Cursul nr. 7

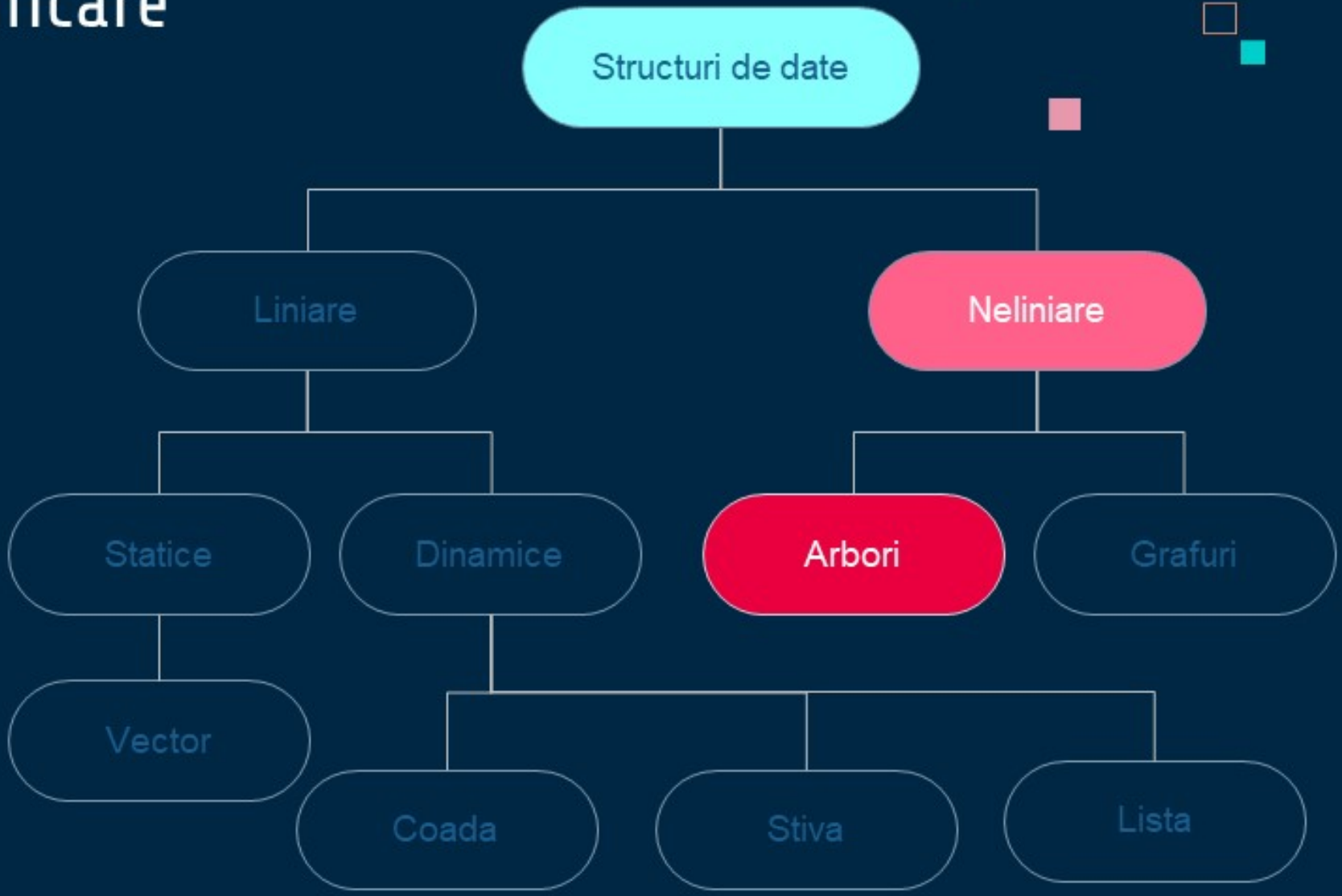
STRUCTURI DE DATE

neliniare

Arbori

Lector dr. Dorin IORDACHE

Clasificare



Agenda

01

Arbori

02

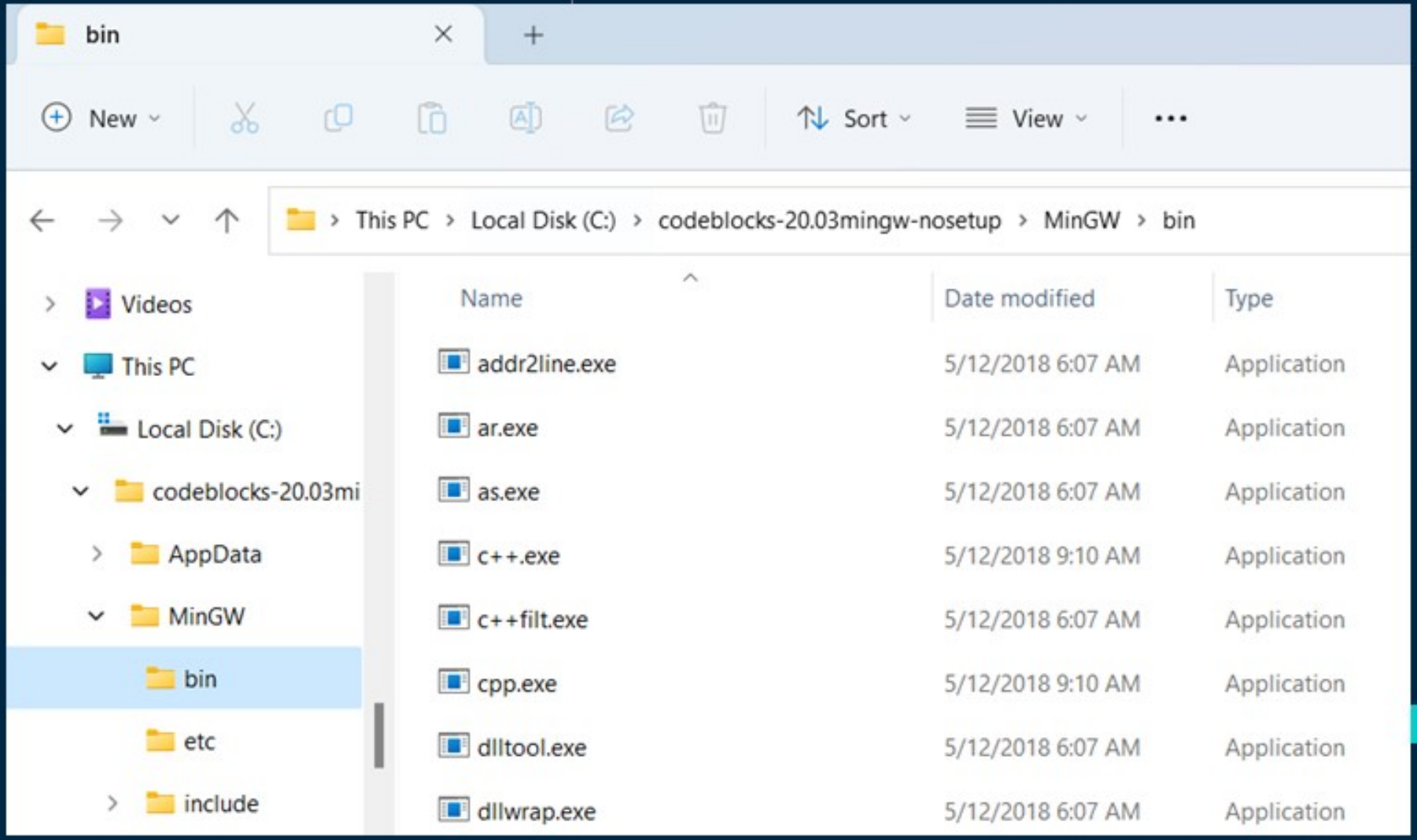
Parcurgeri

03

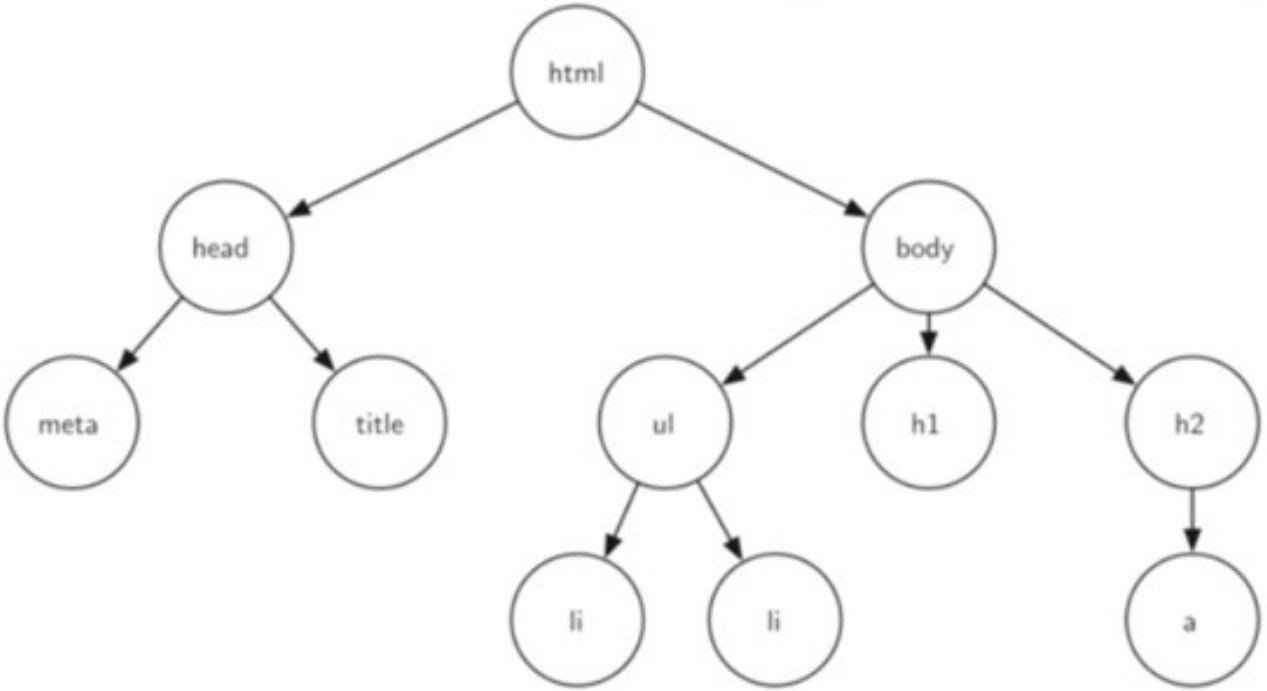
Arbori Binar

Arbori

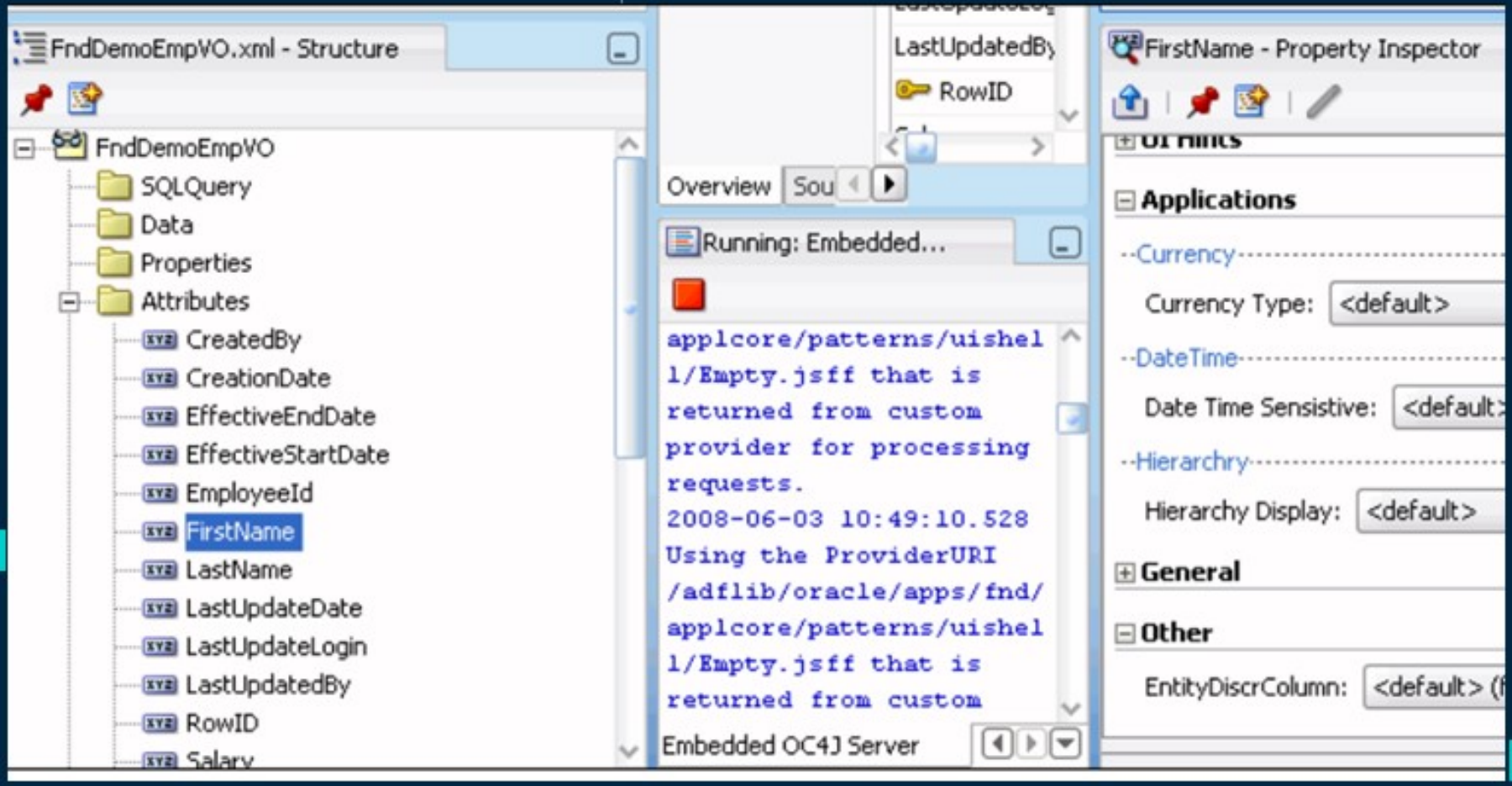
01


bin
New
Sort
View
This PC
Local Disk (C:)
codeblocks-20.03mingw-nosetup
MinGW
bin
Videos
This PC
Local Disk (C:)
codeblocks-20.03mi
AppData
MinGW
bin
etc
include
Name
Date modified
Type
addr2line.exe 5/12/2018 6:07 AM Application
ar.exe 5/12/2018 6:07 AM Application
as.exe 5/12/2018 6:07 AM Application
c++.exe 5/12/2018 9:10 AM Application
c++filt.exe 5/12/2018 6:07 AM Application
cpp.exe 5/12/2018 9:10 AM Application
dlltool.exe 5/12/2018 6:07 AM Application
dllwrap.exe 5/12/2018 6:07 AM Application

