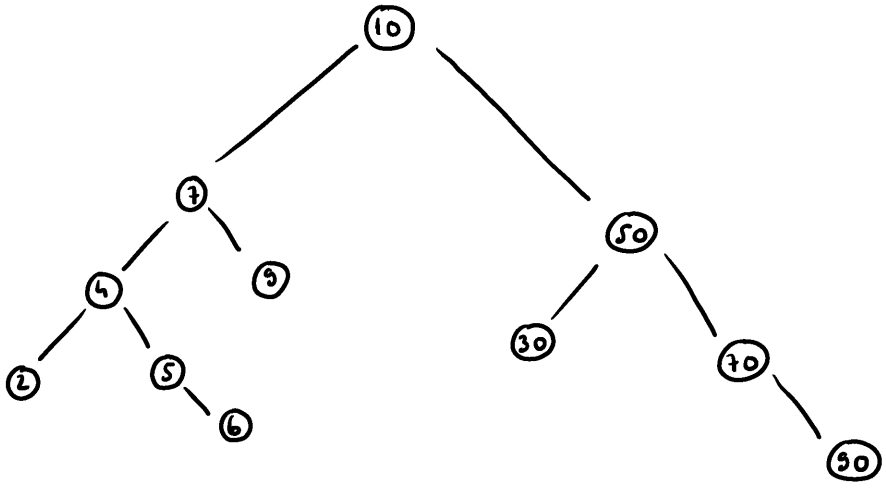
Pred (19) = 17

Succ (70) = 74

Succesor (75) = 80

Succesor (25) = 30

26 Mar 2025



Succesor (6) = 7 ↖ ↖ ↗

Succesor (9) = 10 ↖ ↗

Succesor (30) = 50 ↘

Algoritm

Succesor (x) :

1. Dacă x are fiu drept atunci succesorul va fi minimul din subarborele drept

2. Dacă nu are fiu drept, urcăm în arbore până când dăm de un nod y care este fiu stâng al altui nod, fie el z.
A tunci succesor (x) = z

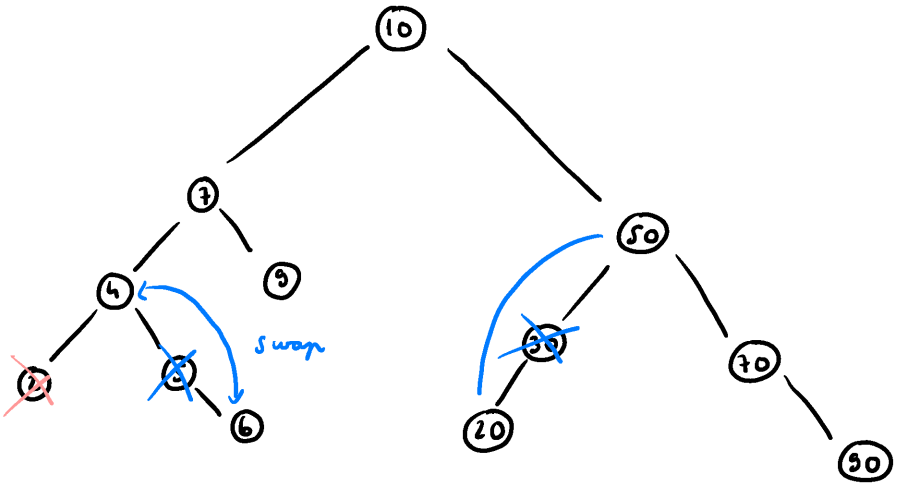
Ștergere

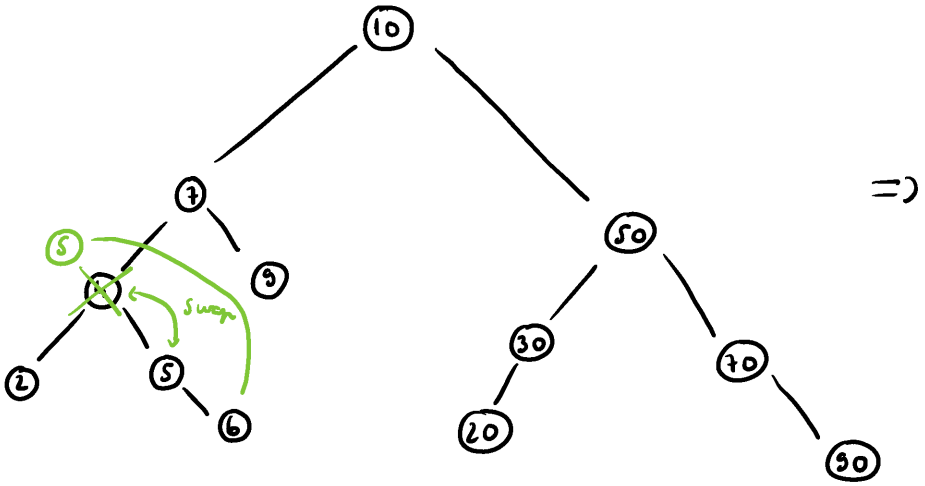
Ștergere (x)

Caz 1 : x nu are fii

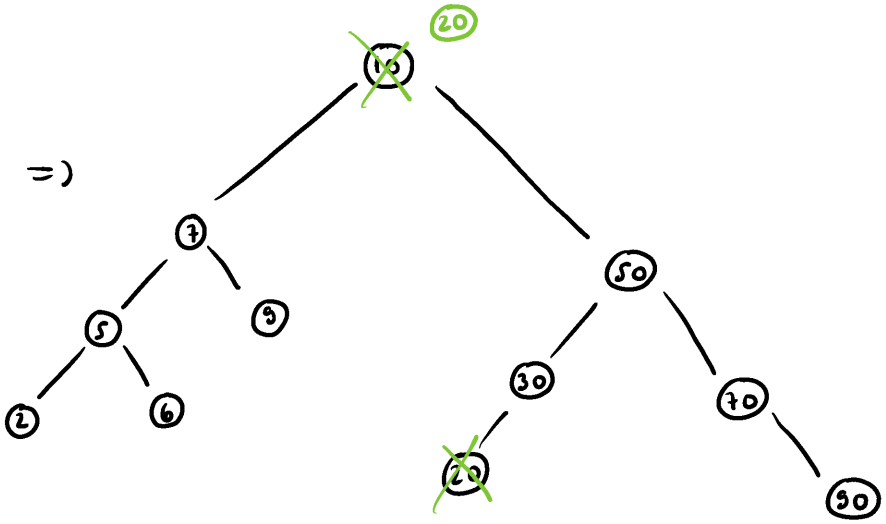
Caz 2 : x are doar fiu stâng
sau doar fiu drept

Caz 3 : x are ambii fii





=)


=)
10
20
7
50
5
9
30
70
2
6
20
50