Cursul nr. 8

STRUCTURI DE DATE

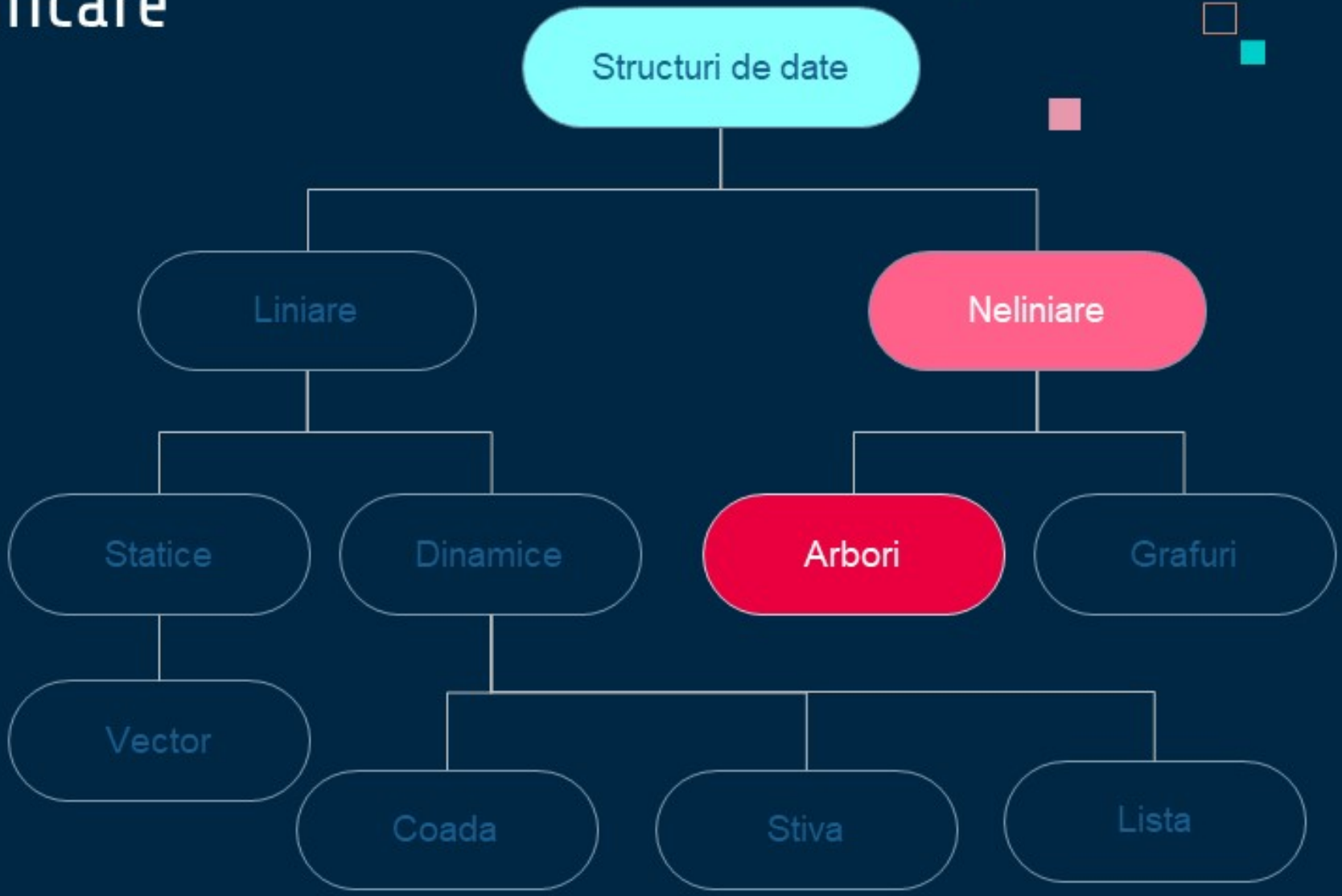
neliniare

Arbore Binar de Cautare

– ABC –

Lector dr. Dorin IORDACHE

Clasificare



Agenda

01

Proprietăți

02

Parcurgeri

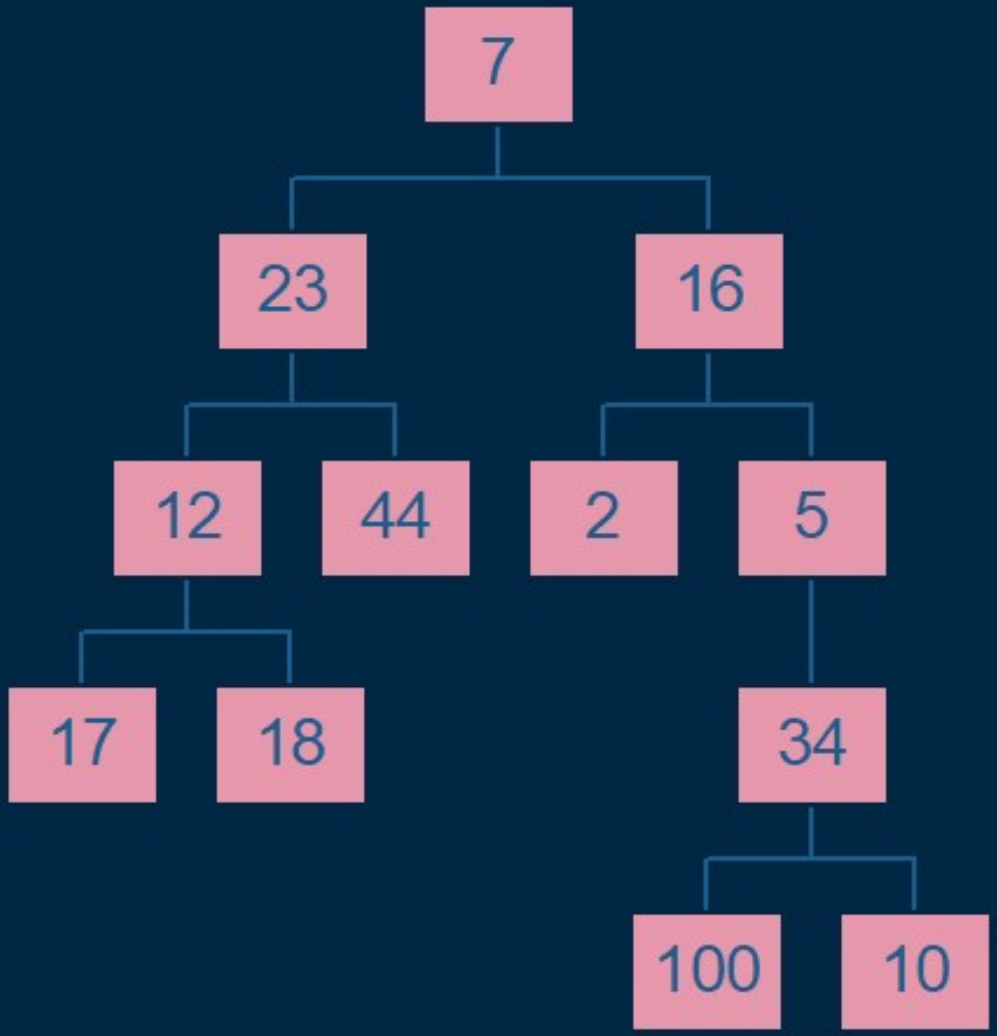
03

Operații

Proprietăți

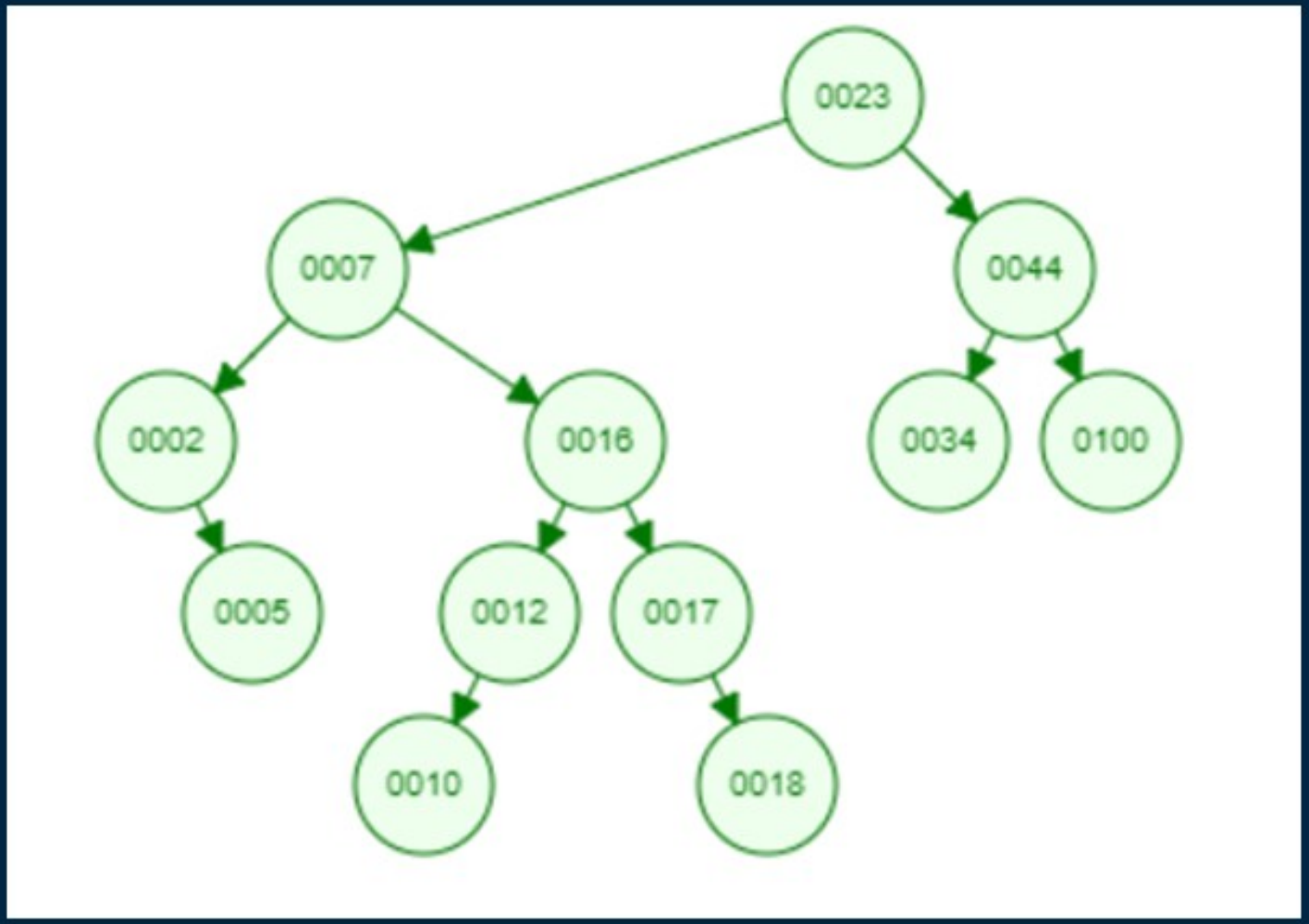
01

AB



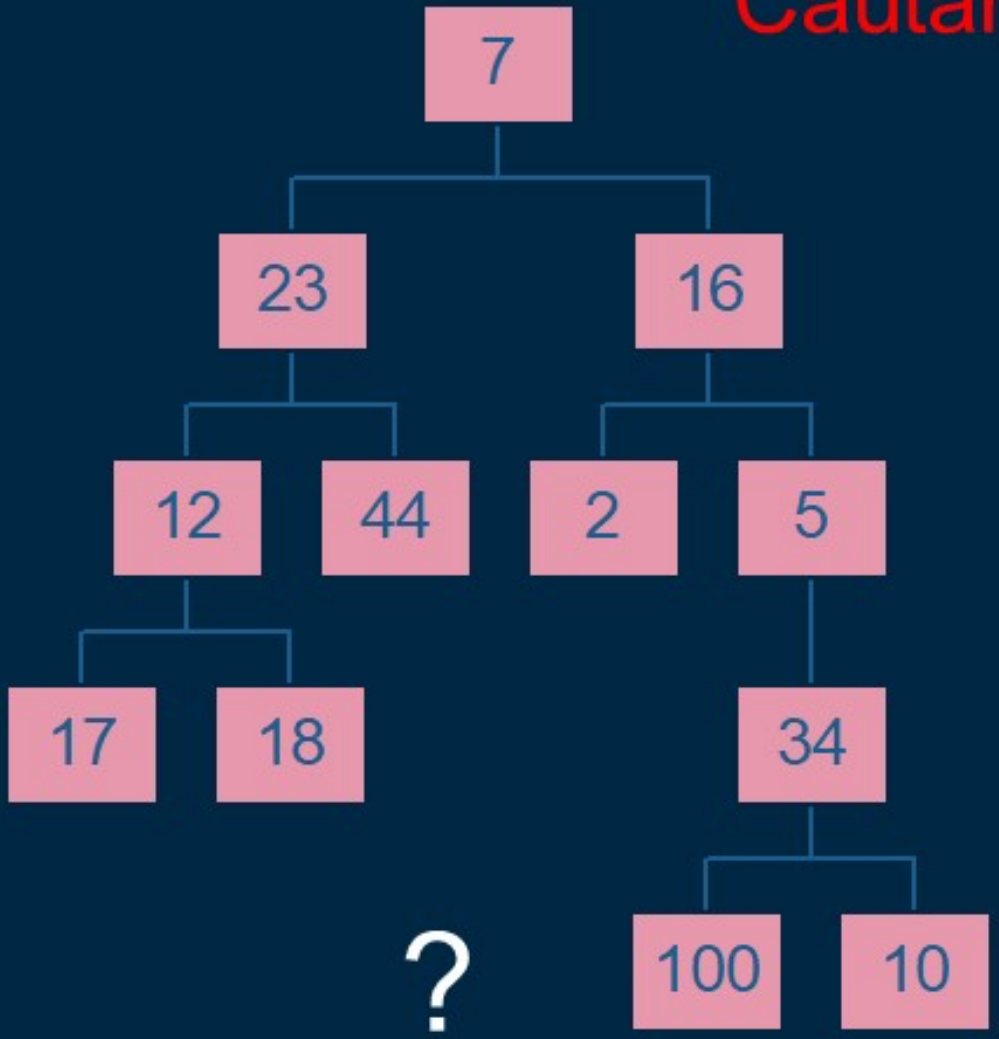
23,7,44,2,16,34,1005,12,17,18,12,10,100,34