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 <title>HTML simplu</title>
 </head>
 <body>
<h1>Un exemplu structura arborescenta</h1>
 <ul>
<li>Item lista 1</li>
<li>Item lista 2</li>
</ul>
<h2><a href="http://fmi.univ-ovidi
</body>
</html>
```



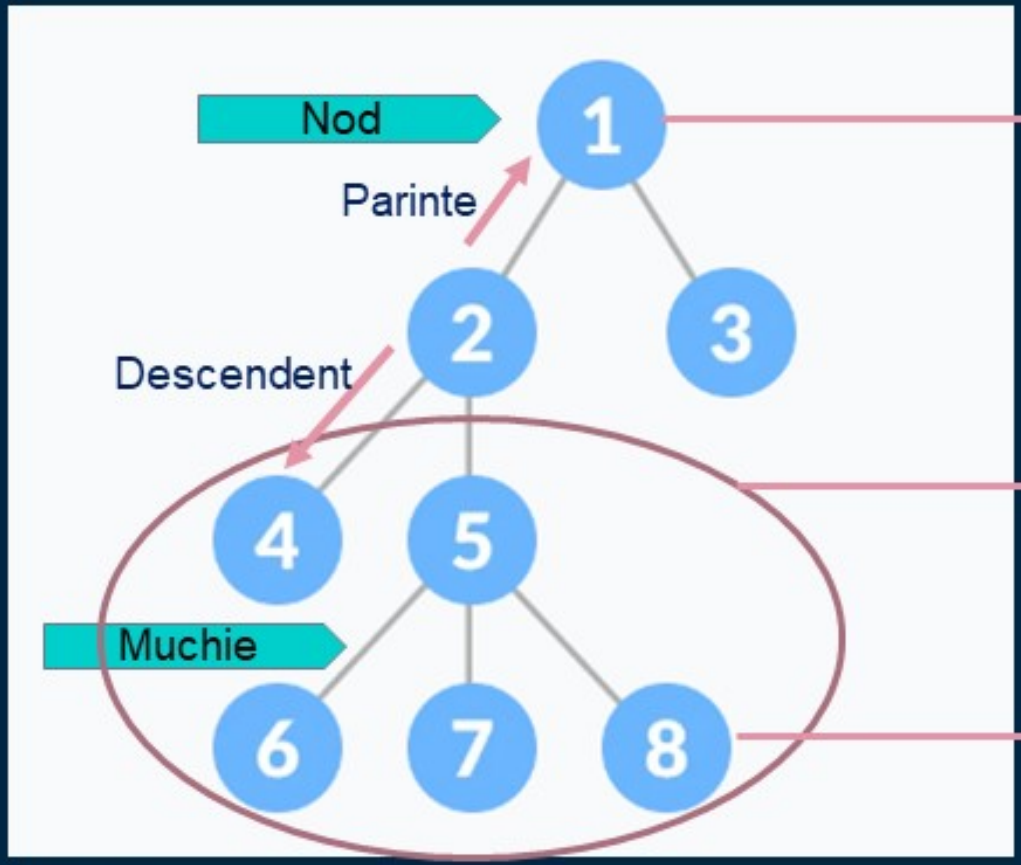
```
/etc
├── acpi
│   ├── asus-keyboard-backlight.sh
│   ├── asus-wireless.sh
│   ├── events
│   │   ├── asus-keyboard-backlight-down
│   │   ├── asus-keyboard-backlight-up
│   │   ├── asus-wireless-off
│   │   ├── asus-wireless-on
│   │   ├── ibm-wireless
│   │   ├── lenovo-undock
│   │   ├── thinkpad-cmos
│   │   └── tosh-wireless
│   ├── ibm-wireless.sh
│   ├── tosh-wireless.sh
│   └── undock.sh
├── adduser.conf
├── aliases
├── alsa
│   └── conf.d
```



https://docs.oracle.com/cd/E28271_01/fusionapps.1111/e15524/ui_tree_mgmt.htm#CIHIGDDG

Arbori

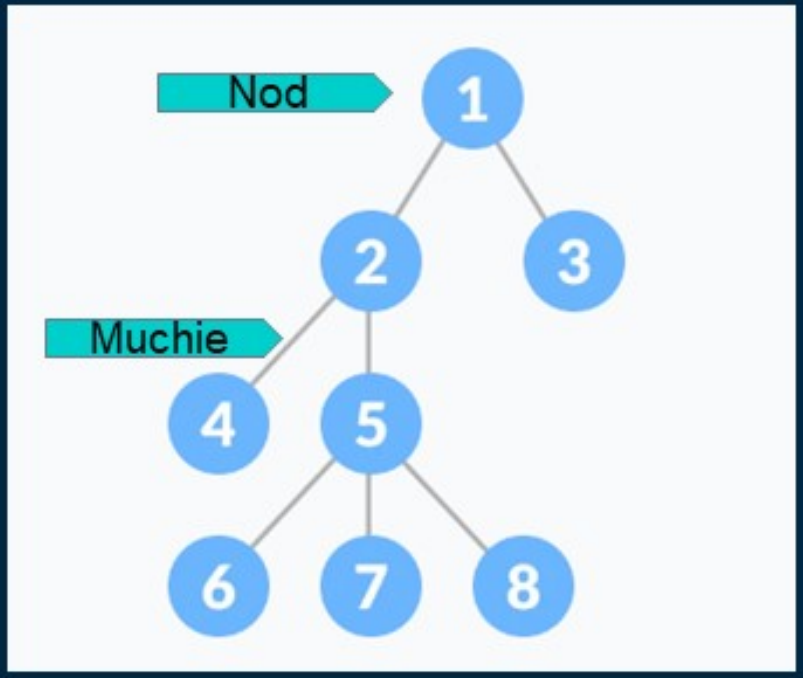
= structură de date ierarhică neliniară care constă din noduri conectate prin muchii.



https://www.programiz.com/dsa/trees

Arbori

= structură de date ierarhică neliniară care constă din noduri conectate prin muchii.



https://www.programiz.com/dsa/trees

Arbori

= structură de date ierarhică neliniară care constă din noduri conectate prin muchii.

Nod = entitate care conține o cheie sau o valoare și legaturi către nodurile sale subordonate.

Ultimele noduri ale fiecărei căi sunt numite **noduri frunză** sau **noduri externe.**

