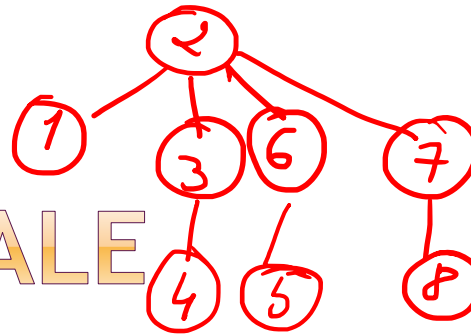
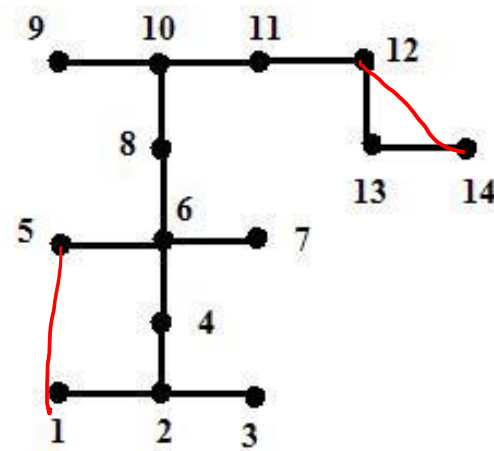
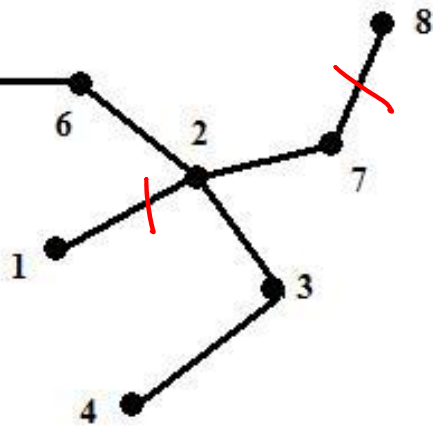


ARBORI

Curs -Algoritmi si structuri de date-an II



NOTIUNI GENERALE

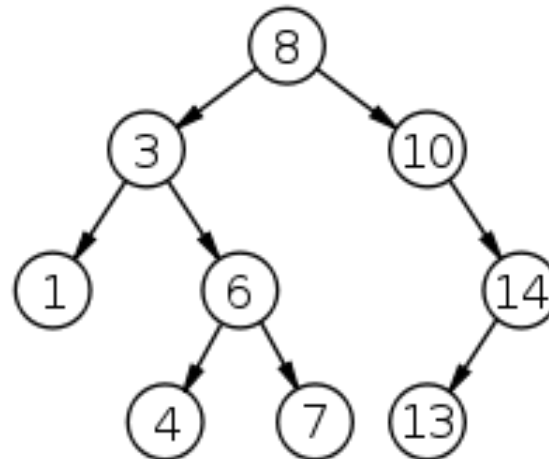


- Un graf neorientat conex fara cicluri se numeste arbore.

TEOREMA DE CARACTERIZARE A ARBORILOR:

- ◉ ***Daca G este un graf neorientat cu $n = \text{numar de varfuri}$, $n \geq 3$ atunci urmatoarele conditii sunt echivalente:***
 - *G este arbore;*
 - *G este minimal conex (G este conex dar daca este eliminata orice muchie a grafului graful obtinut nu mai este conex);*
 - *G este fara cicluri maximal (Intre orice doua varfuri distincte exista exact un singur drum elementar);*
 - *G nu are cicluri si are $n - 1$ muchii;*
 - *G este conex si are $n - 1$ muchii.*
- ◉ **Corolar: *Un arbore cu n varfuri are $n - 1$ muchii.***