ABC

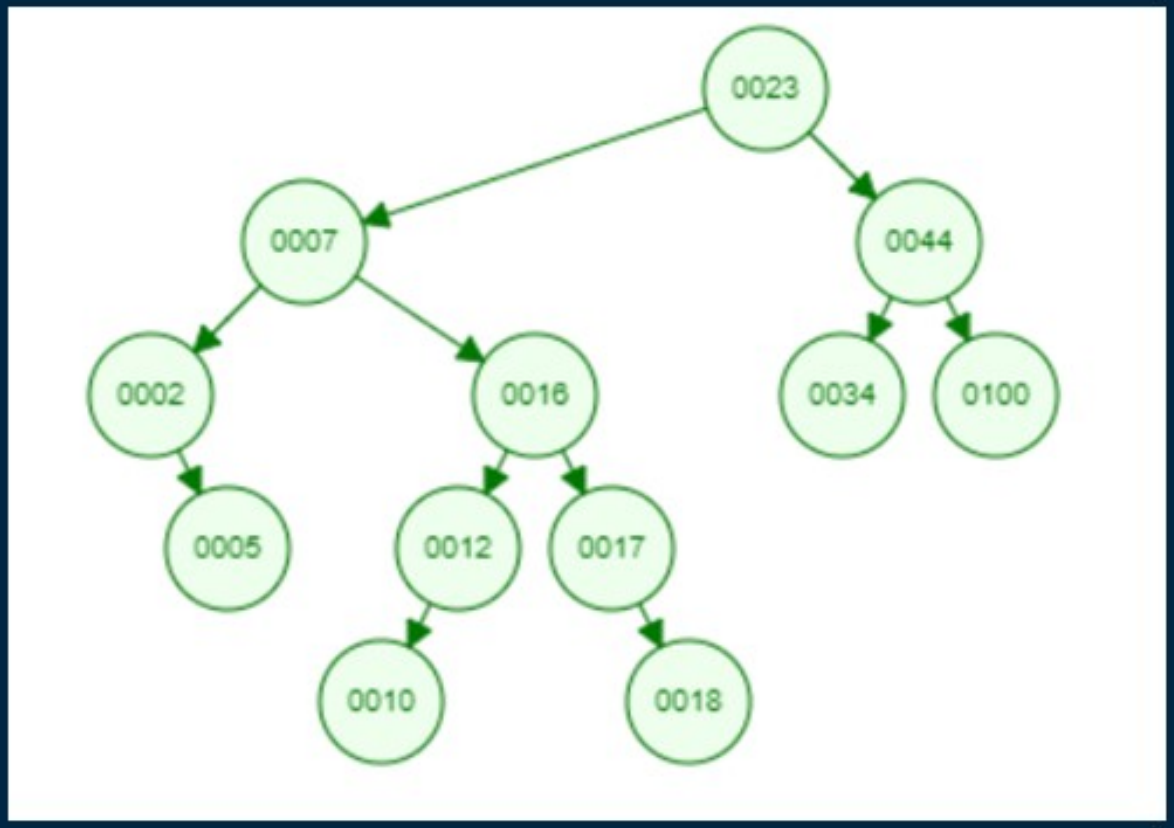


23,7,44,2,16,34,1005,12,17,18,12,10,100,34

Cautare ? 100 ?


7
23
16
12
44
2
5
17
18
34
100
10

?


0023
0007
0044
0002
0016
0034
0100
0005
0012
0017
0010
0018

?

ABC – proprietăți

ABC = un arbore binar

definit prin faptul ca fiecarui nod x i se atribuie o cheie care indeplineste urmatoarele proprietati:

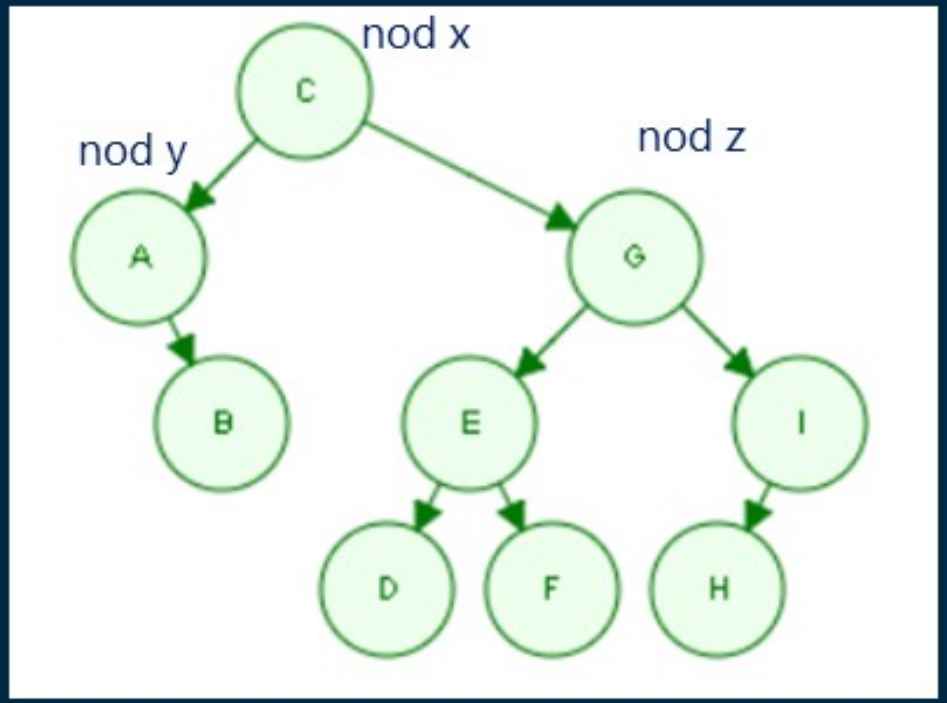
1. cheie(y) < cheie(x), oricare ar fi y un nod din subarborele stang al nodului x;

2. cheie(z) >= cheie(x), oricare ar fi z un nod din subarborele drept al nodului x.
3. Orice sub-arbore este Arbore Binar de Căutare

ABC – proprietăți

arborele binar de cautare de mai jos, se obtine introducand in ordine urmatoarele chei:

C, A, B, G, E, F, D, I, H



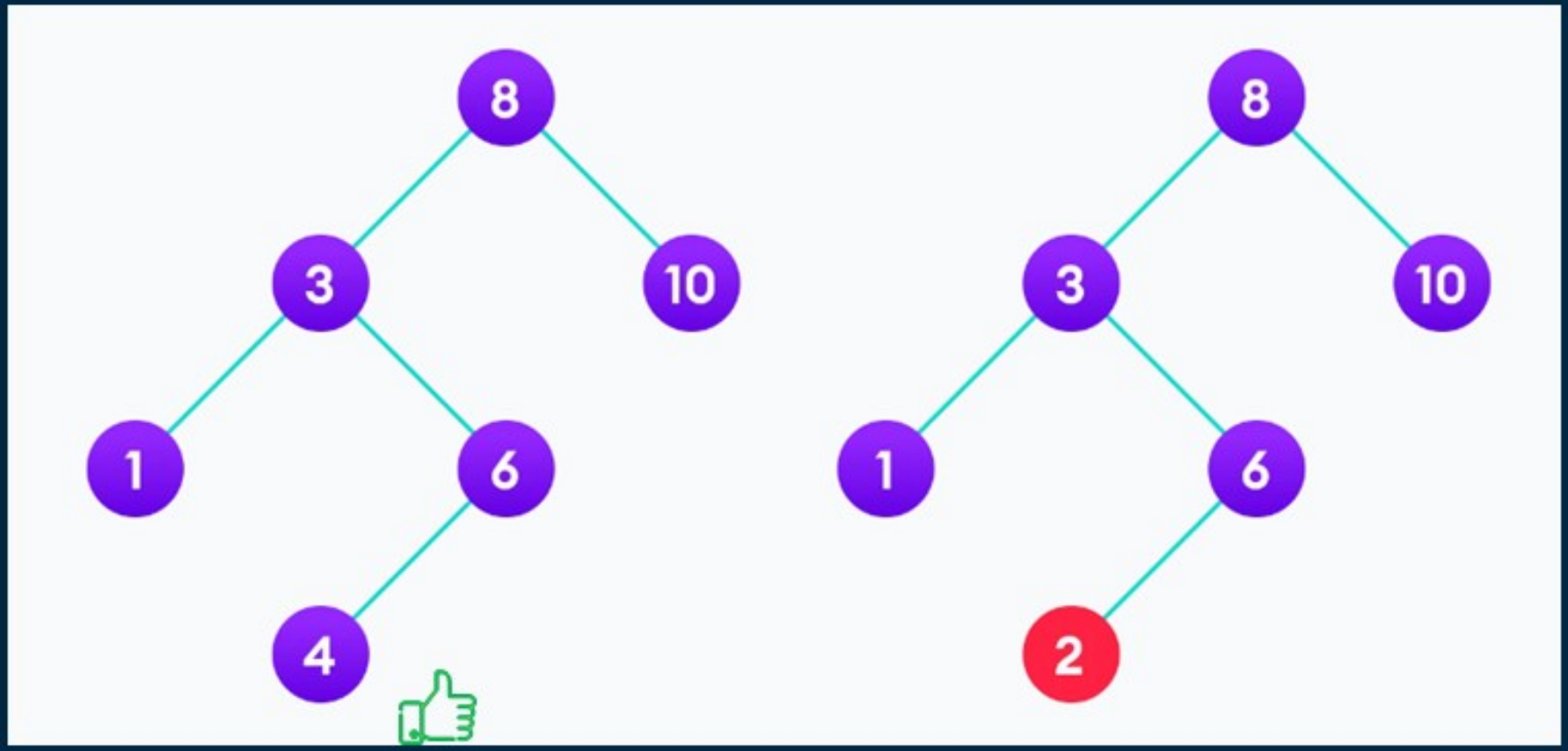
Pentru orice nod: x, y, z sunt îndeplinite relațiile:

1. **cheie(y) < cheie(x)**

2. **cheie(z) >= cheie(x)**

3. **Orice sub-arbore este ABC**

ABC ?



Parcurgeri

02

Parcurgerea ABC

Efectuarea oricarei operații pe un arbore, necesita accesarea unui nod specific.