Nodul care are cel puțin un nod copil se numește **nod intern**.

Muchie Legatura intre 2 noduri

Arbori – notiuni descriptive

Rădăcină (root)
Este nodul cel mai de sus al arborelui. Este unic.

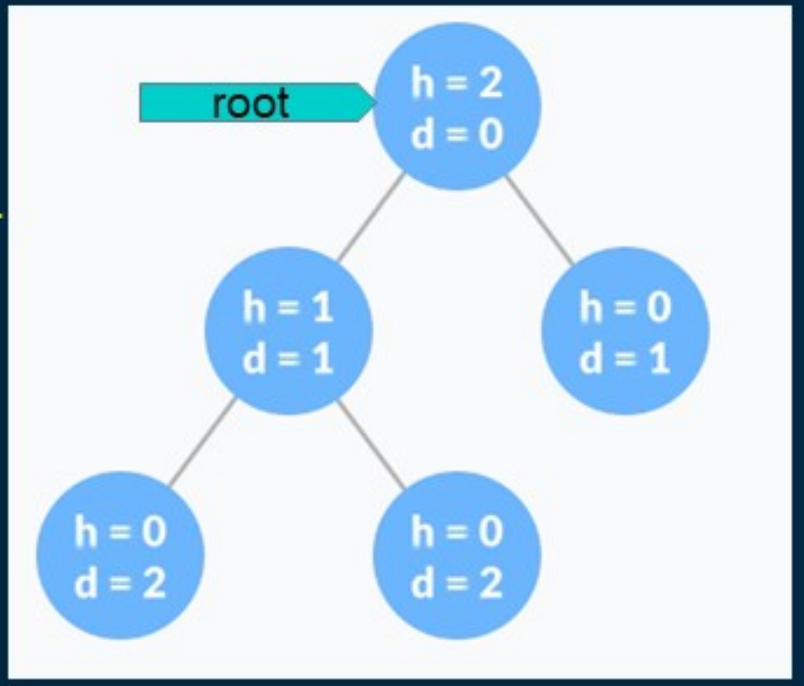
Înălțimea unui nod
Înălțimea unui nod este numărul de muchii de la nod la cea mai adâncă frunză (adică cea mai lungă cale de la nod la un nod de frunză).

Adâncimea unui nod
Adâncimea unui nod este numărul de muchii de la rădăcină la nod.

Înălțimea arborelui
Înălțimea unui arbore este înălțimea nodului rădăcină sau adâncimea celui mai adânc nod.

Gradul unui nod
Gradul unui nod este numărul total de ramuri ale acelui nod.

Padure (Forest)
O colecție de arbori disjunși se numește pădure.

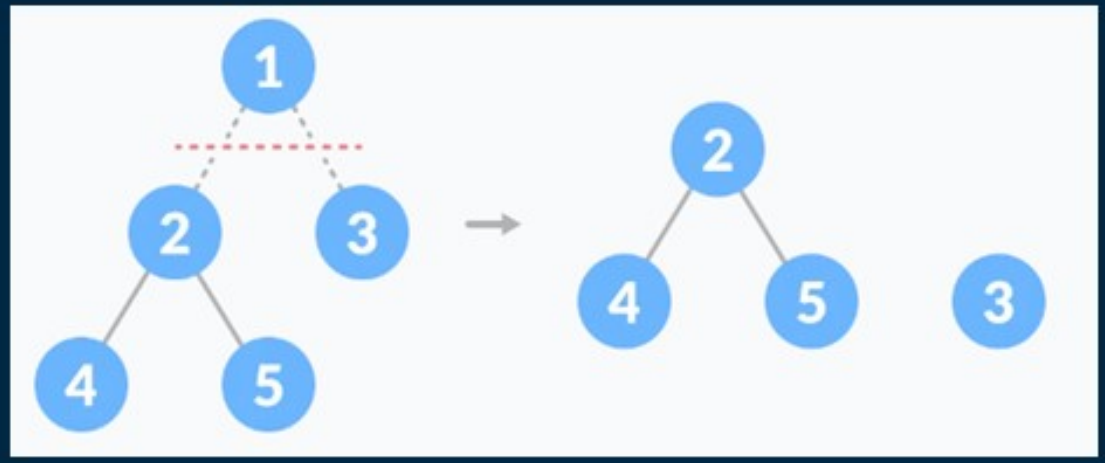


Inaltimea (h) Adancimea (d)

https://www.programiz.com/dsa/trees

Arbori – notiuni descriptive

Padure (Forest)
O colecție de arbori disjunși se numește pădure.



https://www.programiz.com/dsa/trees

Parcurgeri

02

Parcurgerea arborilor

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

Algoritmul de traversare a arborelui – solutia

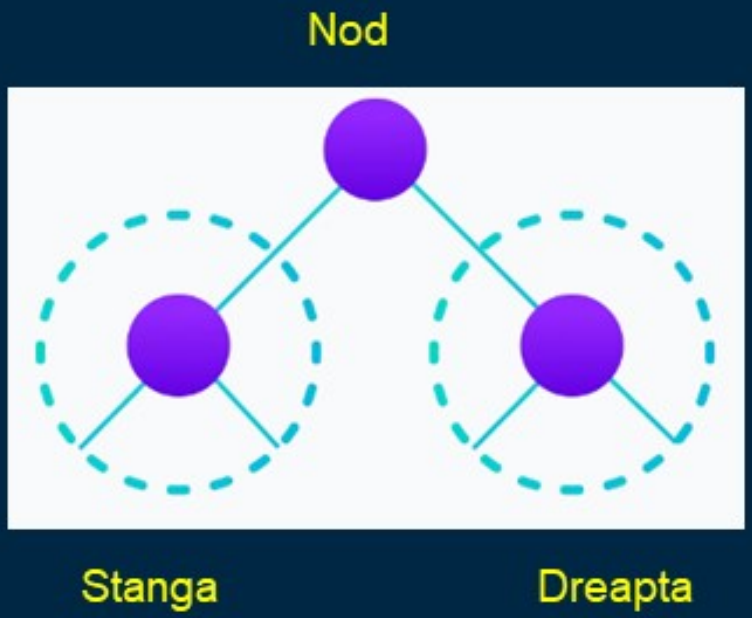
Tipuri de parcurgeri:

- 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

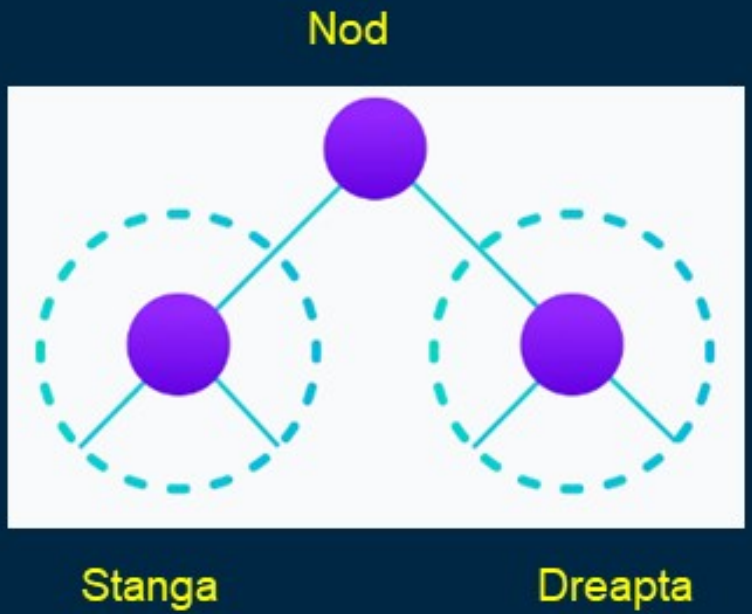
```
InOrdine(root->stanga)
afisare(root->data)
InOrdine(root->dreapta)
```



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