ARBORE FOLOSIT IN INFORMATICA



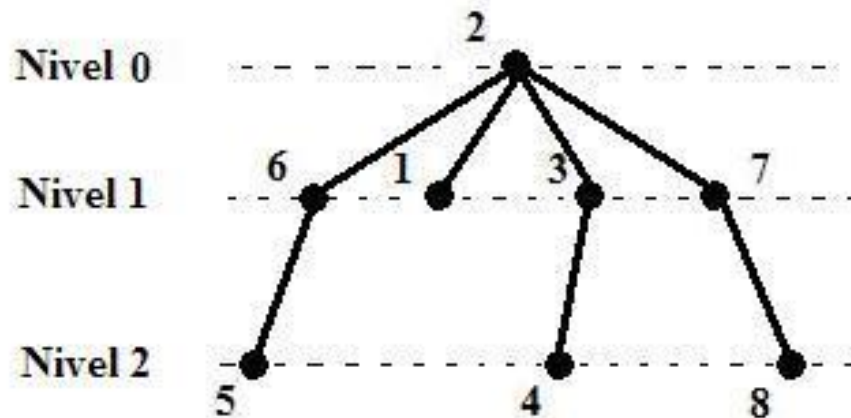
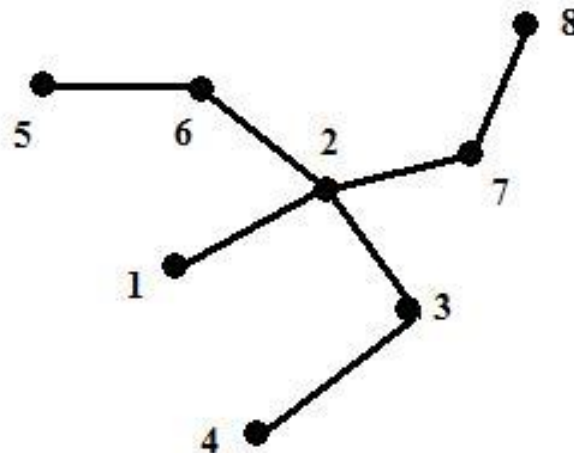
- Definitia de mai sus a notiunii de arbore este cea folosita, in literatura de specialitate, in teoria grafurilor.
- Un tip special de arbore il constituie un arbore in care se pune in evidenta un nod, numit radacina. Acesta este tipul de arbore folosit in informatica. In teoria grafurilor acest tip de arbore se numeste arbore cu radacina.
- In continuare vom folosi aceasta definitie a notiunii de arbore.

REPREZENTAREA ARBORILOR PE NIVELE

- In informatica arborii sunt vizualizati cu radacina in sus si frunzele in jos.
- Intre noduri exista o relatie de tip tata-fiu ca si in cazul arborilor genealogici.
- Nodurile sunt aranjate pe nivele.
 - Pe nivelul 0 se afla un singur nod, radacina.
 - Nodurile fiecarui nivel al aborelui sunt fii nodurilor nivelului precedent.
 - Un nod care are fii se numeste tata.
 - Fii cu acelasi tata se numesc frati.
 - Nodurile care nu au fii se mai numesc frunze sau noduri terminale, iar muchiile dintre noduri, ramuri.

EXEMPLU

- Alegem 2 ca fiind radacina



~~time~~
inălțimea = 2

DEFINITII

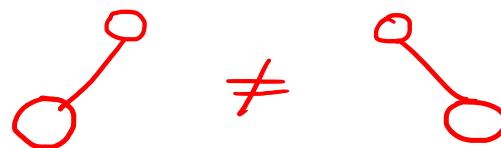
- ◉ Un nod al unui arbore poate avea un numar arbitrar de fii.
- ◉ Daca orice nod al unui arbore nu are mai mult de n fii atunci arborele se numeste arbore n -ar.
- ◉ Un arbore in care orice nod nu are mai mult de 2 fii (descendenti) se numeste arbore binar.
- ◉ Se numeste inaltime a unui arbore lungimea celui mai lung drum de la radacina la un nod terminal din arbore.

EXERCITII

- ◉ Reprezentati arborele de mai sus pe nivele alegand nodul 7 drept radacina.
- ◉ Dați exemplu de un arbore cu cel puțin 5 noduri și înălțime 3.
- ◉ Dați exemplu de un graf care nu este arbore.
- ◉ Dati exemplu de un arbore binar cu cel puțin 7 noduri.