Algoritmul de traversare a arborelui – solutia

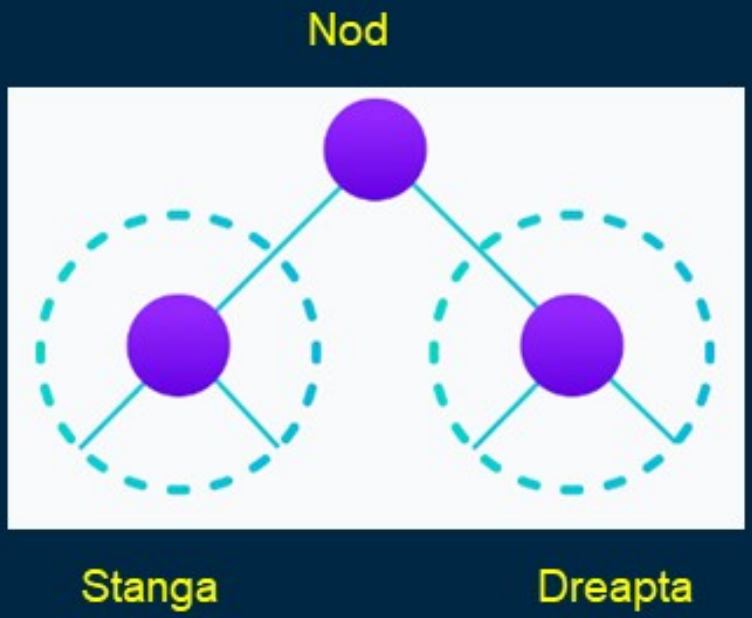
Tipuri de parcurgeri:

- InOrdine
- PreOrdine
- PostOrdine

InOrdine (SND, LNR)

Pas 1: Se parcurg toate nodurile din subarborele din stânga
Pas 2: Apoi nodul rădăcină
Pas 3: Se parcurg toate nodurile din subarborele din dreapta

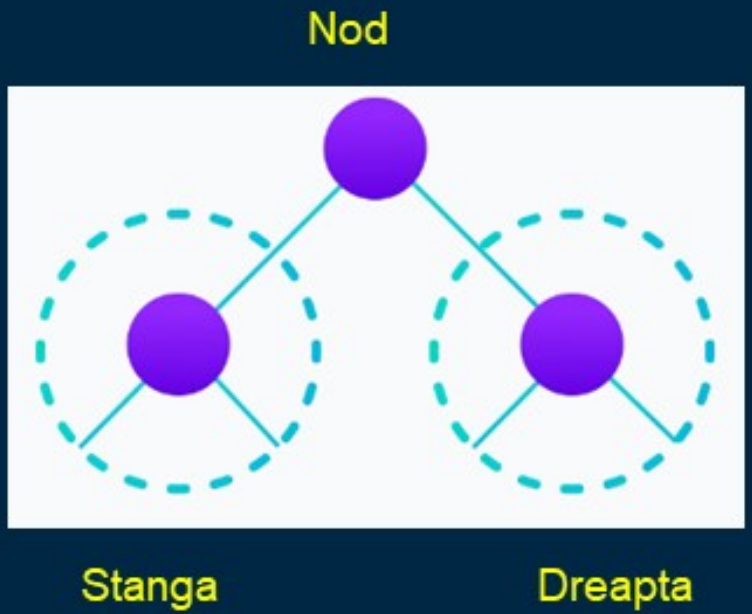
```
InOrdine(root->stanga)
afisare(root->data)
InOrdine(root->dreapta)
```



PreOrdine (NSD, NLR)

Pas 1: Se viziteaza nodul radacina
Pas 2: Se parcurg toate nodurile din subarborele din stânga
Pas 3: Se parcurg toate nodurile din subarborele din dreapta

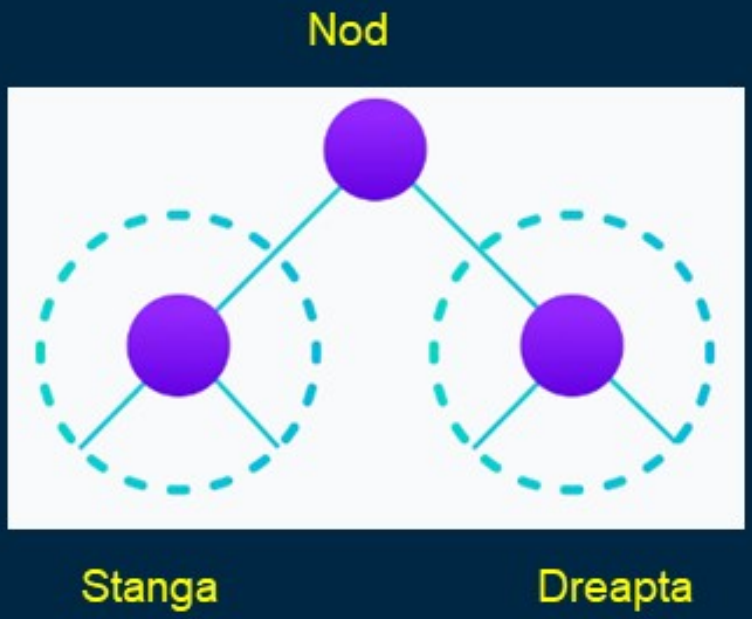
```
afisare(root->data)
PreOrdine(root->stanga)
PreOrdine(root->dreapta)
```



PostOrdine (SDN, LRN)

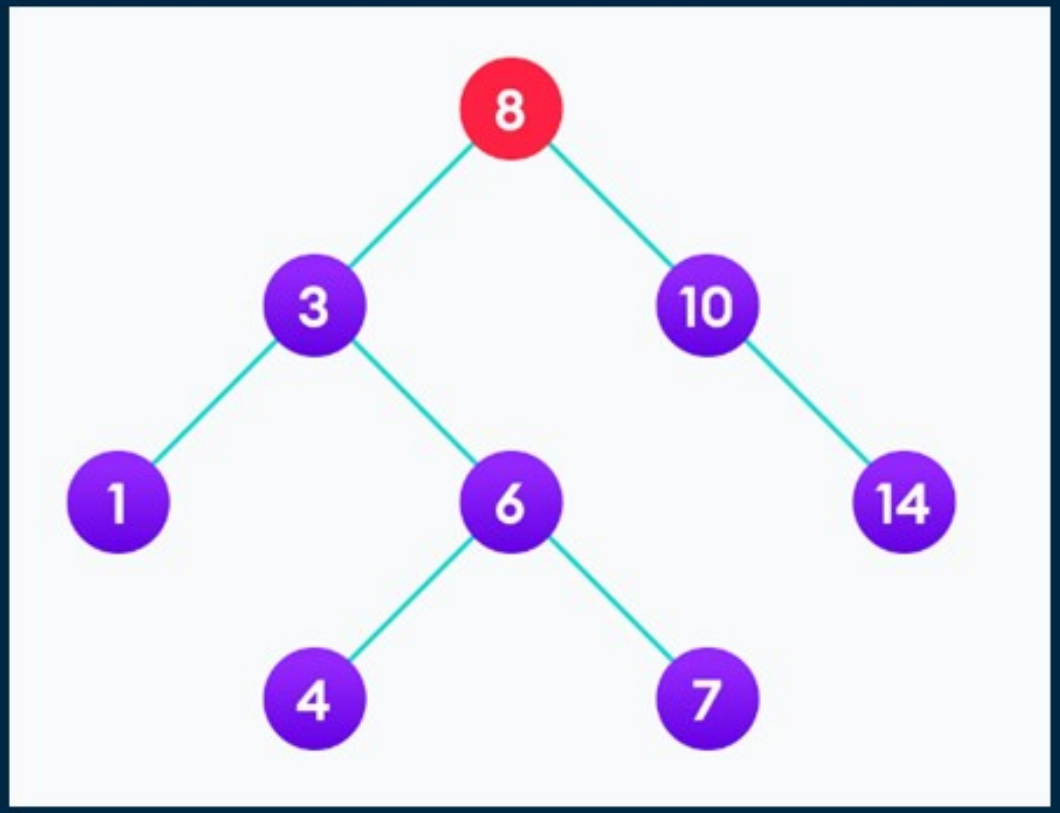
Pas 1: Se parcurg toate nodurile din subarborele din stânga
Pas 2: Se parcurg toate nodurile din subarborele din dreapta
Pas 3: Se viziteaza nodul radacina

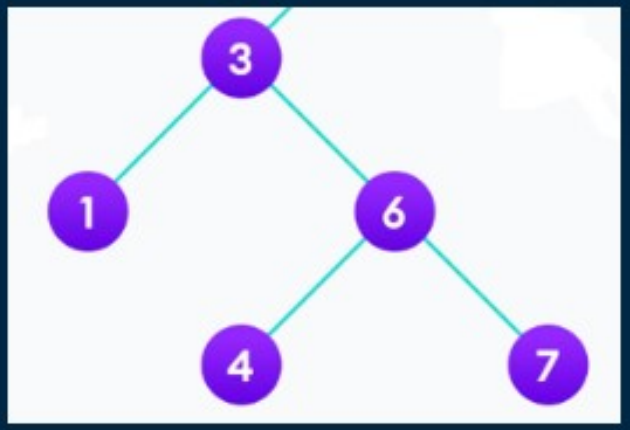
```
PostOrdine(root->stanga)
PostOrdine(root->dreapta)
afisare(root->data)
```

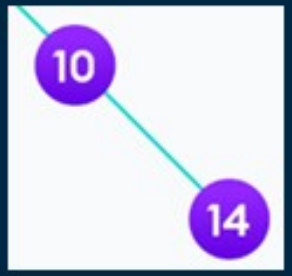


Parcurgeri - exemplu

InOrdine (SND)

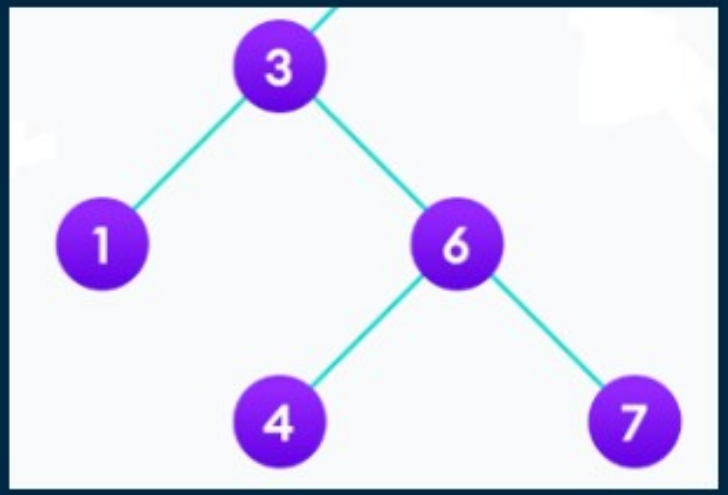


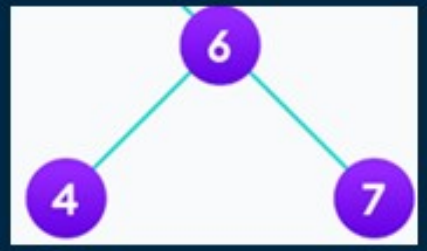


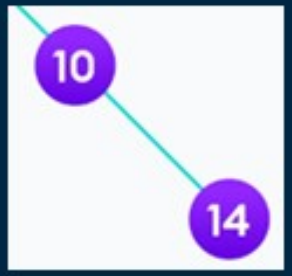


Parcurgeri - exemplu

InOrdine (SND)

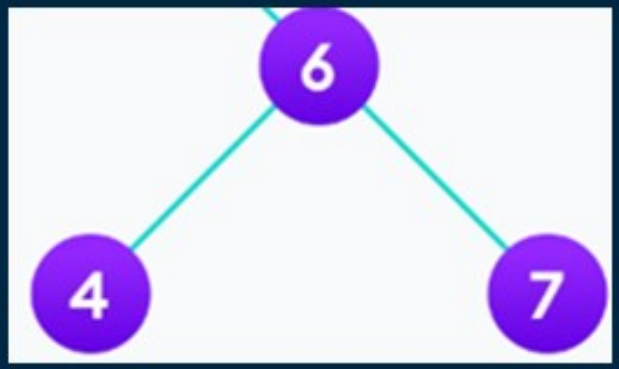


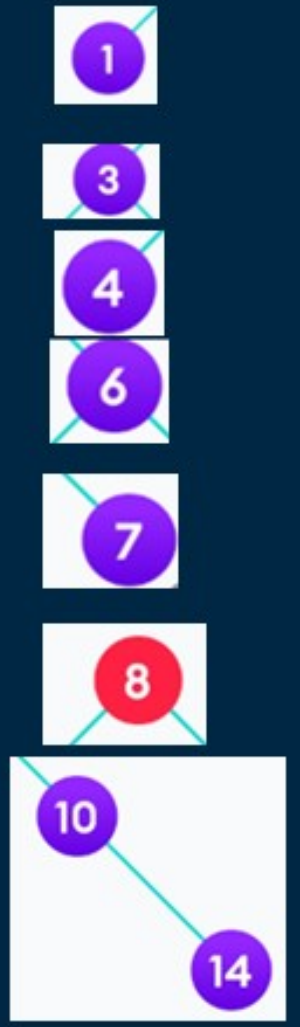




Parcurgeri - exemplu

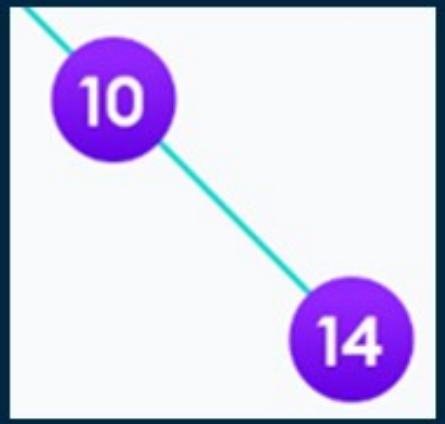
InOrdine (SND)





Parcurgeri - exemplu

InOrdine (SND)