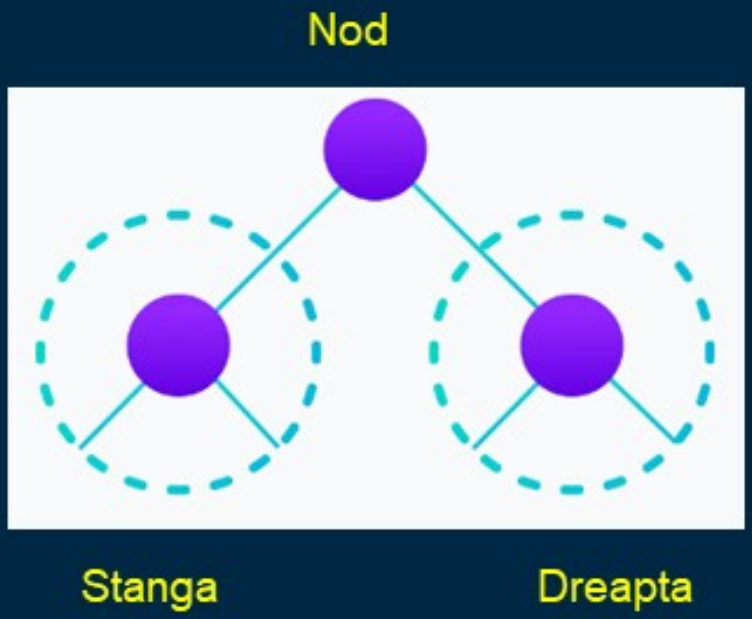
```
afisare(root->data)
PreOrdine(root->stanga)
PreOrdine(root->dreapta)
```



PostOrdine (SDN, LRN)

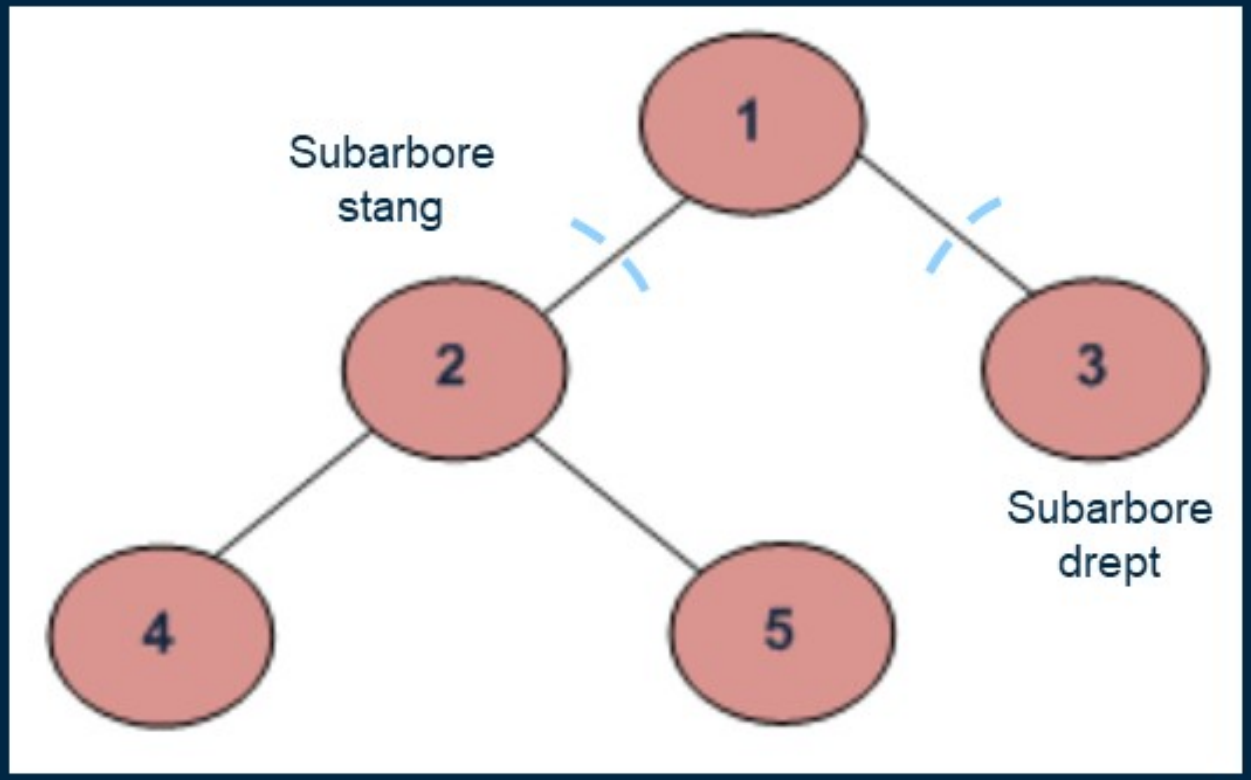
Pas 1: Se parcurg toate nodurile din subarborele din stânga
Pas 2: Se parcurg toate nodurile din subarborele din dreapta
Pas 3: Se viziteaza nodul radacina

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

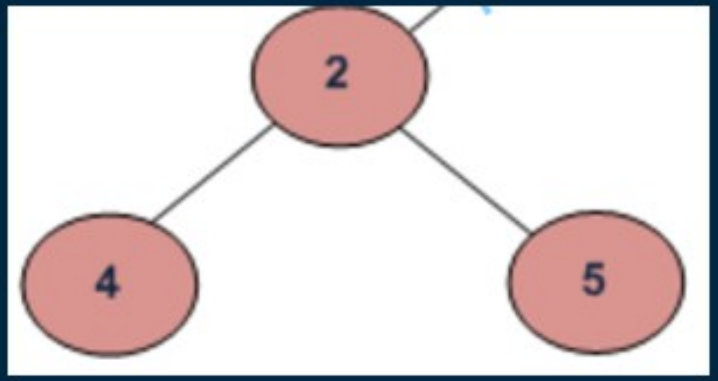


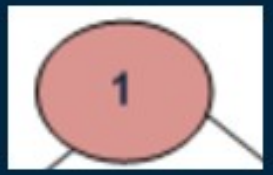
Parcurgeri - exemplu

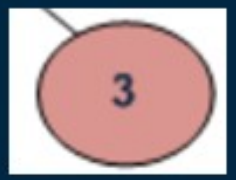
InOrdine (SND)



Stiva



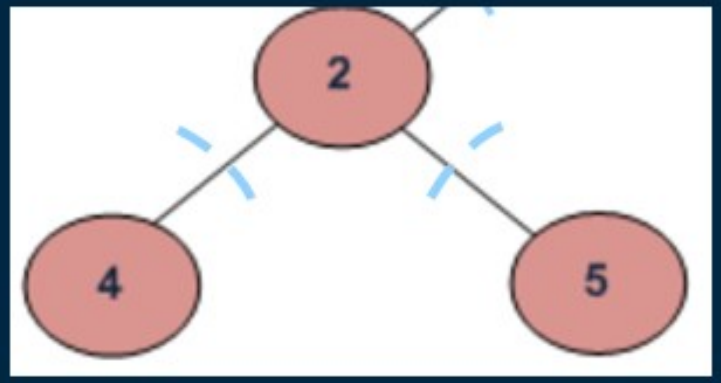




Parcurgeri - exemplu

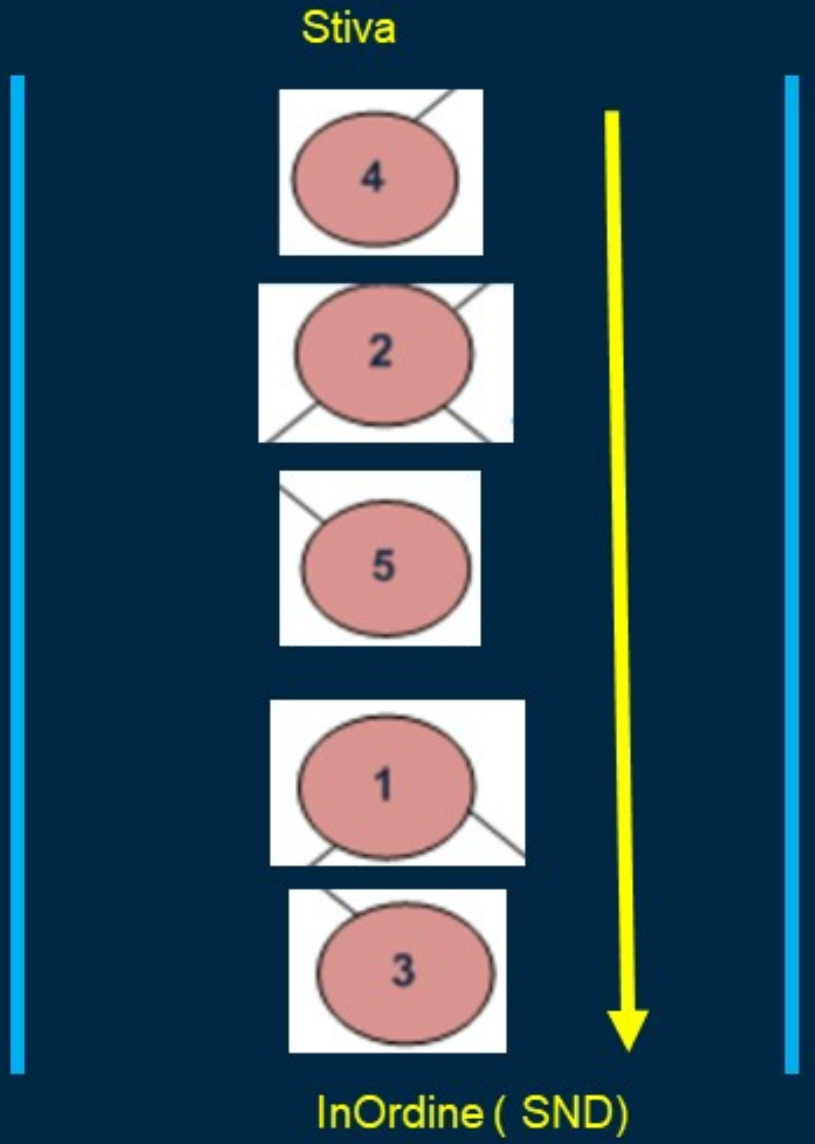
InOrdine (SND):

Root -> subarbore Stang -> subarbore Drept



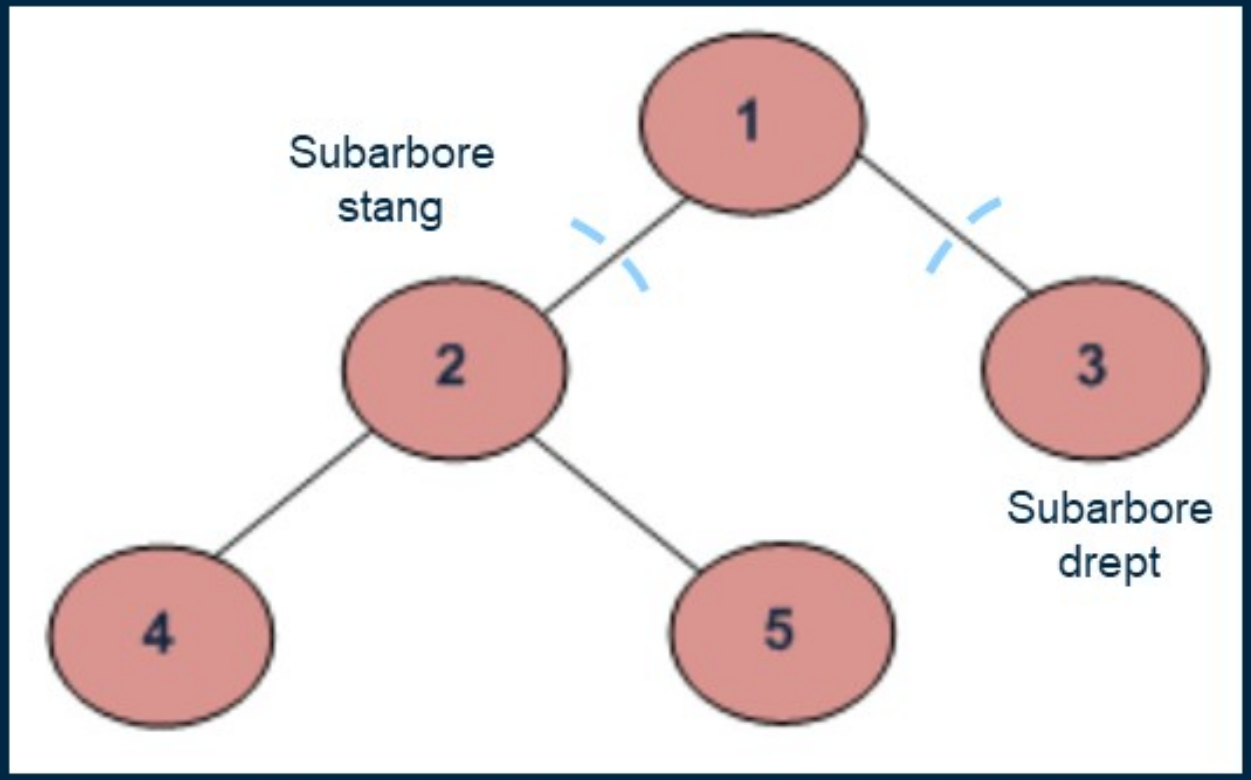
Rezultat parcurgere InOrdine:

4 -> 2 -> 5 -> 1 -> 3

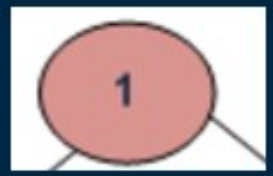


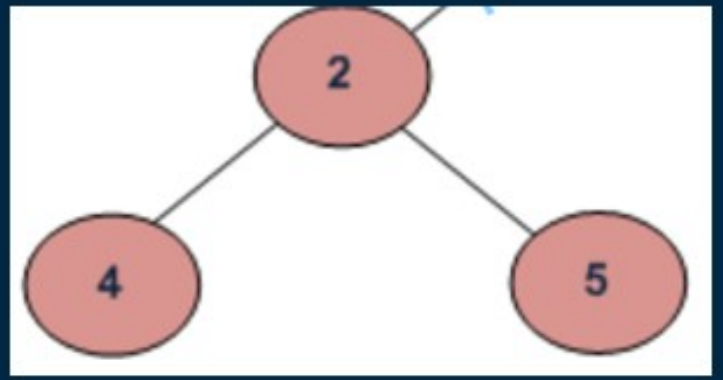
Parcurgeri - exemplu

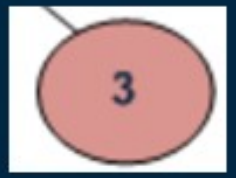
PreOrdine (NSD)



Stiva



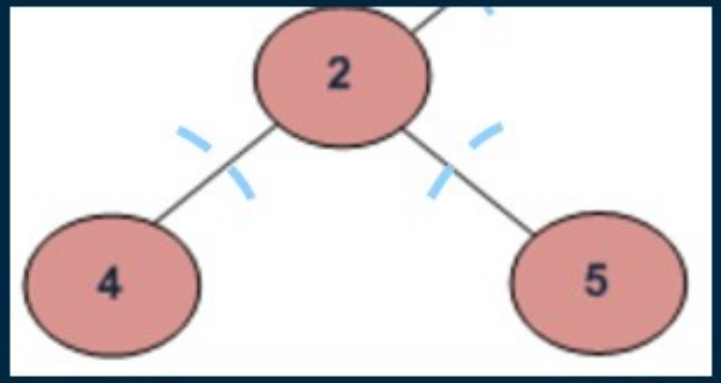




Parcurgeri - exemplu

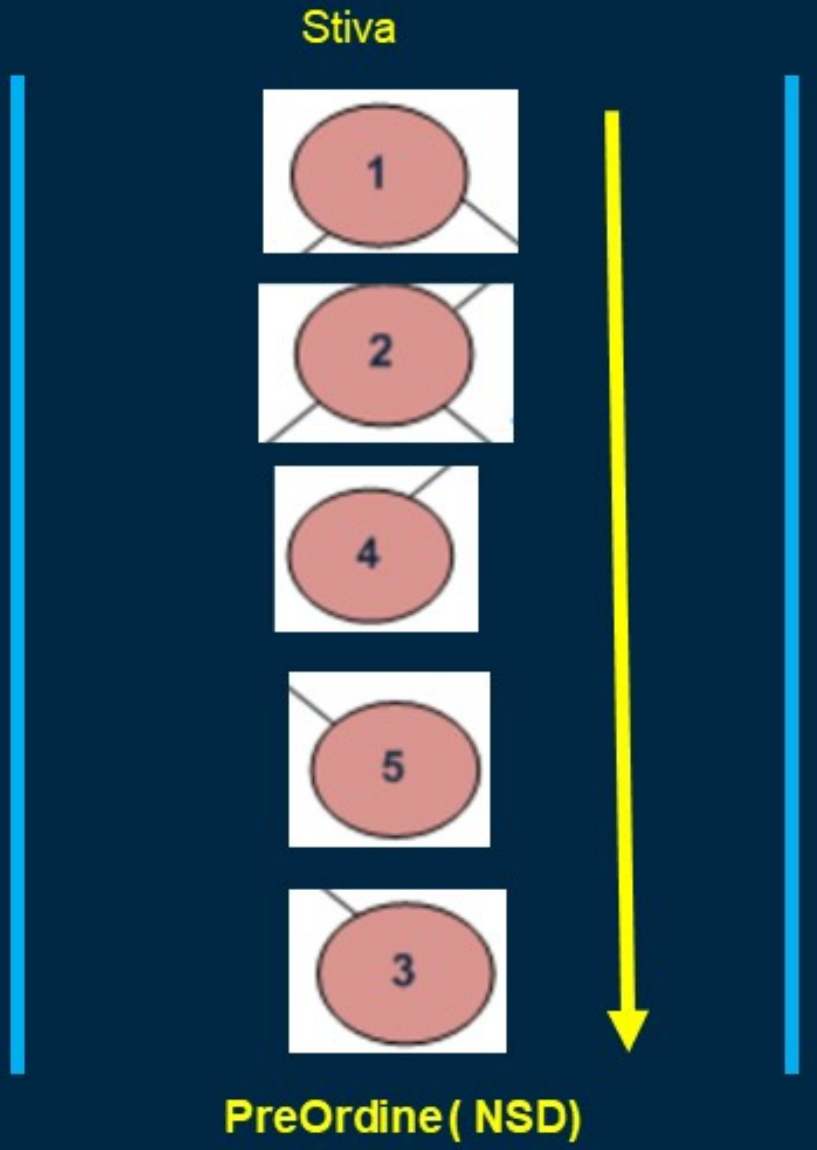
PreOrdine (NSD):

Root -> subarbore Stang -> subarbore Drept



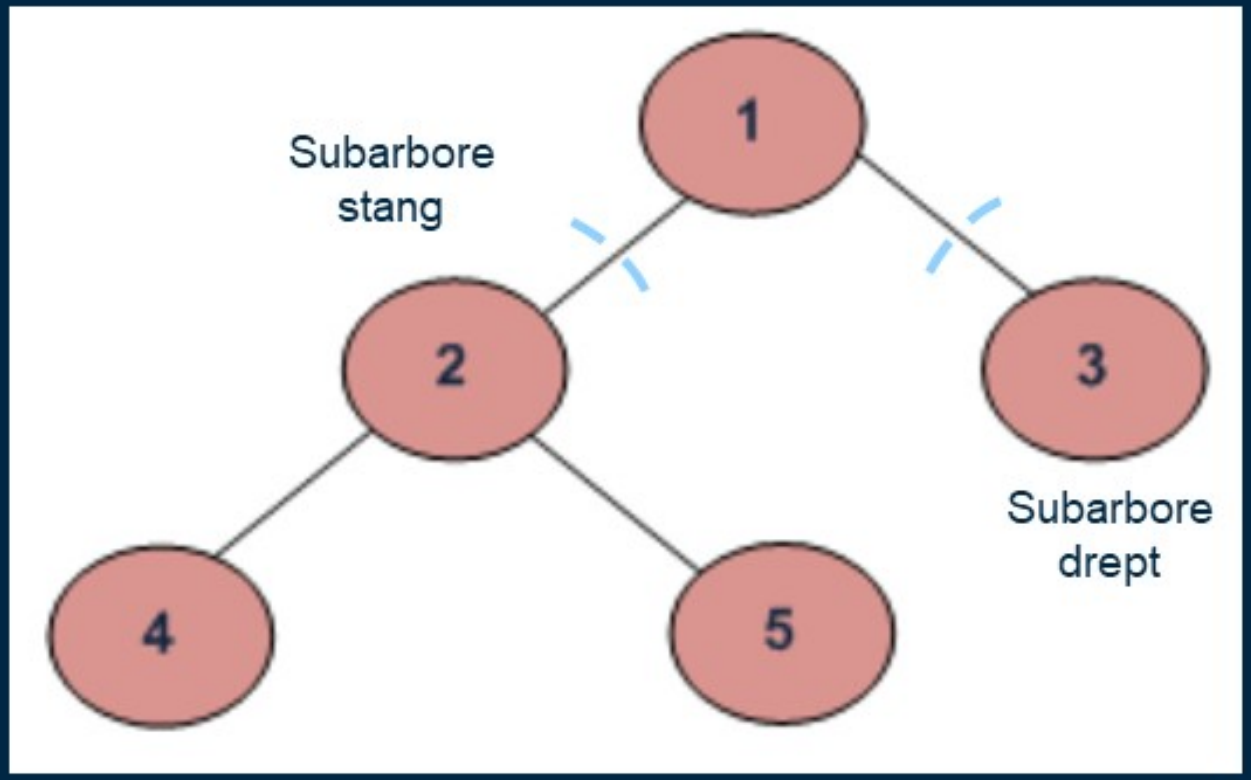
Rezultat parcurgere PreOrdine:

1 -> 2 -> 4 -> 5 -> 3

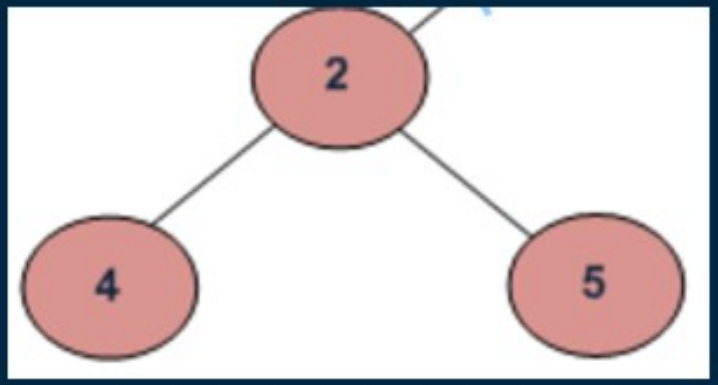


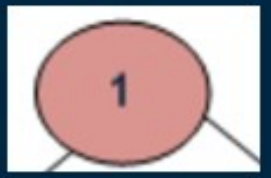
Parcurgeri - exemplu

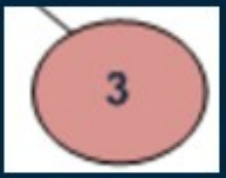
PostOrdine (SDN)



Stiva



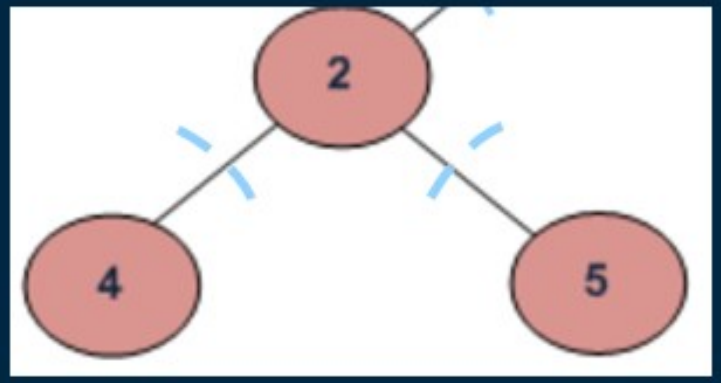




Parcurgeri - exemplu

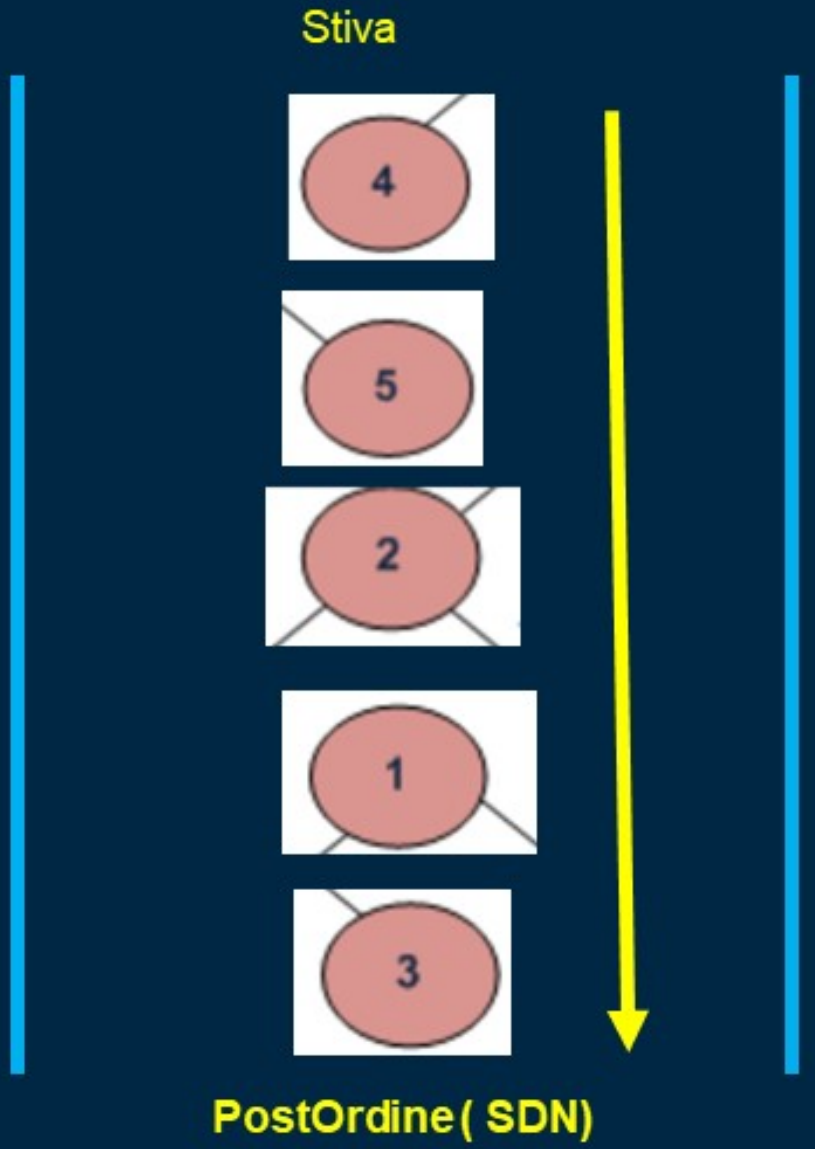
PostOrdine (SDN):

subarbore Stang -> subarbore Drept -> root



Rezultat parcurgere PostOrdine:

4 -> 5 -> 2 -> 3 -> 1



Aplicații ale arborilor

- Arborele de căutare binar (BST) – operații de căutare.
- Heap - folosit pentru sortarea heap.
- A stoca informații de rutare in echipamente de retea.
- SGBD - B-Trees și T-Trees, pentru a stoca datele.
- Compilatoare - arbore de sintaxă pentru a valida sintaxa fiecărui program.

Arbori Binar

03

Arbori binari (AB)

- Structura de date simpla și eficientă structură utilizată în majoritatea sistemelor software.
- AB poate fi implementat ca o matrice folosind idei de Binary Heap.
- Există o gamă largă de variante ale arborelui binar..

Arbori binari (AB)

Variante:

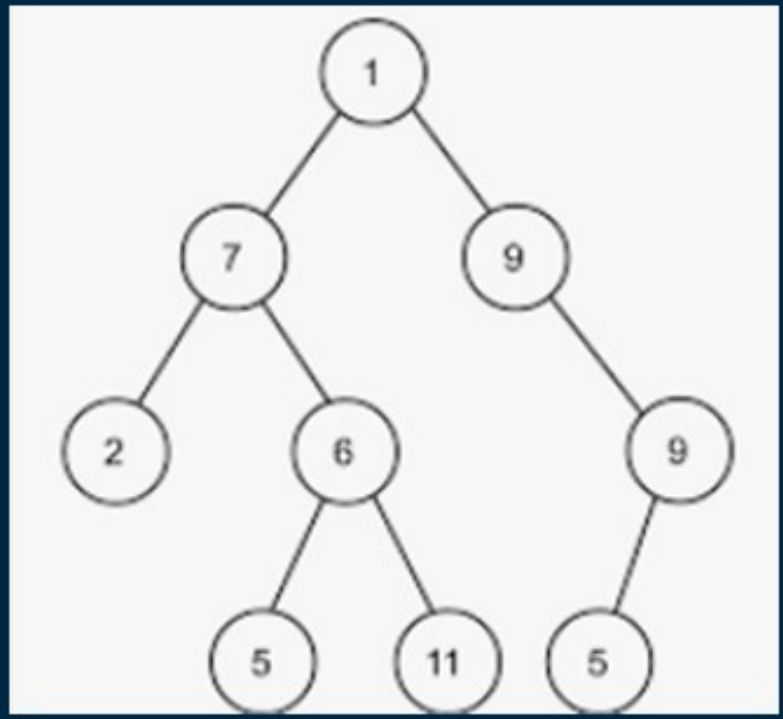
- B-Tree si B+ Tree: index BD
- Space Partitioning Tree: jocuri de mari dimensiuni
- Quadtree
- Tree pyramid (T-pyramid)
- Octree
- k-d (K dimensional) tree
- R-tree: pentru a detrmina calea ce amai scurtă sau obiectele apropiate în grafuri 3D

Arbore binar

structură de date arborescentă în care fiecare nod părinte poate avea cel mult doi copii.

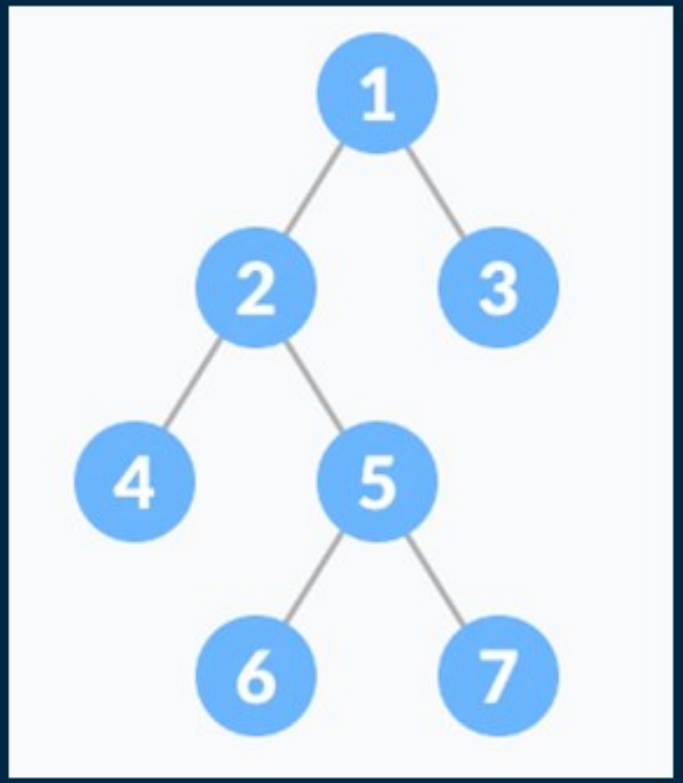
Fiecare nod al unui arbore binar este format din trei elemente:

- element de date
- adresa copilului stâng
- adresa copilului drept



Arbore binar - tipologii