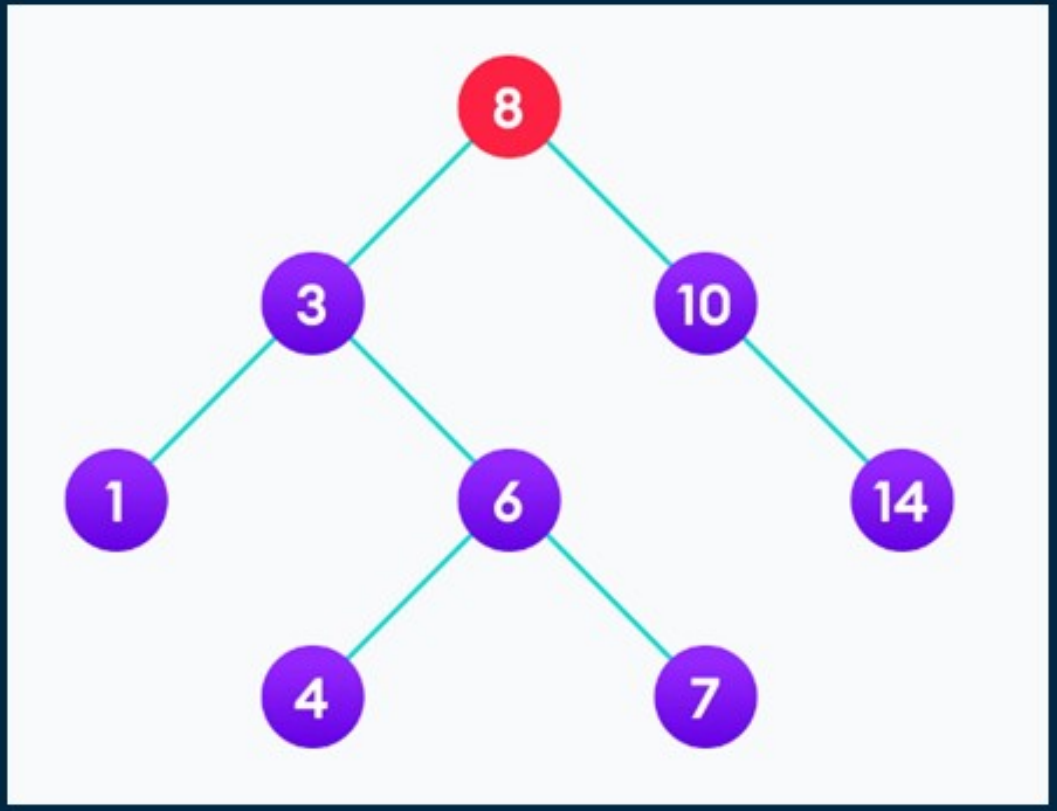
Rezultat parcurgere InOrdine:

1 -> 3 -> 4 -> 6 -> 7 -> 8 -> 10 -> 14

Parcurgere InOrdine (SND) A.B.C. = ordonare crescătoare

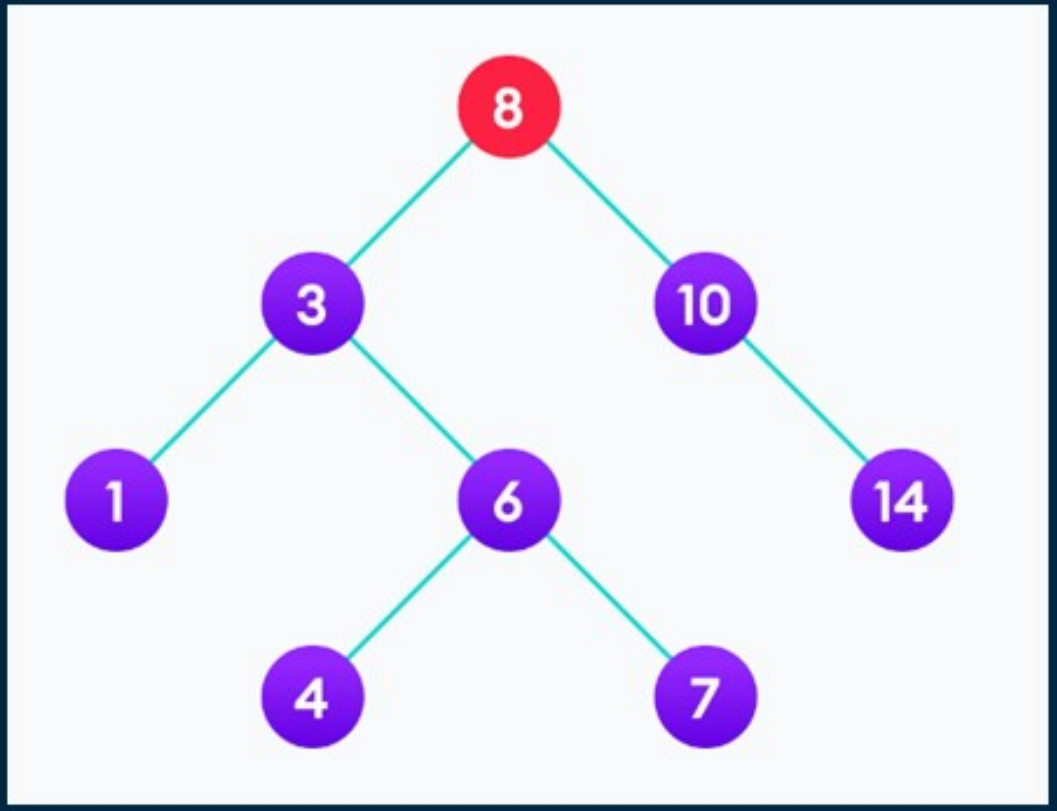
Schimbând ordinea de parcurgere DNS, obținem ordonarea elementelor în criteriul invers

Parcurgeri - ?



PreOrdine (NSD)

Parcurgeri - ?



PostOrdine(DSN)

Operații – ABC –

03

Operații Arbori Binar de Căutare

- Căutare
- Adăugare
- Ștergere

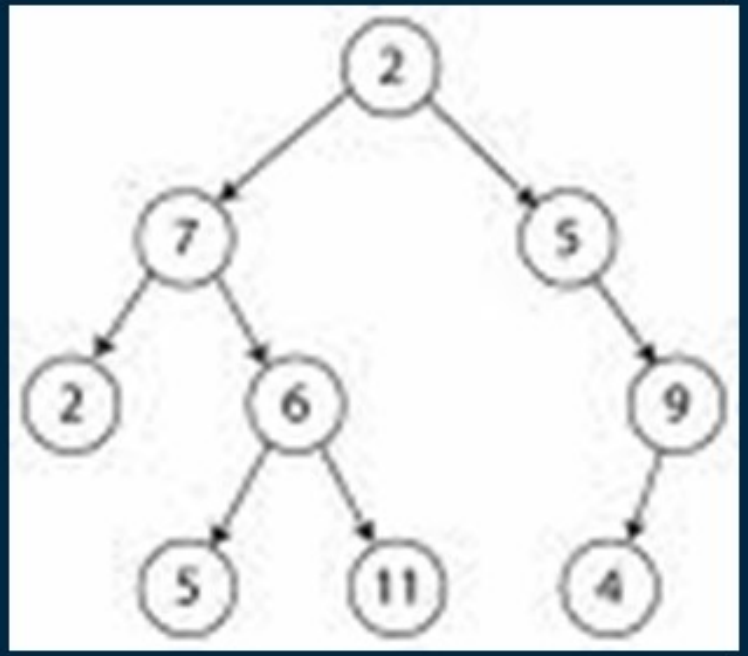
Operații Arbori Binar de Căutare

- Căutare

1) Pornim de la radacina
(2) Cautam in AB nivel cu nivel pana gasim un element sau am ajuns la o frunza.

Este aceasta cautare mai buna decat cautarea intr-o lista inlantuita?

Arbore Binar



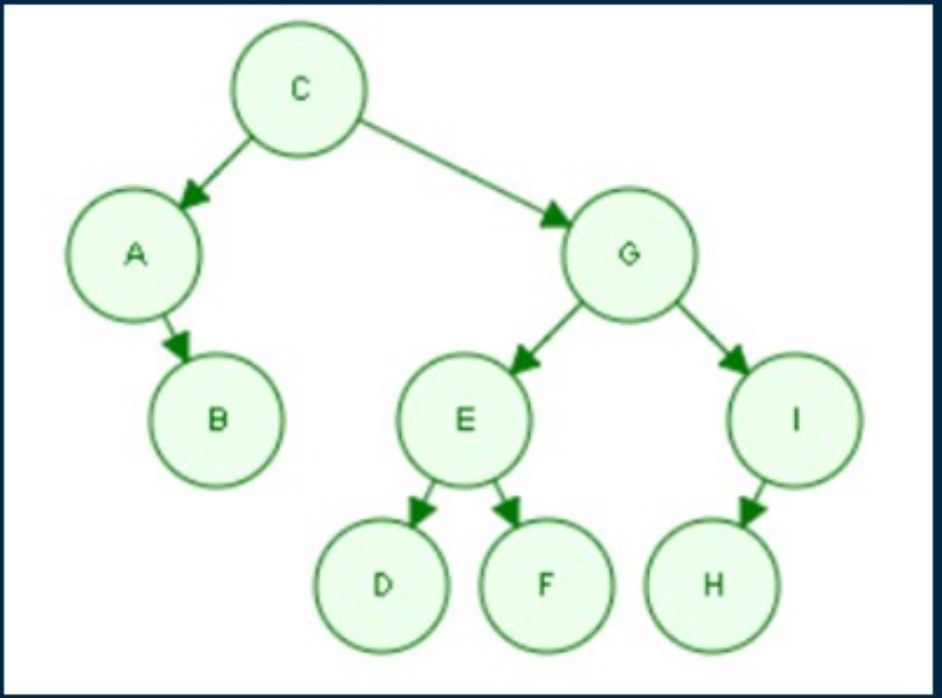
Operații Arbori Binar de Căutare

- Căutare

(1) Pornim de la radacina
(2) Comparăm valoarea de căutat cu cheia nodului.
Dacă valoarea de căutat este mai mică decât cheia nodului, vom continua căutarea în subarborele stâng,
În caz contrar, în subarborele drept
(3) Se continuă până când găsim valoarea de căutat sau nu mai există subarbori

Este aceasta cautare mai buna decat cautarea intr-o lista inlantuita?

Arbore Binar de Căutare



Operații Arbori Binar de Căutare

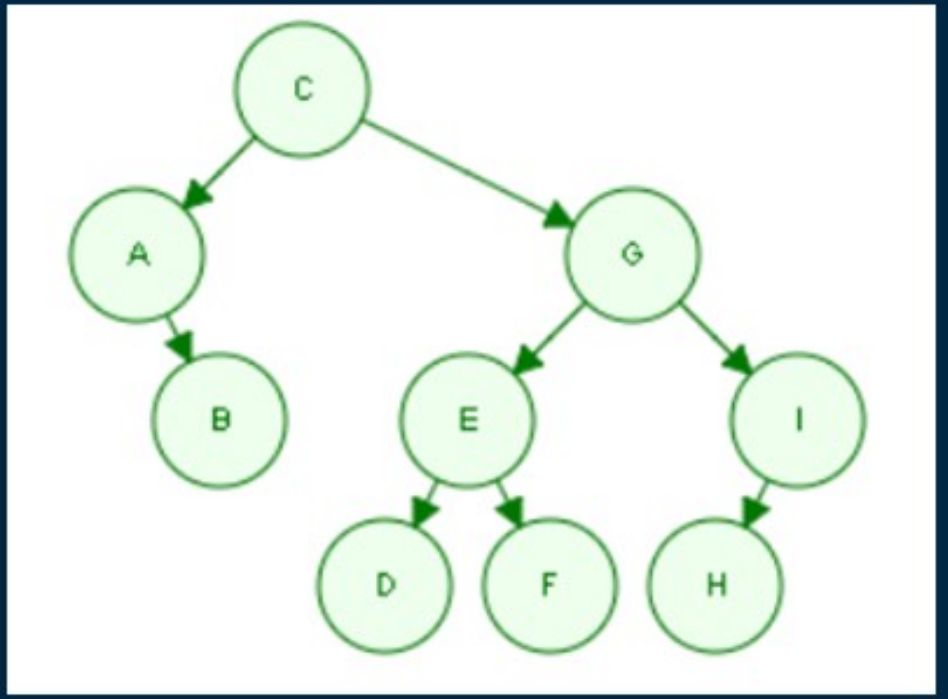
- Căutare

Exercițiu:

1. căutăm valoarea E
2. căutăm valoarea K

?