1. Arbore Binar **plin**
   fiecare nod intern are doi sau
   niciun nod subordonat

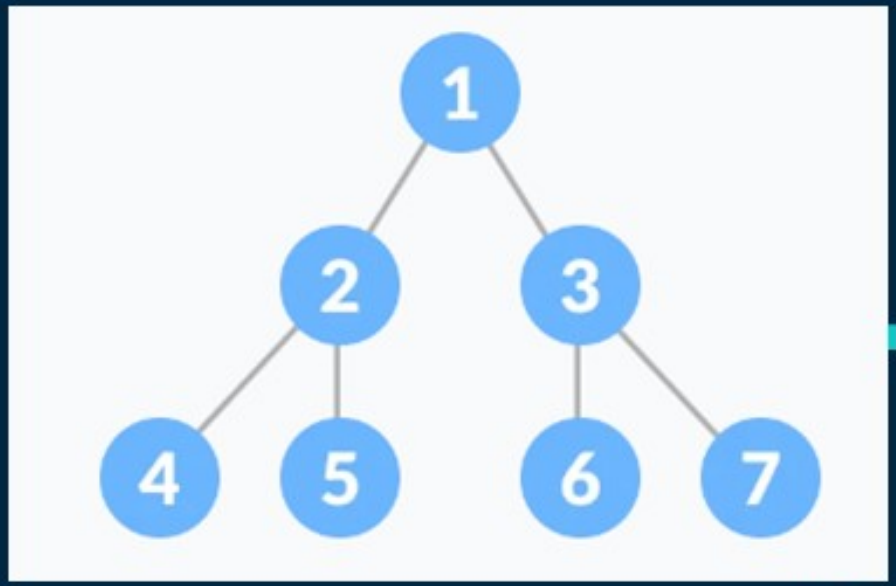


https://www.programiz.com/dsa/binary-tree

Arbore binar – tipologii

2. Arbore Binar perfect

fiecare nod intern are exact două noduri subordonate, iar nordurile externe sunt la acelasi nivel

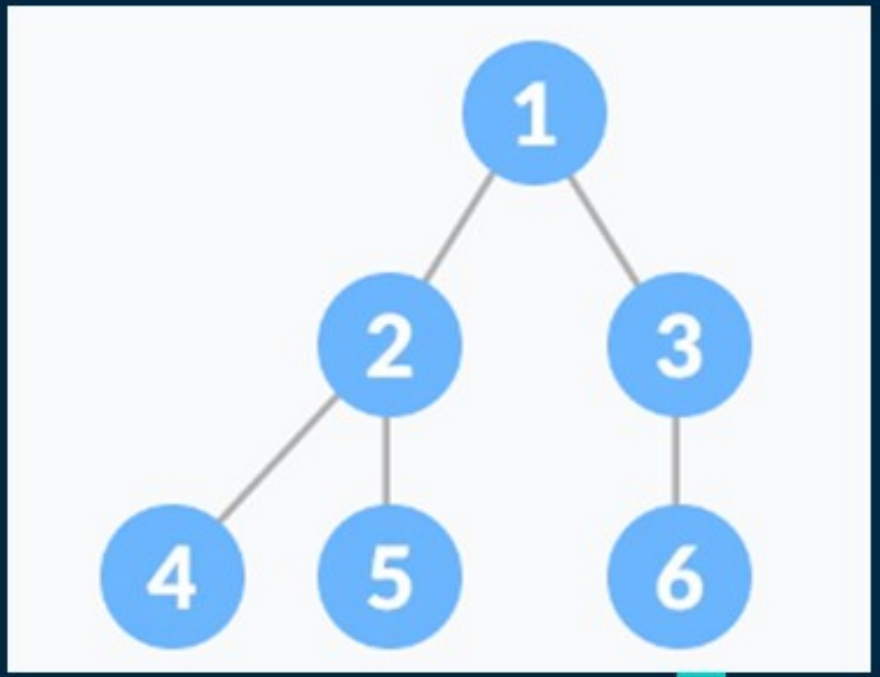


https://www.programiz.com/dsa/binary-tree

Arbore binar – tipologii

3. Arbore Binar complet

fiecare nod intern are
exact două noduri subordonate,
cu excepția celui mai din dreapta nod,
iar nodurile externe sunt la același nivel

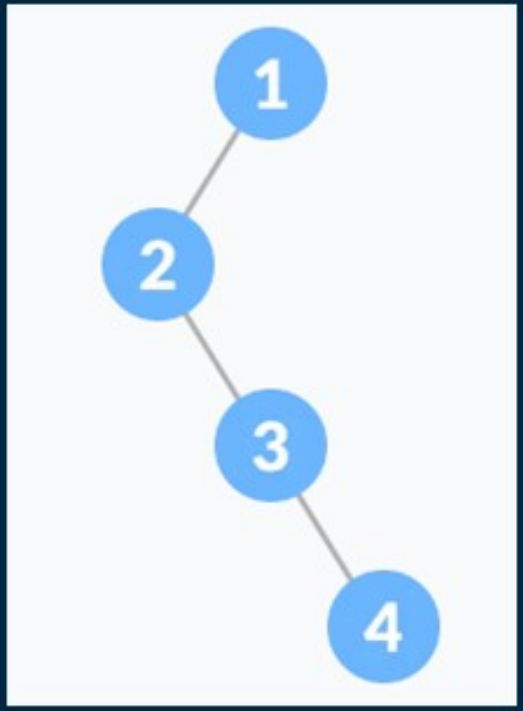


https://www.programiz.com/dsa/binary-tree

Arbore binar – tipologii

4. Arbore Binar degenerat

fiecare nod intern are cel mult un nod subordonat

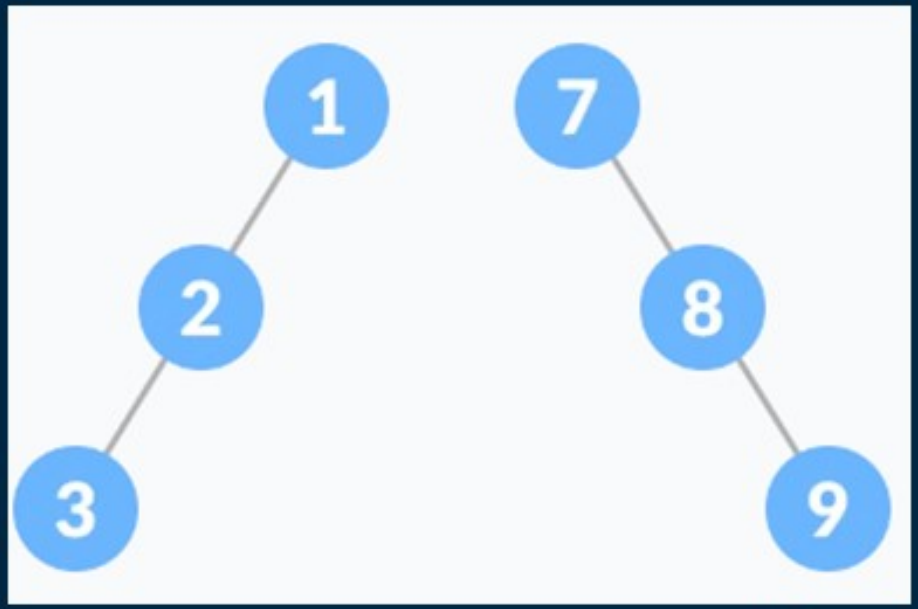


https://www.programiz.com/dsa/binary-tree

Arbore binar – tipologii

5. Arbore Binar debalansat

fiecare nod intern are cel mult un nod subordonat, fie doar stânga, fie doar dreapta

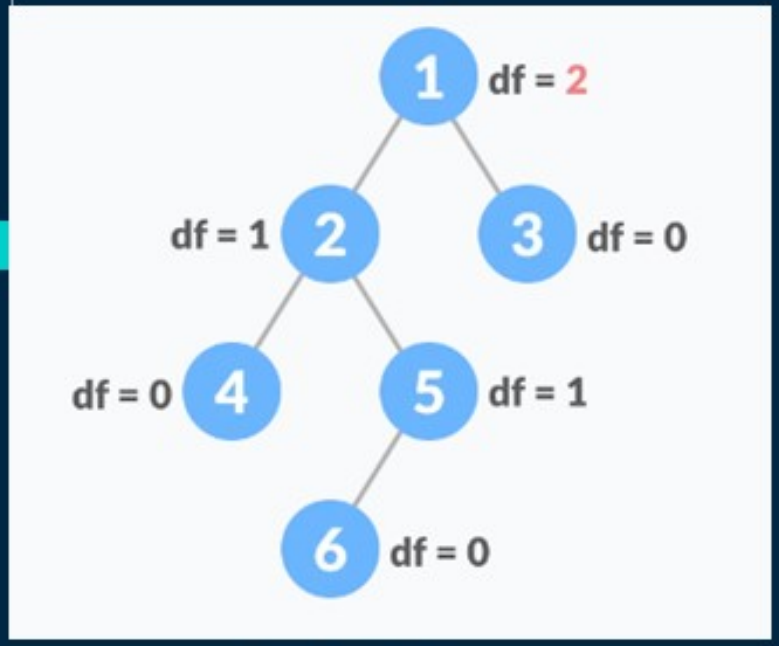


https://www.programiz.com/dsa/binary-tree

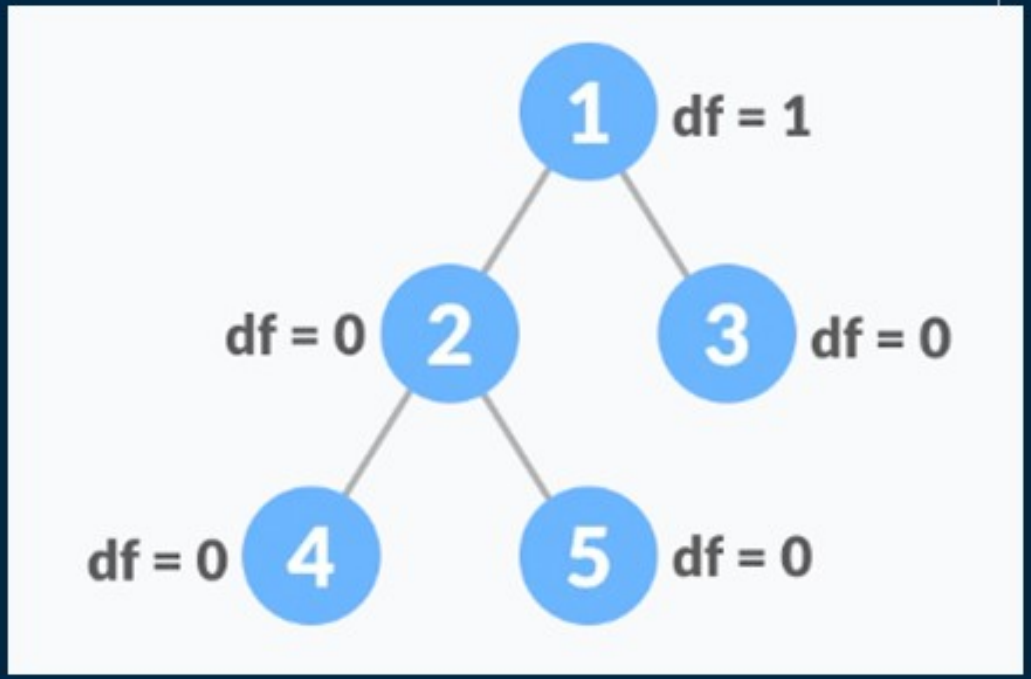
Arbore binar – tipologii

6. Arbore Binar echilibrat

in care diferența dintre înălțimea subarborelui din stânga și din dreapta pentru fiecare nod este fie 0, fie 1



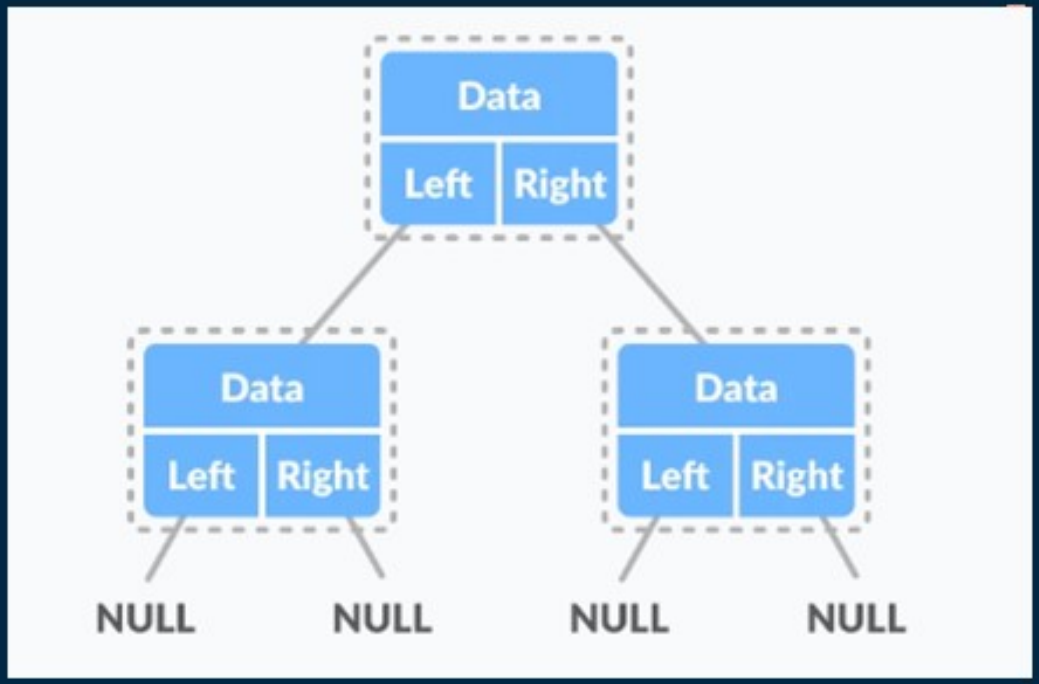
Neechilibrat



https://www.programiz.com/dsa/binary-tree

Arbore binar - reprezentare