Arbore Binar de Căutare



Operații Arbori Binar de Căutare

- Căutare

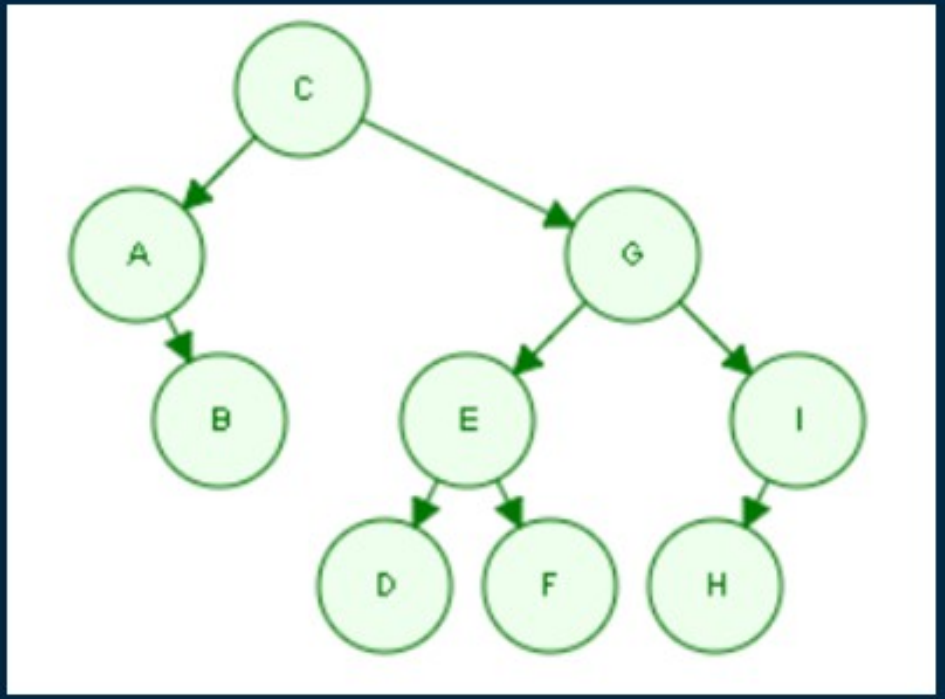
```
begin procedure caut
        if root == NULL then
                return NULL
        endif
        if valoare == root->data then
                return root->data
        endif
        if valoare < root->data then
                return caut(root->stânga)
        else
                return caut(root->dreapta)
        endif
end procedure caut
```

Operații Arbori Binar de Căutare

- Adăugare

(1) Pornim de la radacina
(2) Comparăm valoarea de adăugat cu cheia nodului.
Dacă valoarea este mai mică decât cheia nodului, vom continua procedeul în subarborele stâng,
În caz contrar, în subarborele drept
(3) Se continuă până când găsim un nod extern (frunză) si vom adăuga, în subramura stânga, sau dreapta, după caz.

Arbore Binar de Căutare



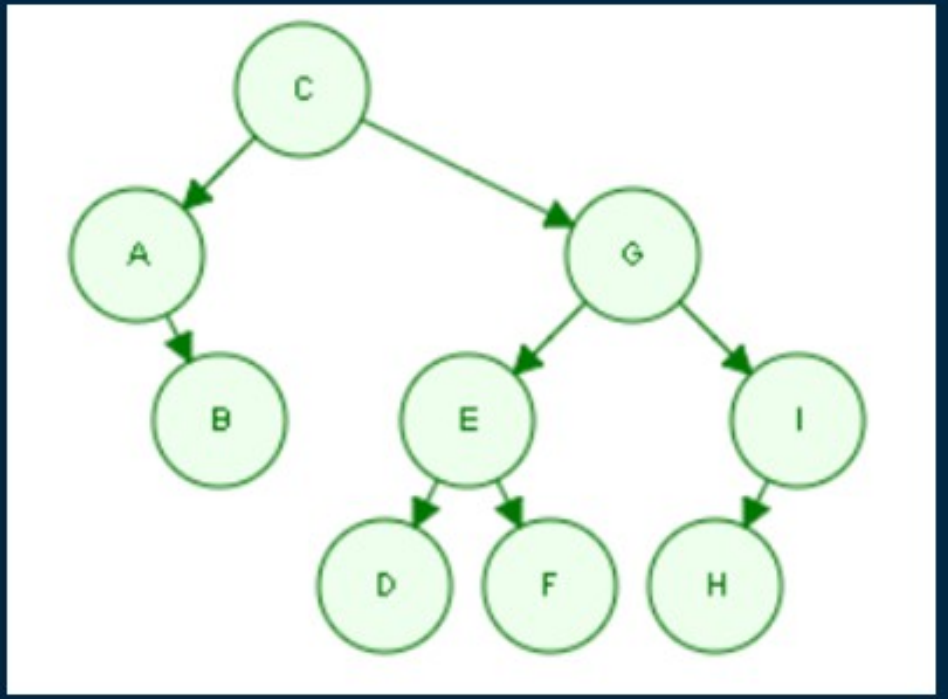
Operații Arbori Binar de Căutare

- Adăugare

Exercițiu:

1. adăugăm valoarea K
2. adăugăm valoarea E

Arbore Binar de Căutare

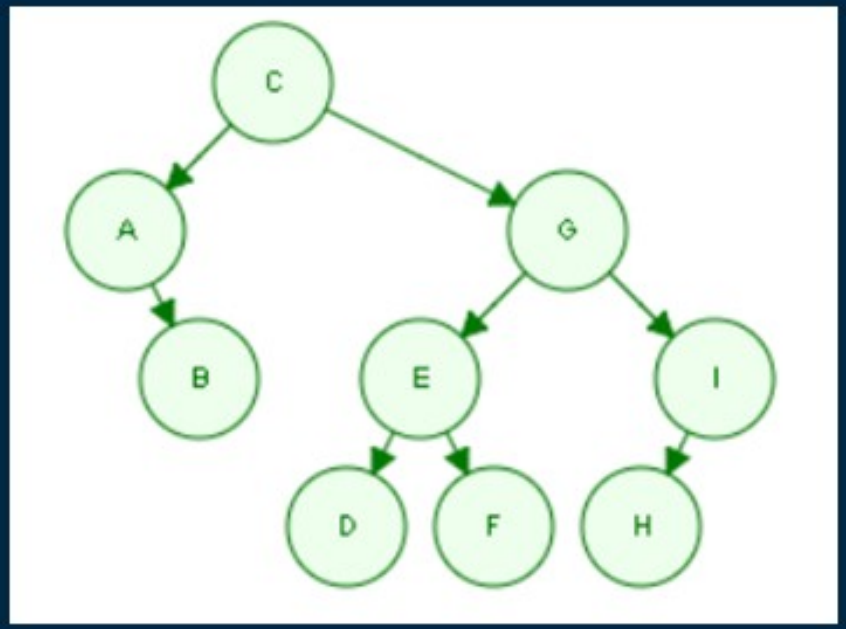


Operații Arbori Binar de Căutare

- Adăugare

Exercițiu:

1. adăugăm valoarea K



K > C

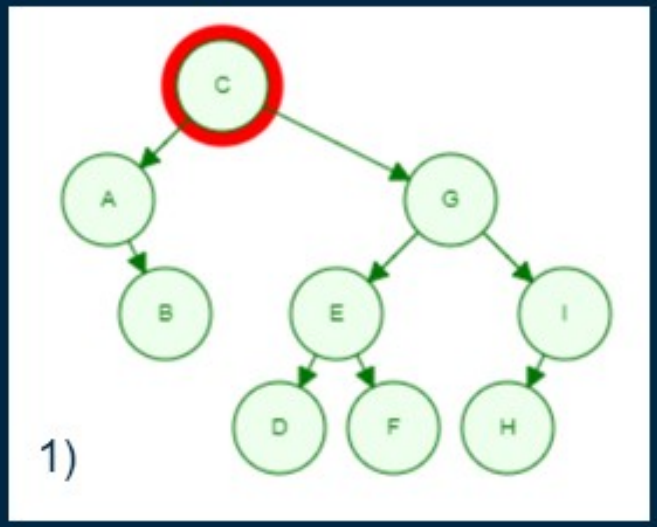
K > G

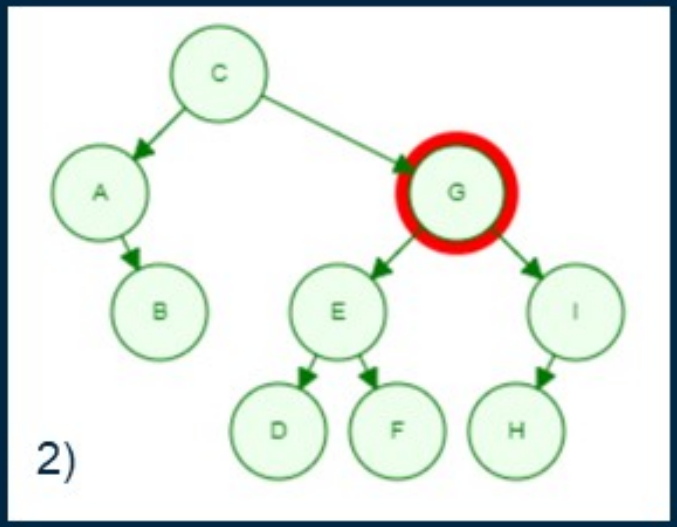
K > I

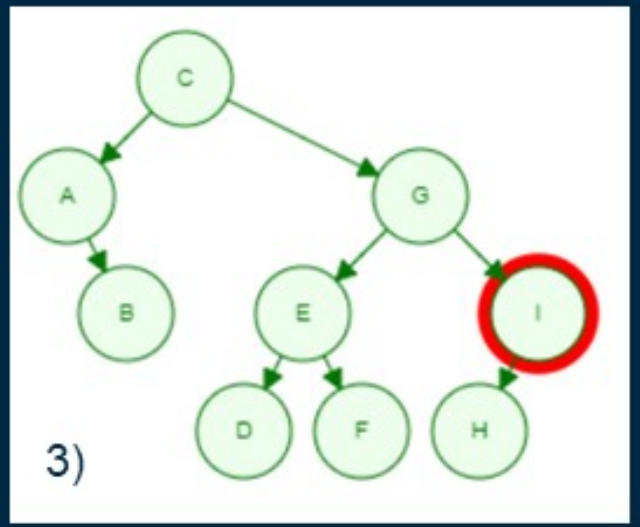
I – nod frunză

Operații Arbori Binar de Căutare

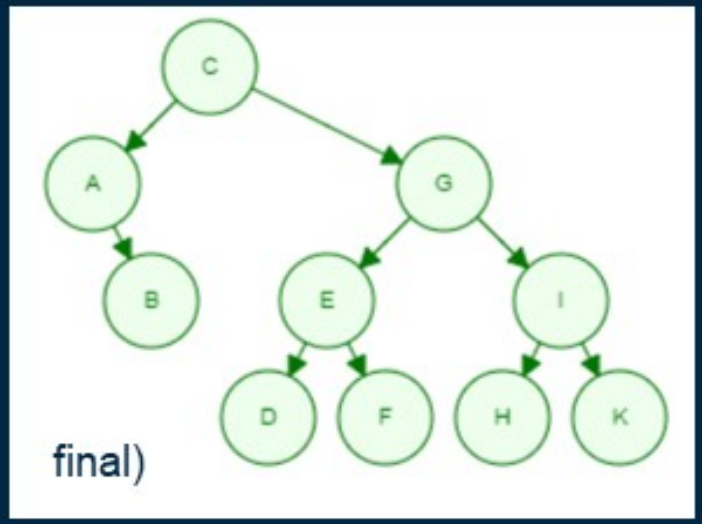
- Adăugare







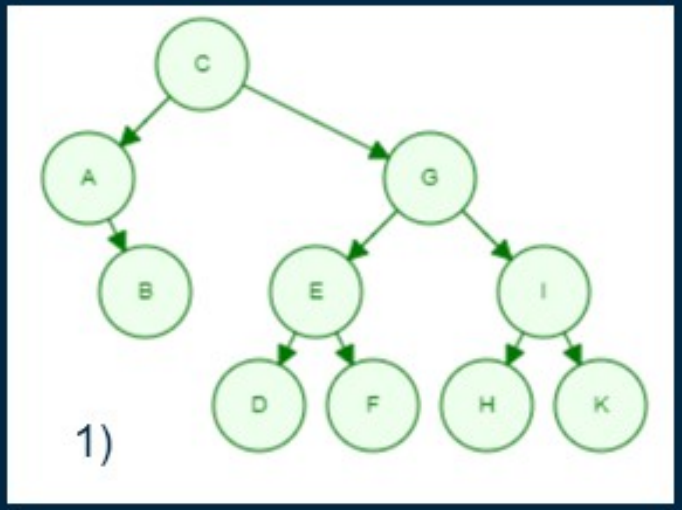
1. adăugăm valoarea K

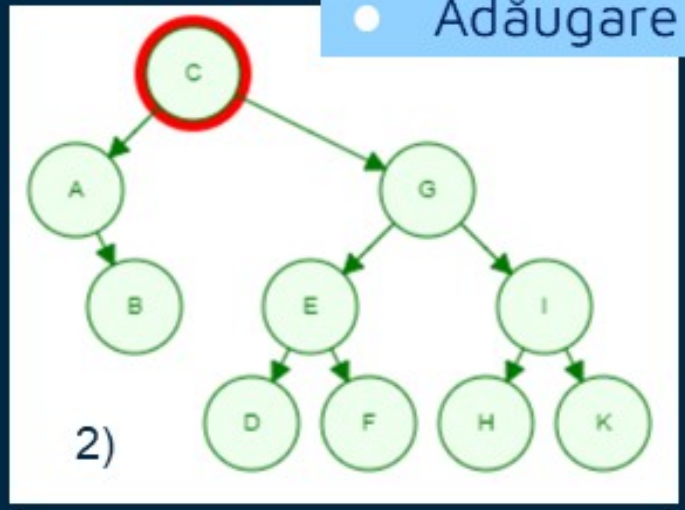


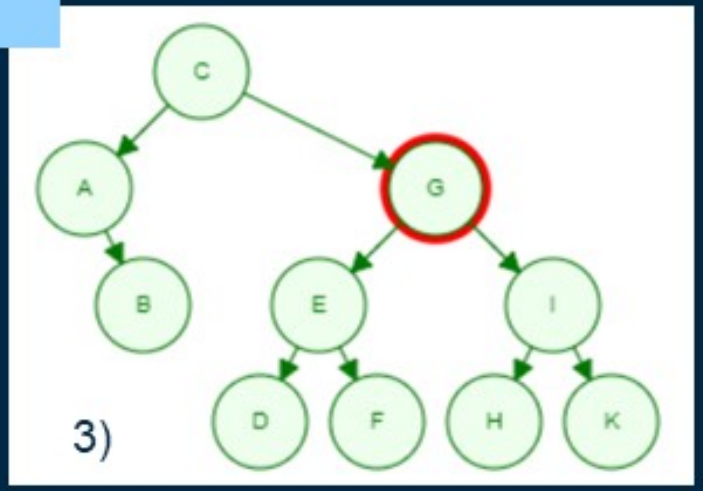
Operații Arbori Binar de Căutare

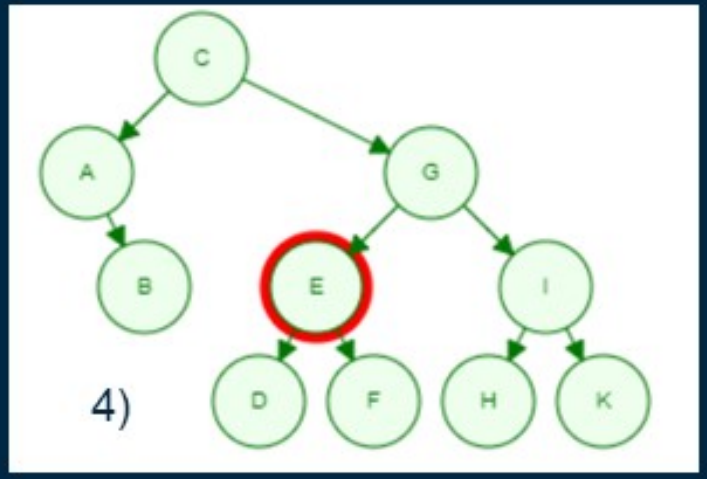
2. adăugăm valoarea E

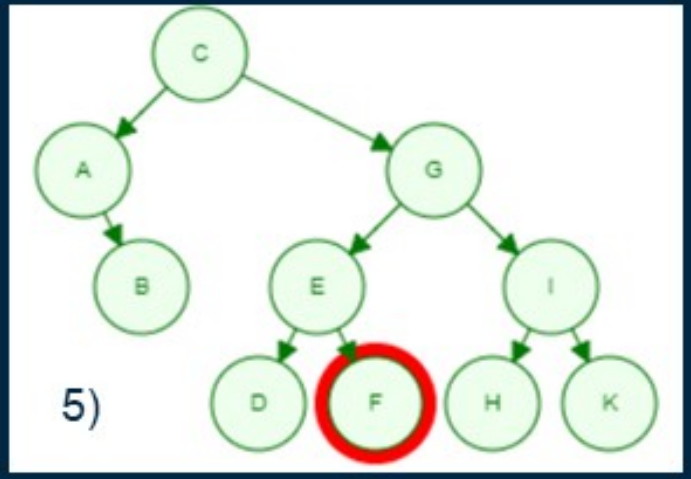
- Adăugare

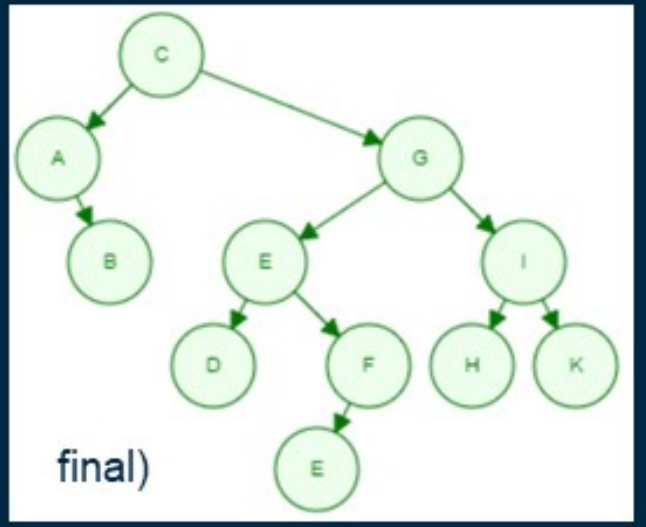












Operații Arbori Binar de Căutare

- Adăugare

```
begin procedure adaugNod
        if nod == NULL then
                return creareNod(data)
        endif
        if (data < nod->data) then
                nod->stanga = adaug(nod->left, data)
        else
                if (data > nod->data) then
                        node->dreapta = adaug(nod->dreapta, data)
                endif
        endif
        return nod
end procedure adaugNod
```

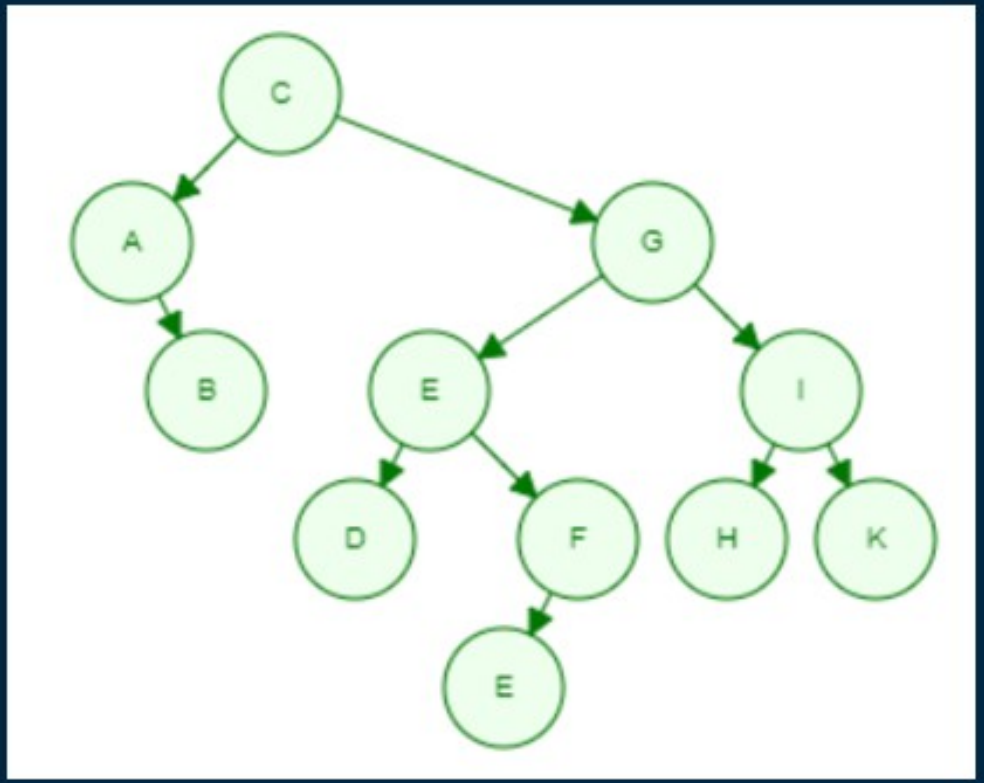
Operații Arbori Binar de Căutare

- Ștergere

Caz 1: Nodul de șters este nod frunză

Caz 2: Nodul de șters are un singur copil (stânga sau dreapta)

Caz 3: Nodul de șters are doi copii

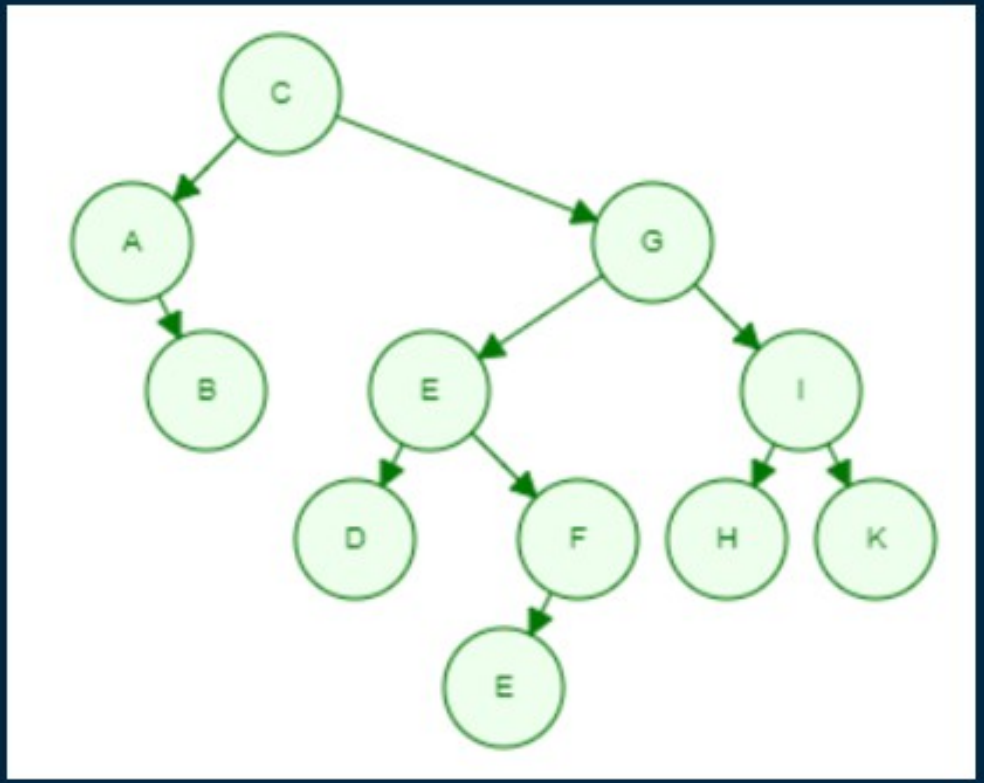


Operații Arbori Binar de Căutare

- Ștergere

Caz 1: Nodul de șters este nod frunză

Se șterge nodul frunză

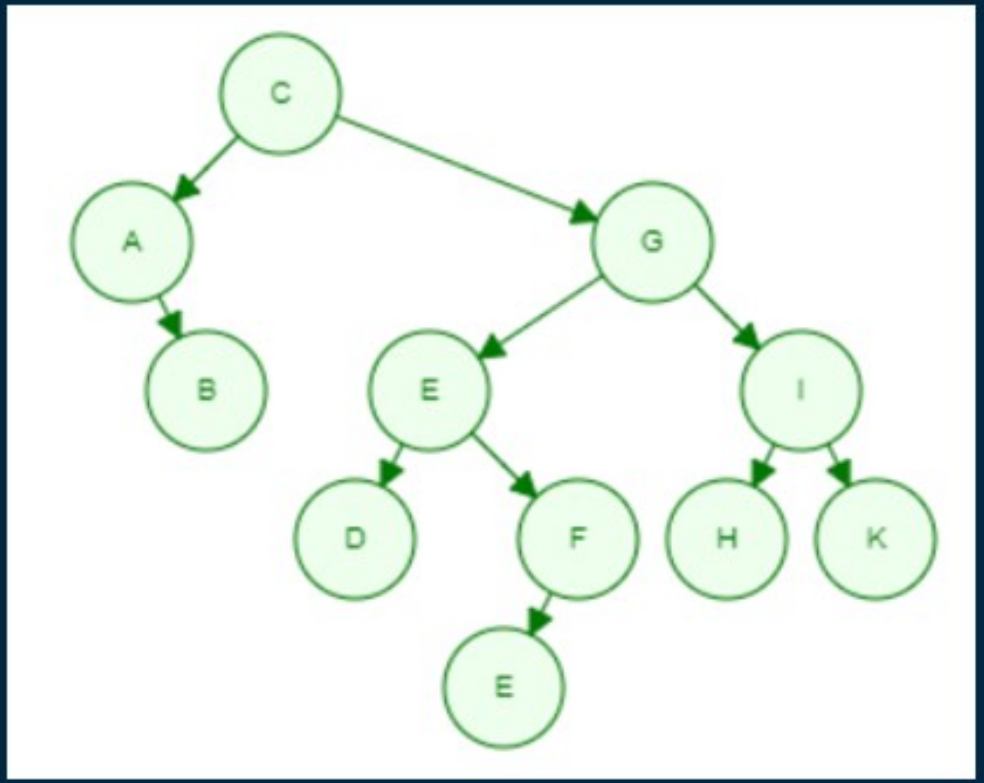


Operații Arbori Binar de Căutare

- Ștergere

Caz 1: Nodul de șters este nod frunză

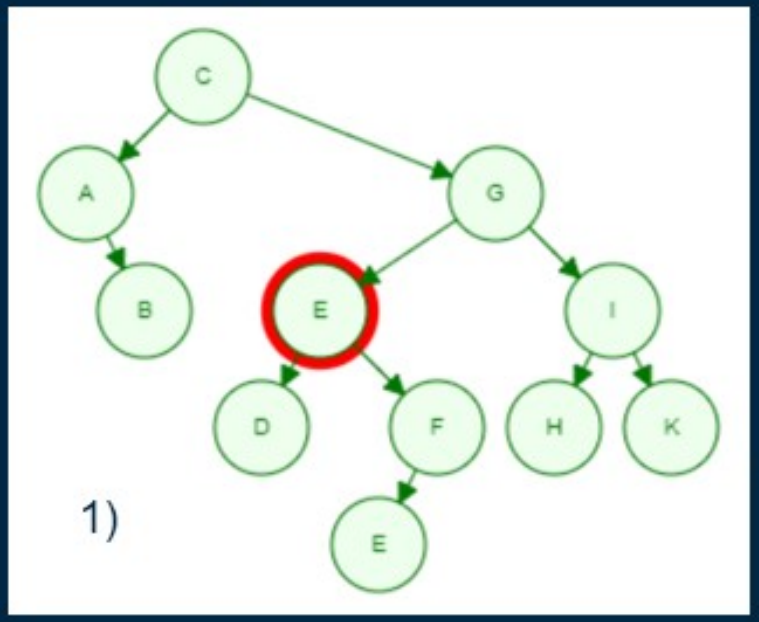
Se șterge nodul cu cheia D

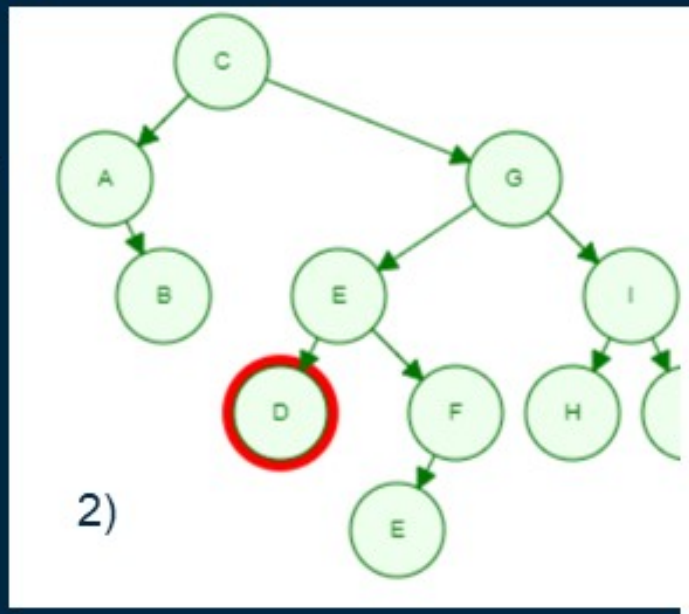


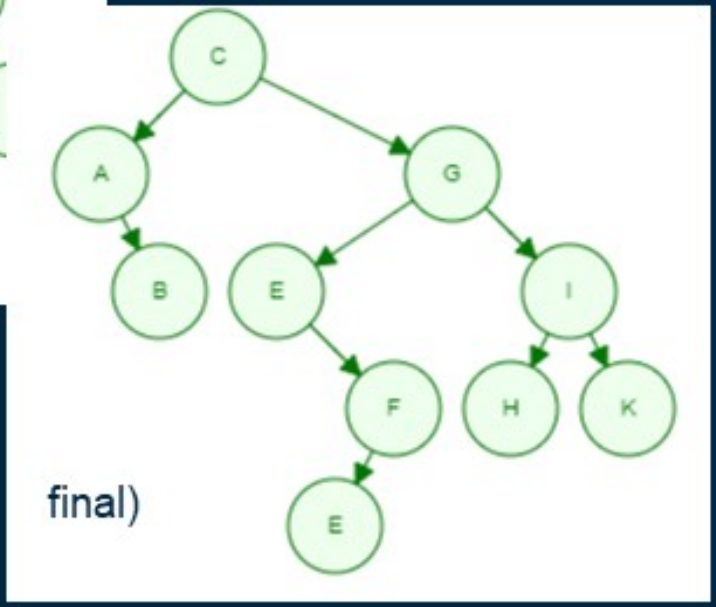
Operații Arbori Binar de Căutare

- Ștergere

Se șterge nodul cu cheia D





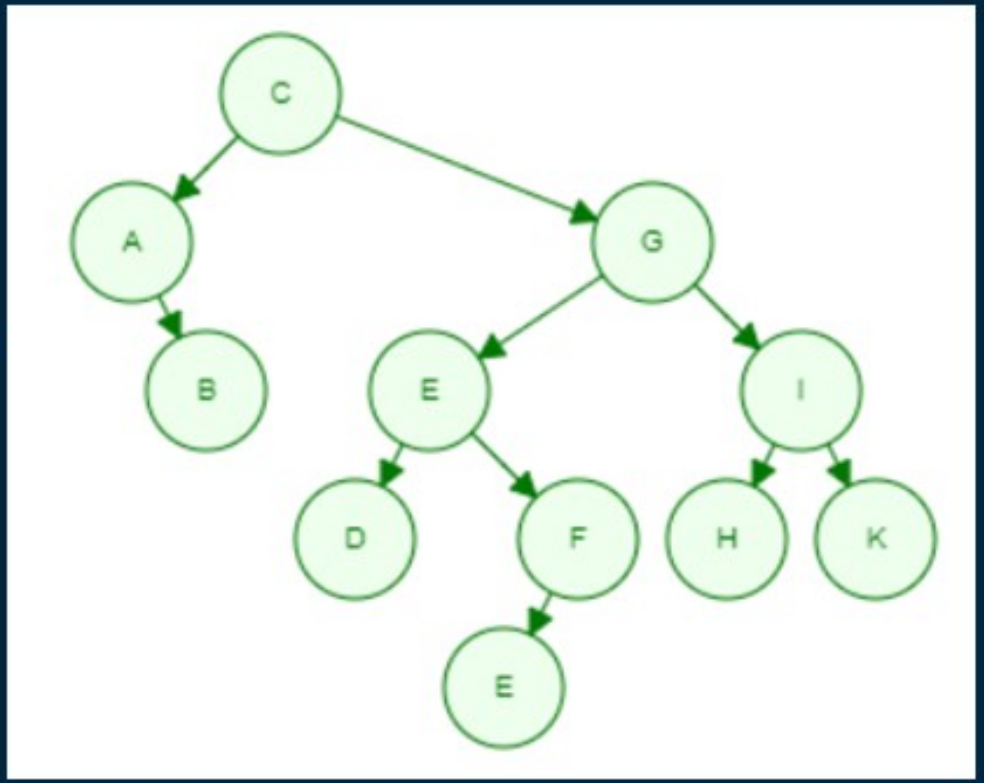


Operații Arbori Binar de Căutare

- Ștergere

Caz 2: Nodul de șters are un singur copil
(stânga sau dreapta)

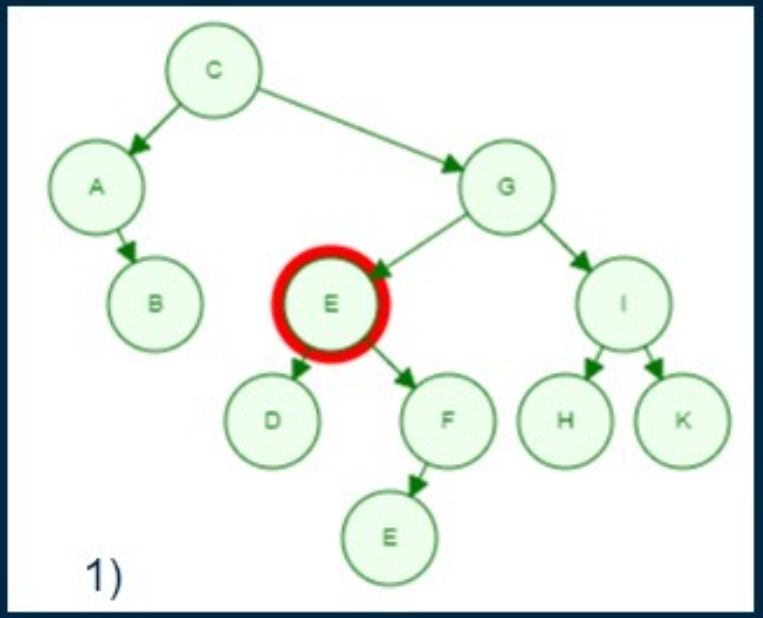
Pas 1: Se interschimbă nodul cu nodul său copil.
Pas 2: Se aplică algoritmul Caz 1.