```
struct nod {
        int data;
        struct nod *stanga;
        struct nod *dreapta;
};
```



https://www.programiz.com/dsa/binary-tree

Probleme

https://medium.com/techie-delight/binary-tree-interview-questions-and-practice-problems-439df7e5ea1f

https://iq.opengenus.org/list-of-binary-tree-problems/

Aplicații ale arborilor binari

- Acces ușor și rapid la date.
- Algoritmi de rutare in rețele informatice.
- Structuri de date heap.
- Compilatoare - arbore de sintaxă pentru a valida sintaxa fiecărui program.
- Compresie – Huffman
- Merkle tree, blockchain
  - asigurarea integrității datelor identice în sistemele distribuite
  - verificarea inconsecvenței (modificarea datelor)
  - identificarea corectă a datelor modificate
  - sisteme de monedă virtuală

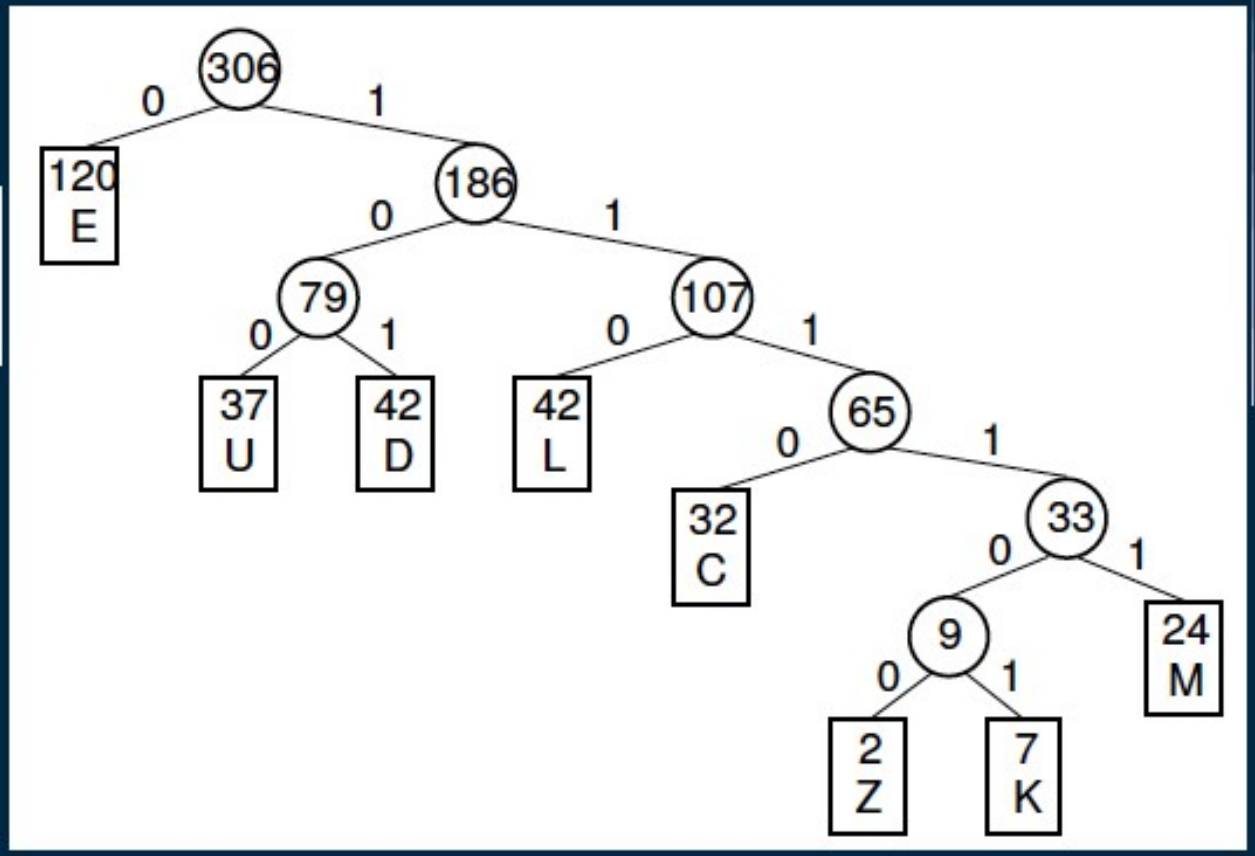
Aplicații ale arborilor binari

- Generare functii de numere pseudoaleatoare - Goldreich, Goldwasser and Micali (GGM)
- Paul-Carole game model - Scapegoat tree
- Treap – un arbore binar care pastreaza proprietatea de ABC si Heap

Exemple

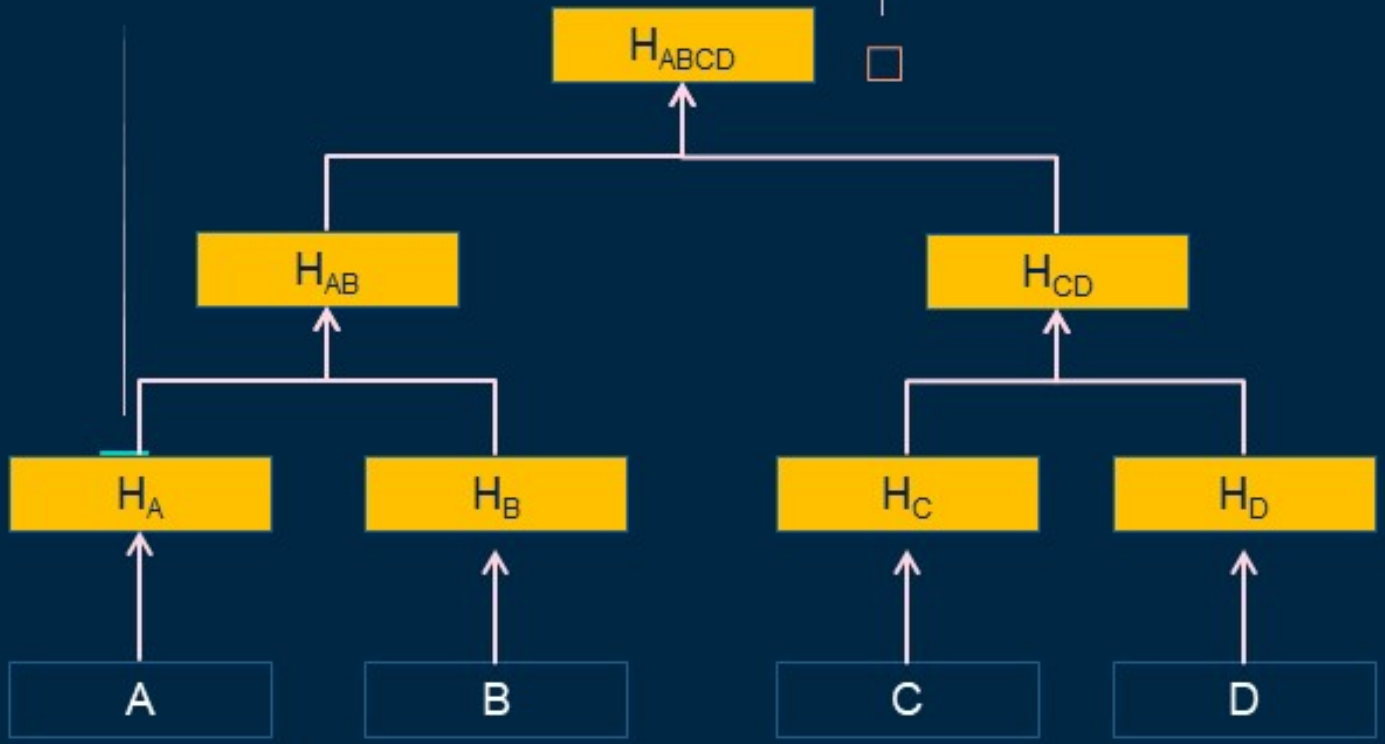
Letter	Z	K	M	C	U	D	L	E
Frequency	2	7	24	32	37	42	42	120

DEED 10100101 (8 bits)
MUCK 1111110011101111101 (18 bits)



Huffman

Exemple



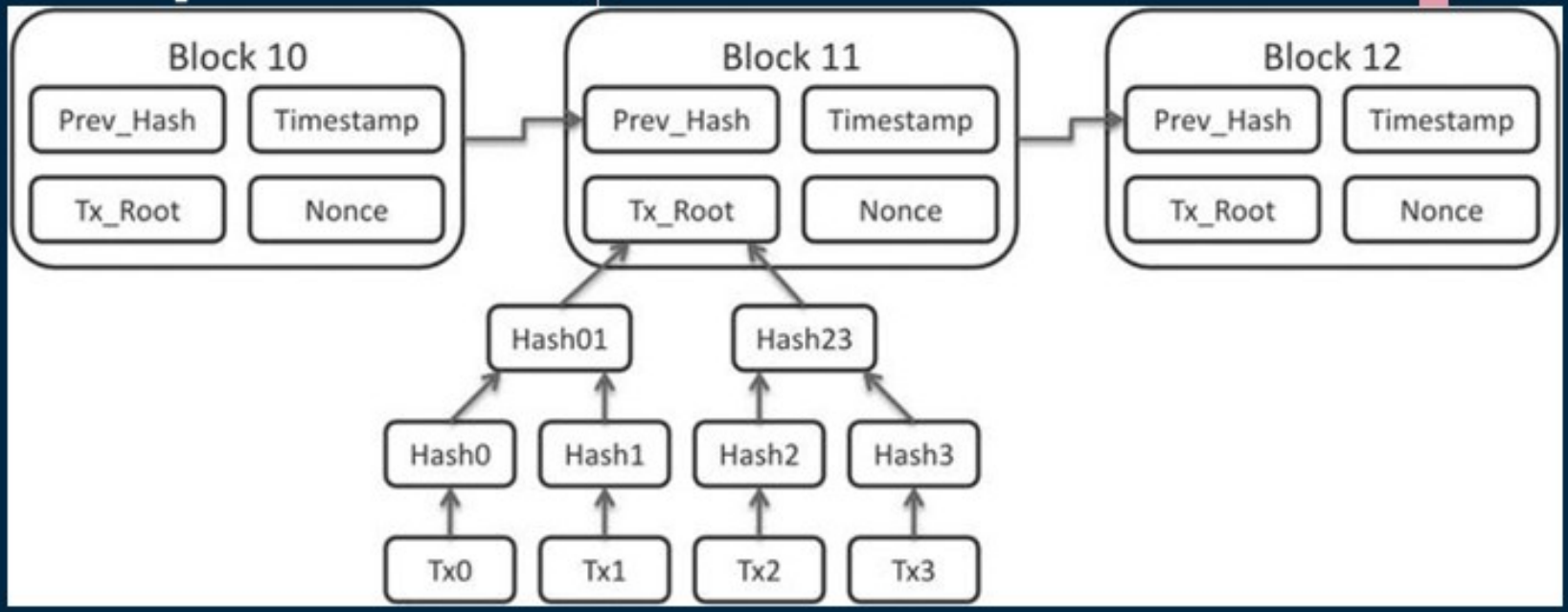