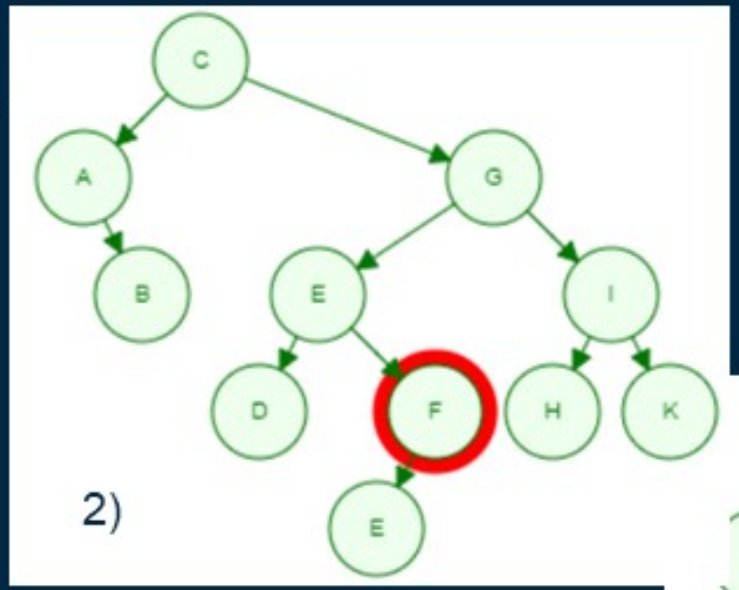


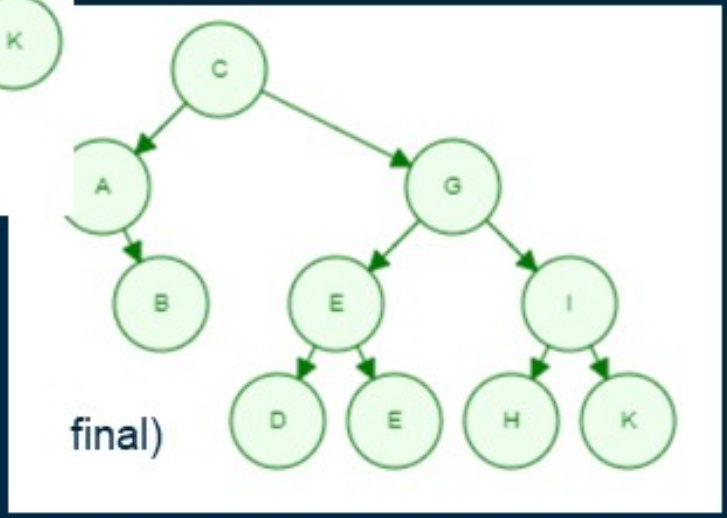
Operații Arbori Binar de Căutare

- Ștergere

Se șterge nodul cu cheia F





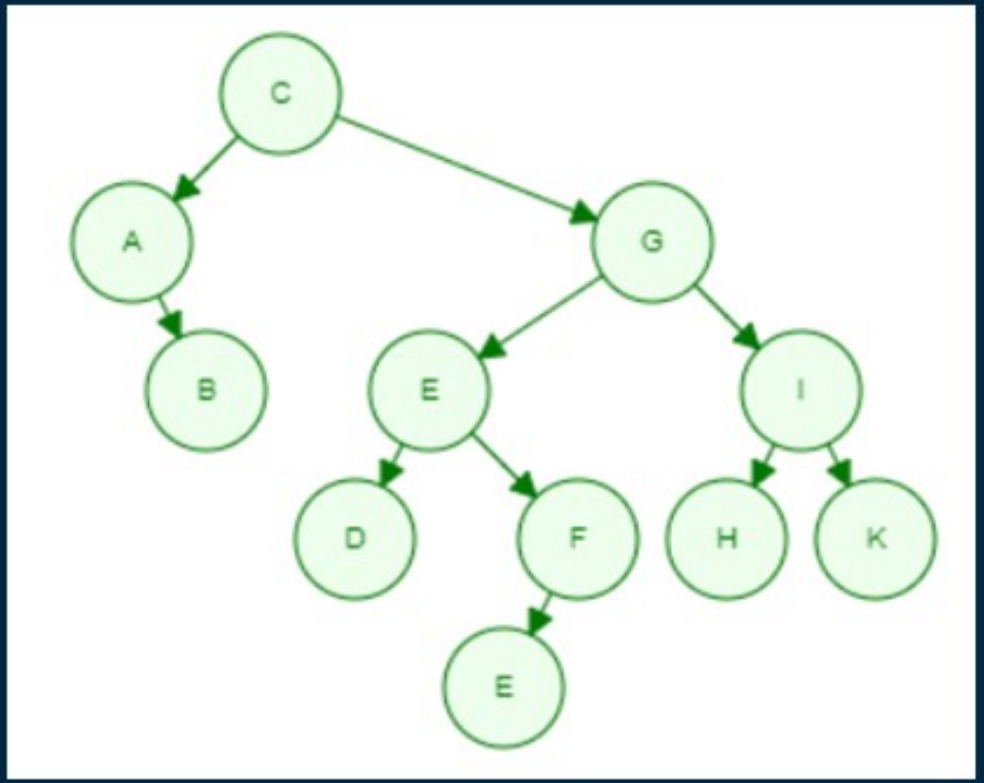


Operații Arbori Binar de Căutare

- Ștergere

Caz 3: Nodul de șters are doi copii

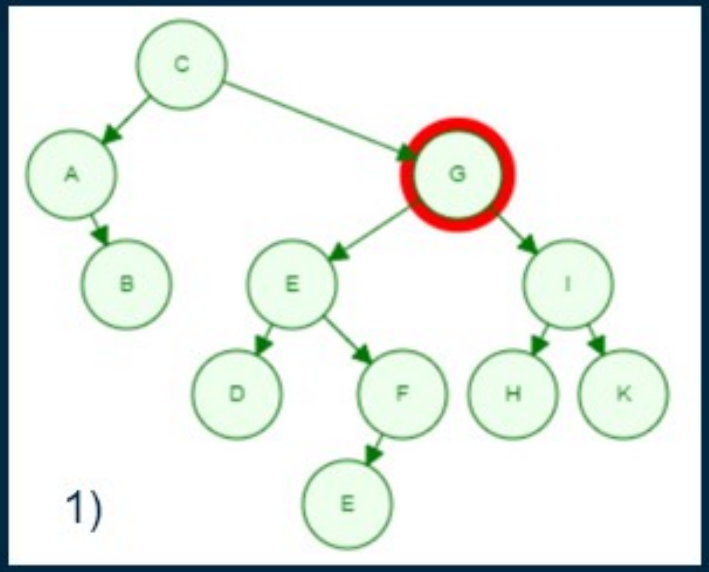
Pas 1: Se determină succesorul InOrdine
Pas 2: Se interschimbă nodul determinat la Pas 1. cu nodul de șters
Pas 3: Se șterge nodul

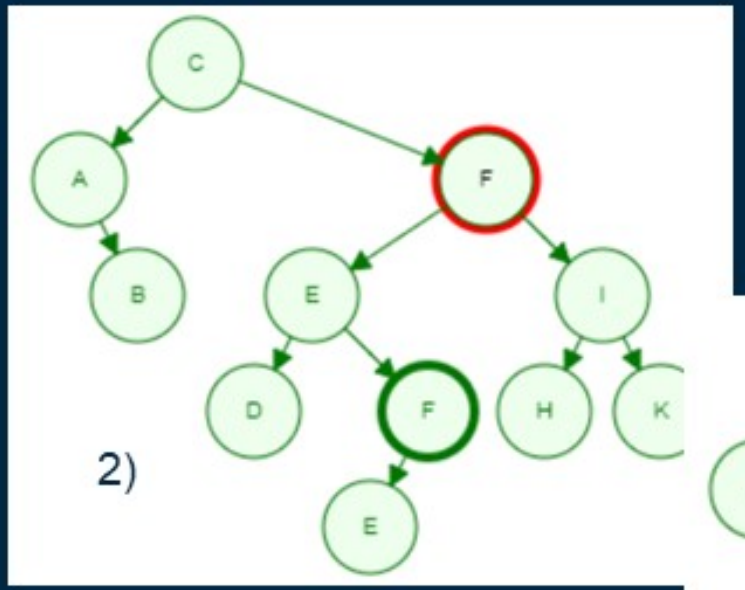


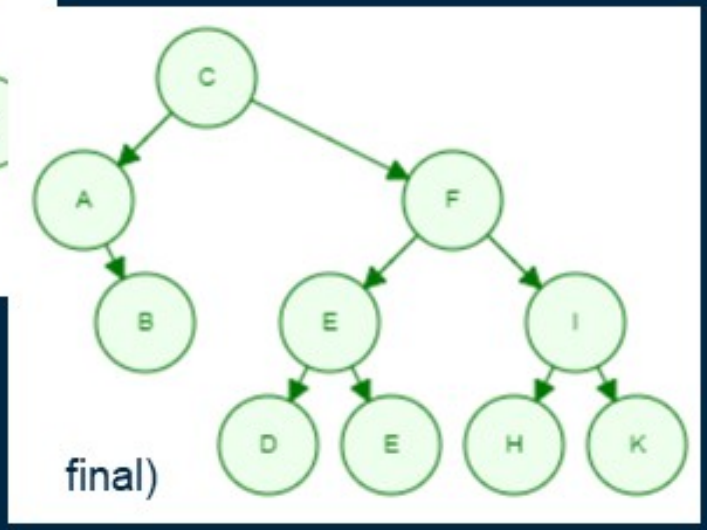
Operații Arbori Binar de Căutare

- Ștergere

Se șterge nodul cu cheia G





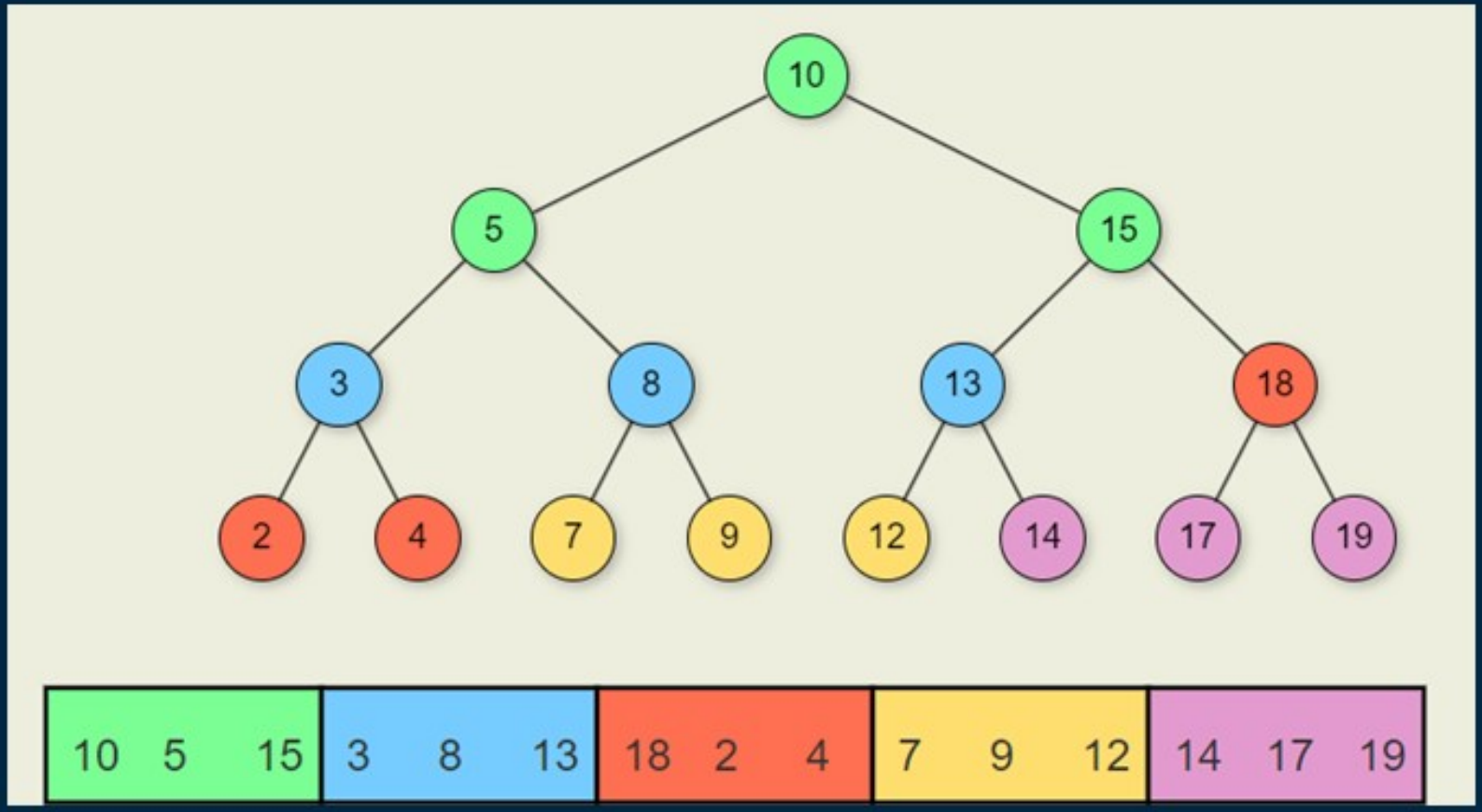


Exemple

Index

ABC asociat

Cautare – 9
Acces disk - ?

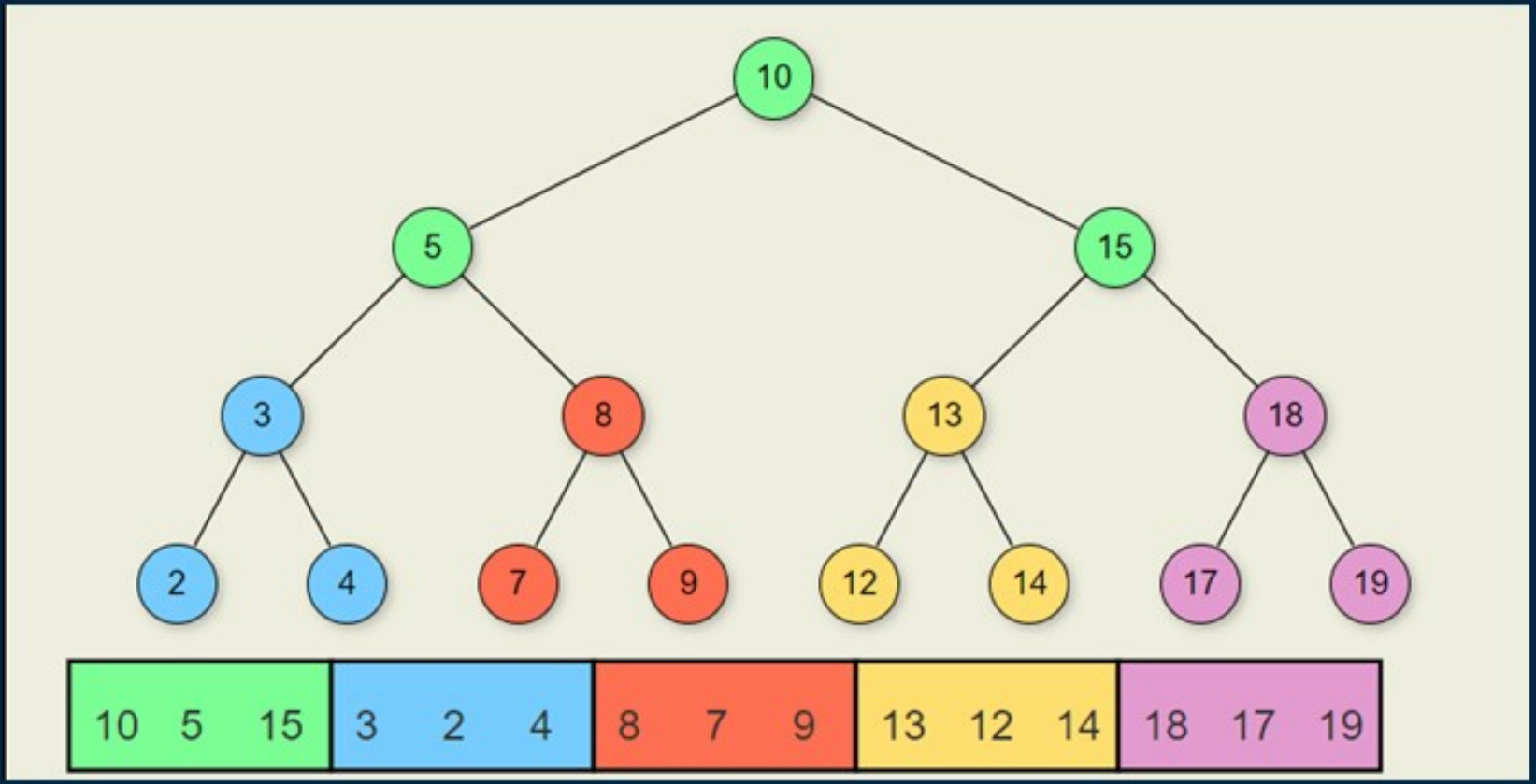


Exemple

Index

ABC asociat

Cautare – 9
Acces disk - ?



Arbori Binar de Căutare – complexitate (timp)

Operația	Caz favorabil	Caz mediu	Caz nefavorabil
Căutare	O(log n)	O(log n)	O(n)
Adăugare	O(log n)	O(log n)	O(n)
Ștergere	O(log n)	O(log n)	O(n)

Aplicații ale Arborilor Binari de Căutare

- indexarea pe mai multe niveluri în baza de date
- sortare dinamică
- gestionarea zonelor de memorie virtuală în nucleul Unix

Intrebari?

dorin.lordache@365.univ-ovidius.ro

Mulțumesc