$H_{ABCD}=hash(H_{AB}+H_{CD})$

$H_{XY}=hash(hash(x)+hash(y))$

$H_X=hash(X)$

Merkle tree

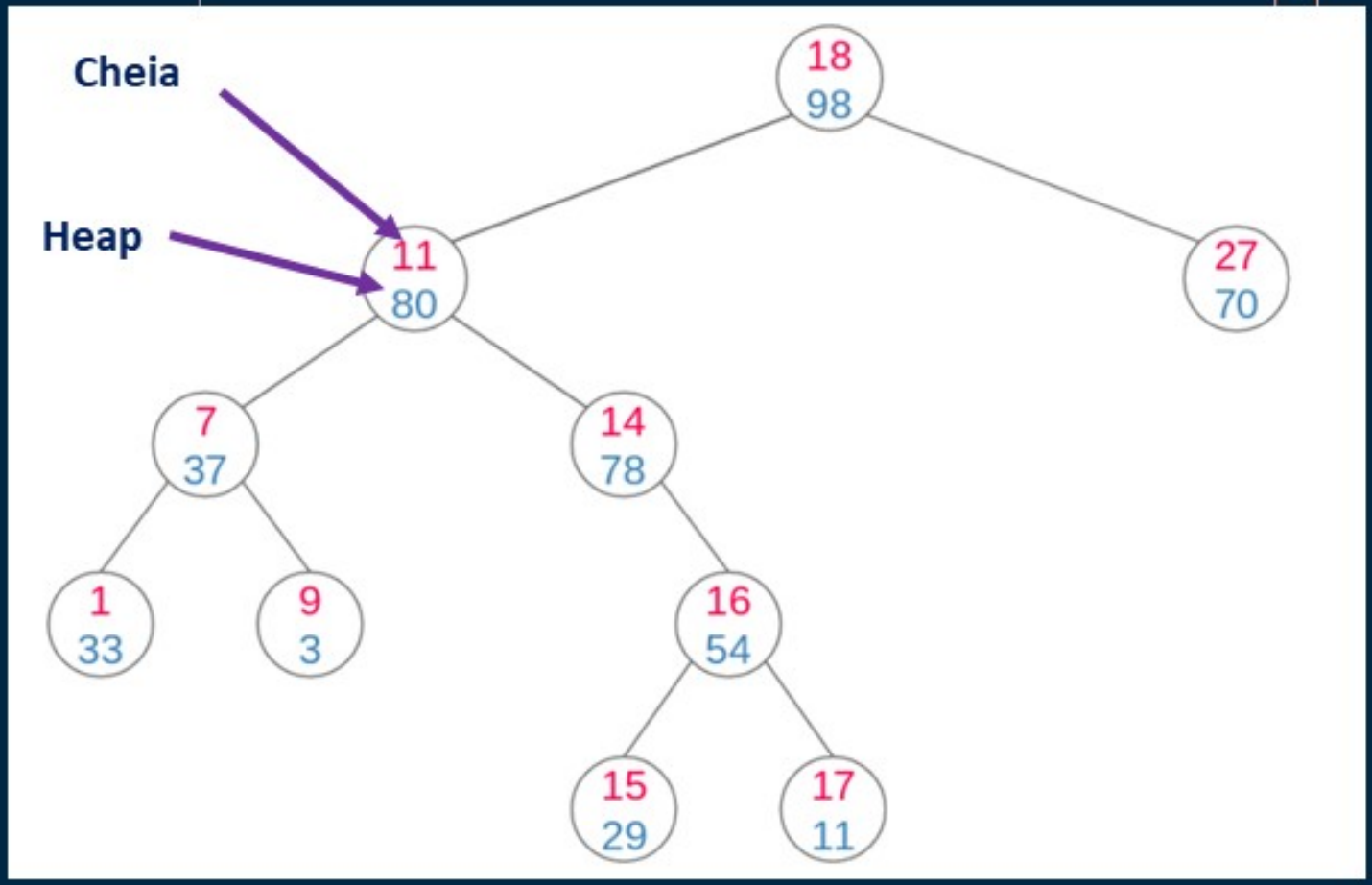
Exemple



Blockchain

- **Integritate date**
- **Non – repudiere**
- **Disponibilitate**
- **Confidentialitate**

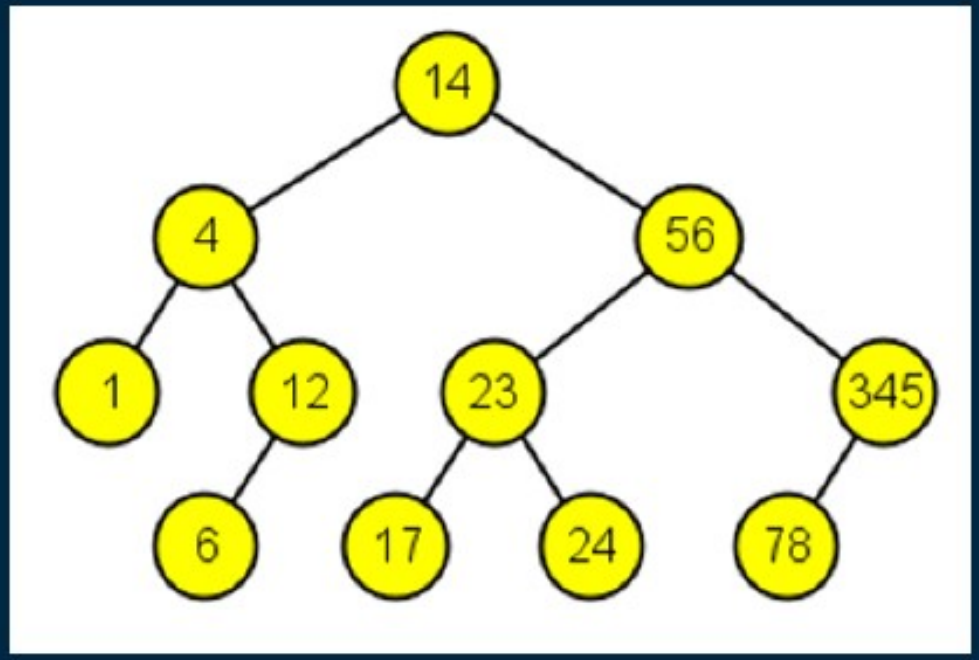
Exemple



Treap

Exemple

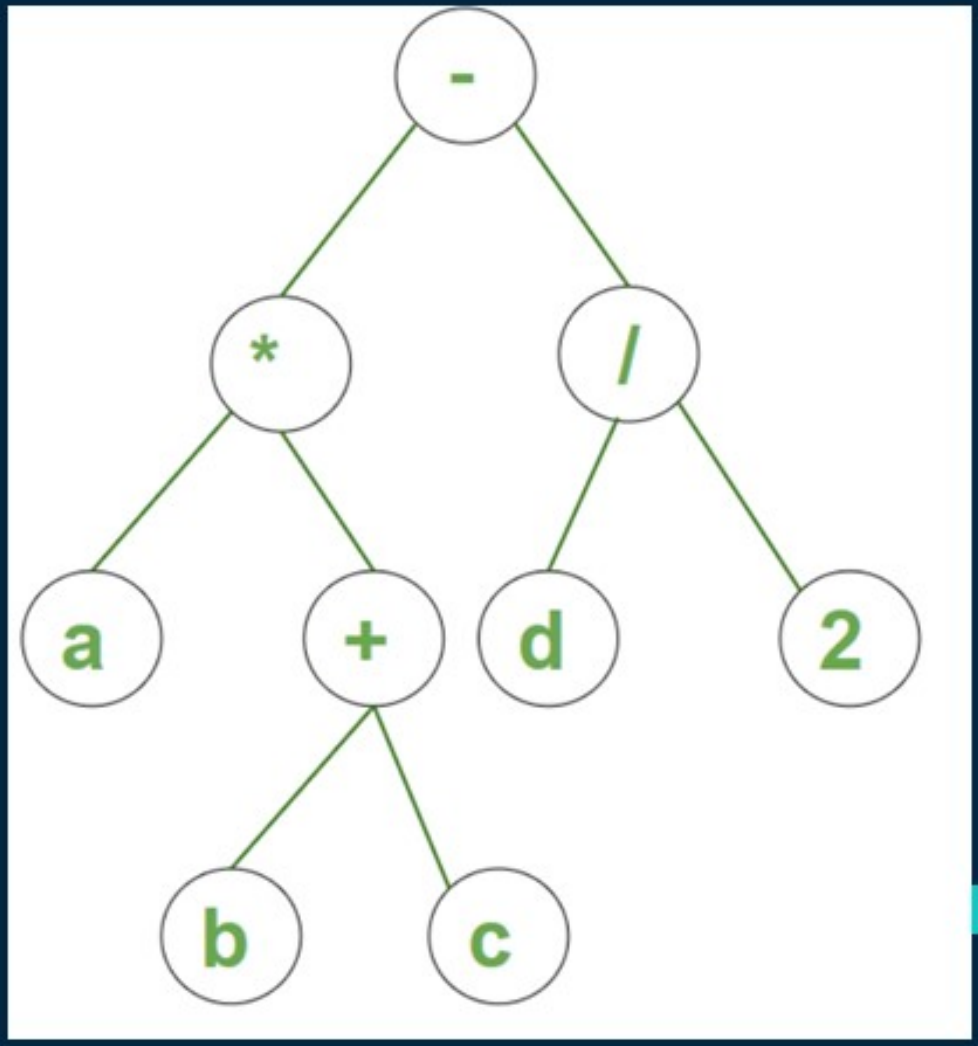
un arbore Scapegoat este un arbore de căutare binar cu auto-echilibrare



Scapegoat tree
(Țap ispășitor)

Exemple

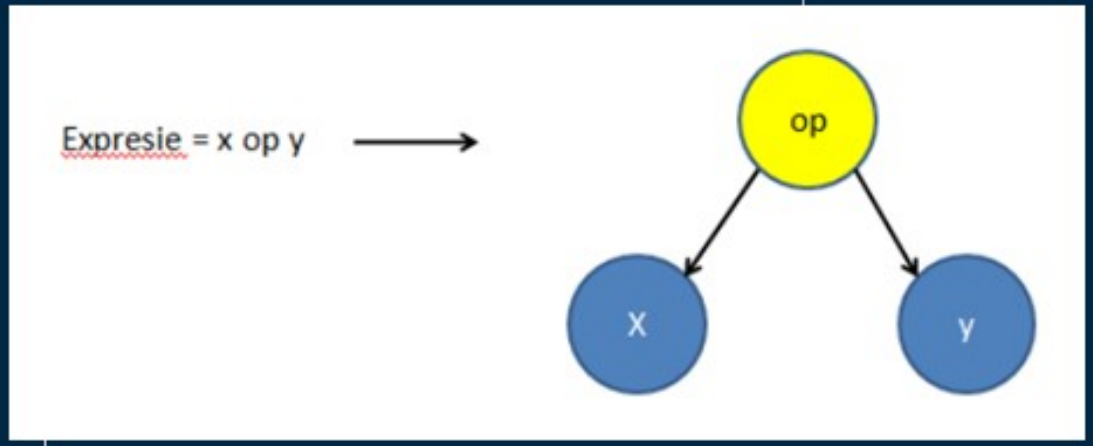
a * (b + c) – d /2

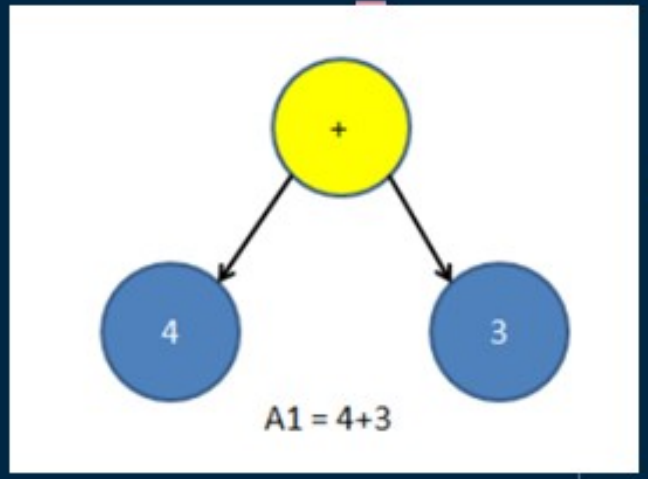


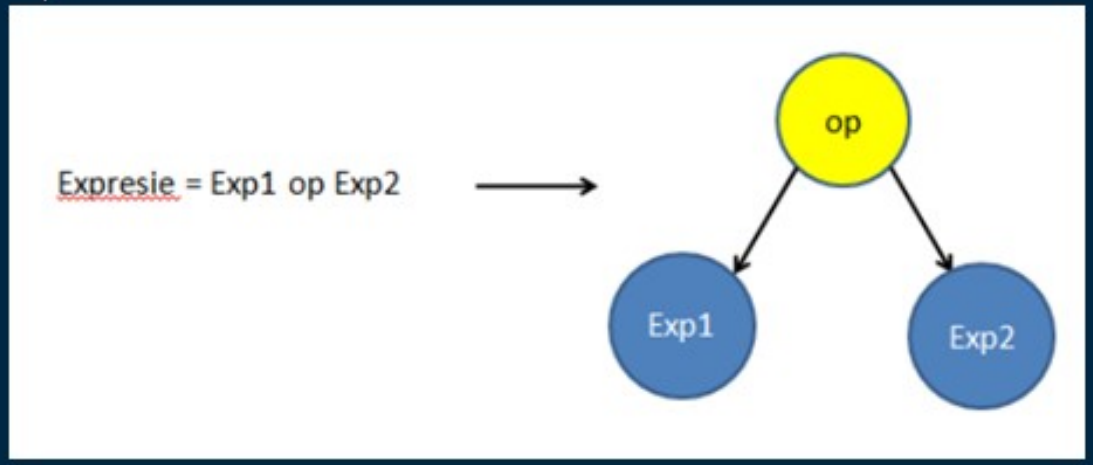
Arbore de sintaxa

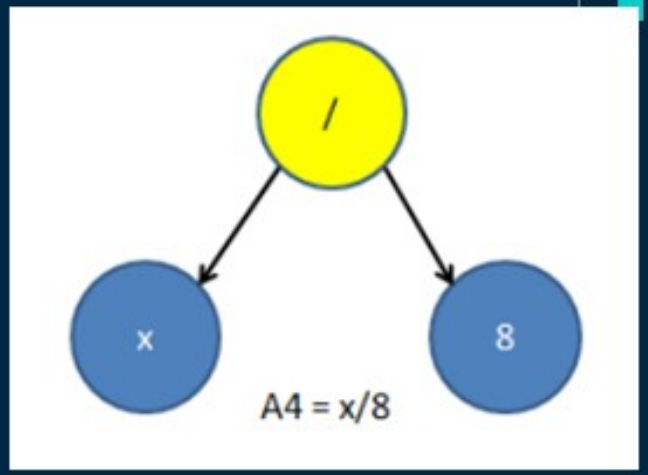
Exemple

Forma poloneza a expresiilor aritmetice







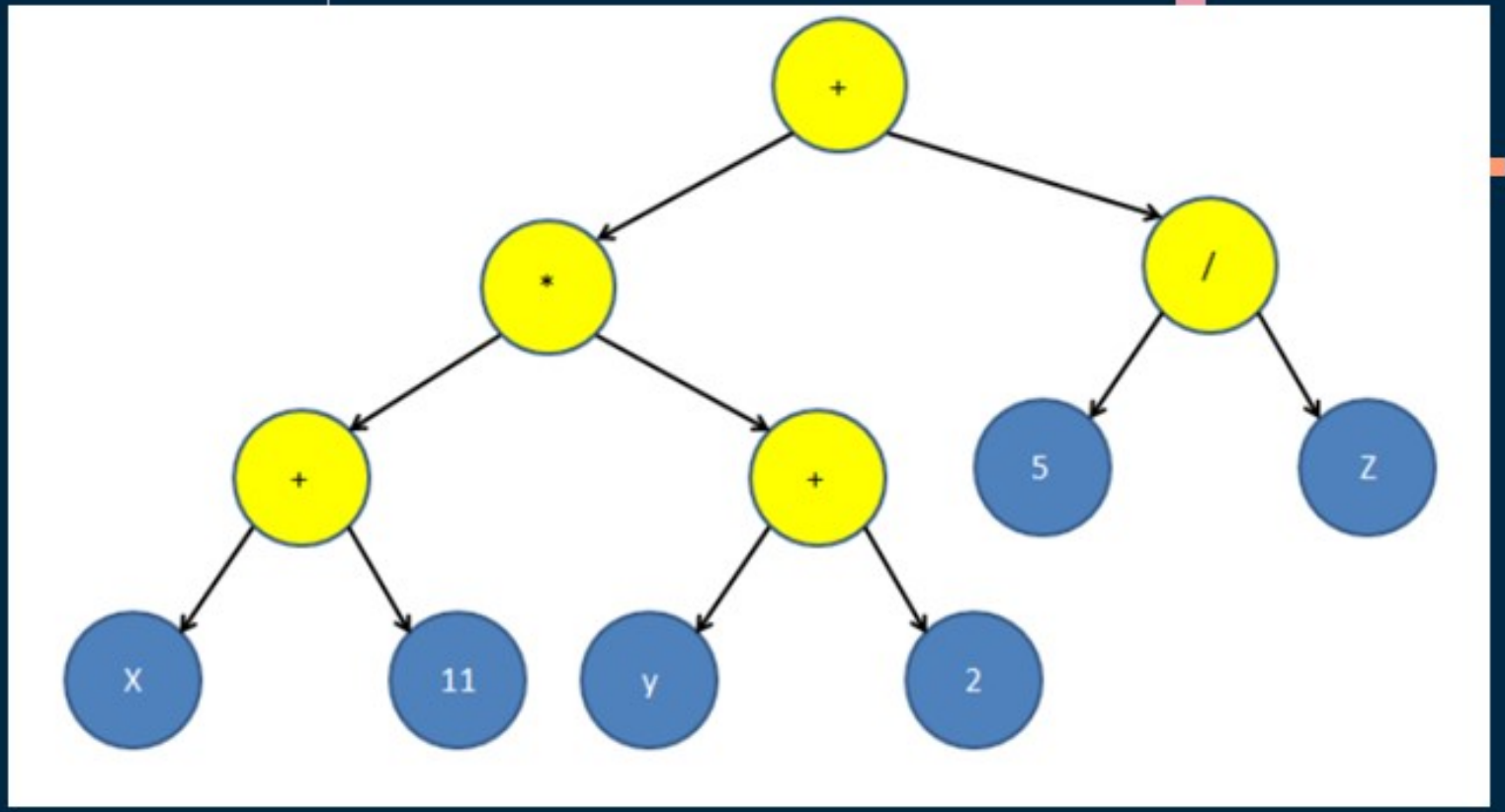


Exemple

Forma poloneza a expresiilor aritmetice

E= (x+11)*(y+2)+5/z

Infix



Problema

Care este adâncimea nodului E? dar a nodului A?

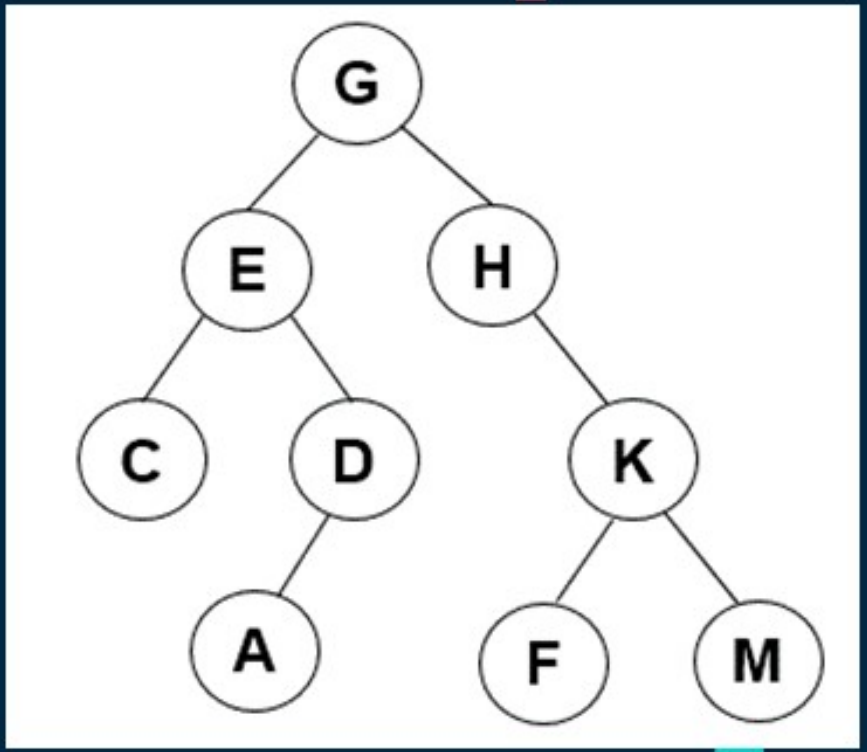
Care este înălțimea arborelui?

Care este ordinea cheilor pentru traversare PreOrdine ?

Care este ordinea cheilor pentru traversare InOrdine ?

Care este ordinea cheilor pentru traversare PostOrdine ?

Care ar fi rezultatul unei traversări în Latime?



Intrebari?

dorin.lordache@365.univ-ovidius.ro

Mulțumesc