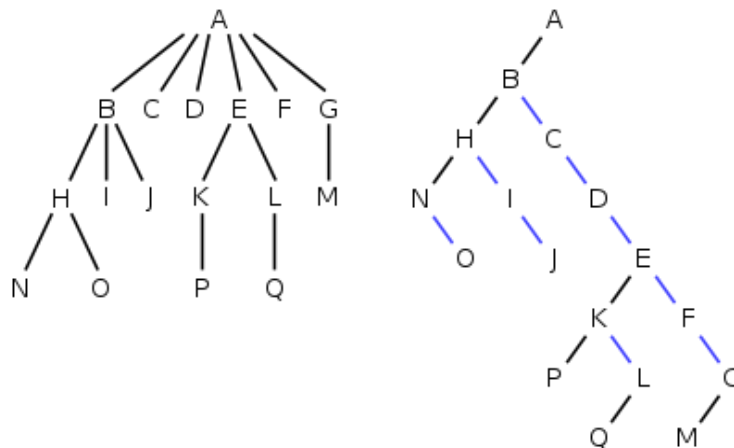
ARBORI BINARI

- Un **arbore binar** este un arbore in care orice nod are cel mult doi descendenți facandu-se distinctie clara intre descendentul drept si descendentul stang.



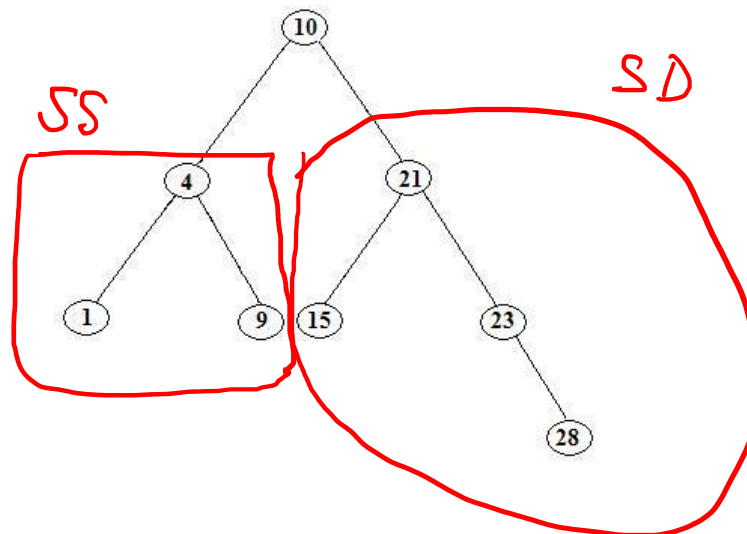
Orice arbore poate fi transformat intr-un arbore binar echivalent.

Imaginea din dreapta este un exemplu.



ARBORI SI SUBARBORI

- Radacina unui arbore binar are doi subarbori:
 - subarborele stang, cel care are drept radacina fiul stang si
 - subarborele drept, cel care are ca radacina fiul drept.
- Orice subarbore al unui arbore binar este el insusi arbore binar.



Radacina 10, are drept fiu stang nodul 4, iar fiu drept nodul 21. Nodul 21 are subarborele stang format din nodul 15 si subarborele drept format din nodurile 23 si 28.

ST:

2	4	6	0	0	0	0	0
---	---	---	---	---	---	---	---

DR:

3	5	7	0	0	0	8	0
---	---	---	---	---	---	---	---

REPREZENTAREA SECVENTIALA

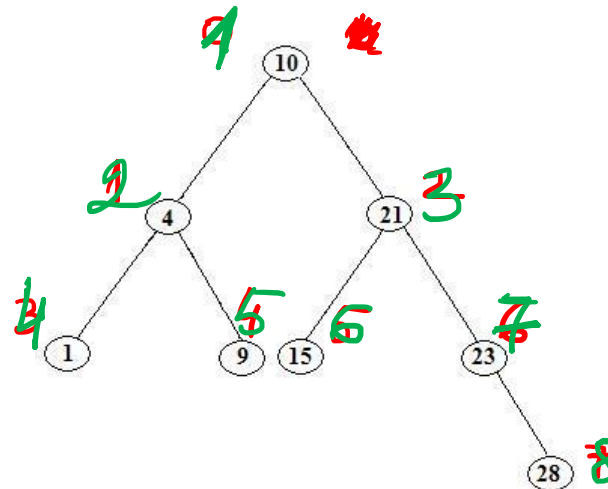
- Pentru fiecare nod al arborelui se precizeaza informatia si descendentii directi ca elemente a trei vector diferiti: INFO, ST SI DR. Daca i este indicele asociat unui nod:

- $INFO[i]$ = informatia nodului i
- $ST[i]$ = fiul stang al nodului i
- $DR[i]$ = fiul drept al nodului i

Alternativa

vectori tati:

(0, 1, 1, 2, 2, 3, 3, 7)



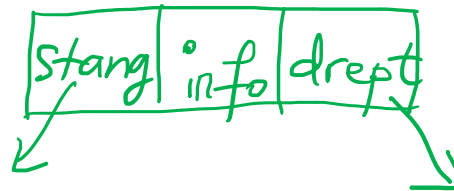
- Pentru arborele de mai sus, daca numerotam nodurile incepand cu nivelul 0, de la stanga la dreapta, obtinem urmasorii vectori, cu conventia ca radacina este nodul 1:
- ■ $INFO = (10, 4, 21, 1, 9, 15, 23, 28)$,
 - ■ $ST = (2, 4, 6, 0, 0, 0, 0, 0)$,
 - $DR = (3, 5, 7, 0, 0, 0, 8, 0)$.

$ST[i] = \text{nodul stang al lui } i$

REPREZENTAREA INLANTUITA

- Pentru fiecare nod al arborelui se precizeaza informatia si descendenti directi ca elemente ale unei structuri definita astfel:

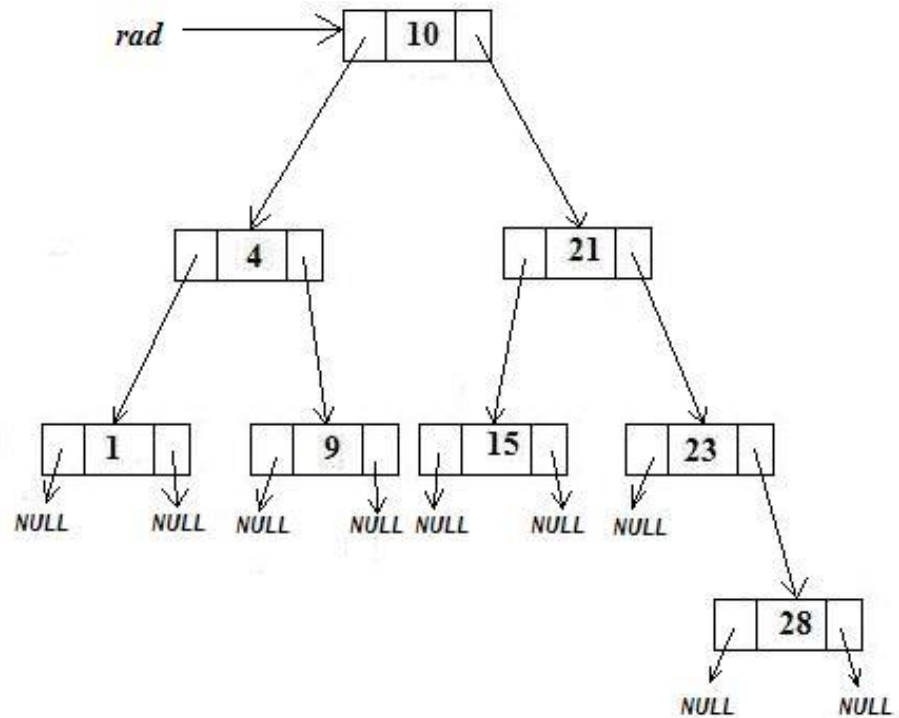
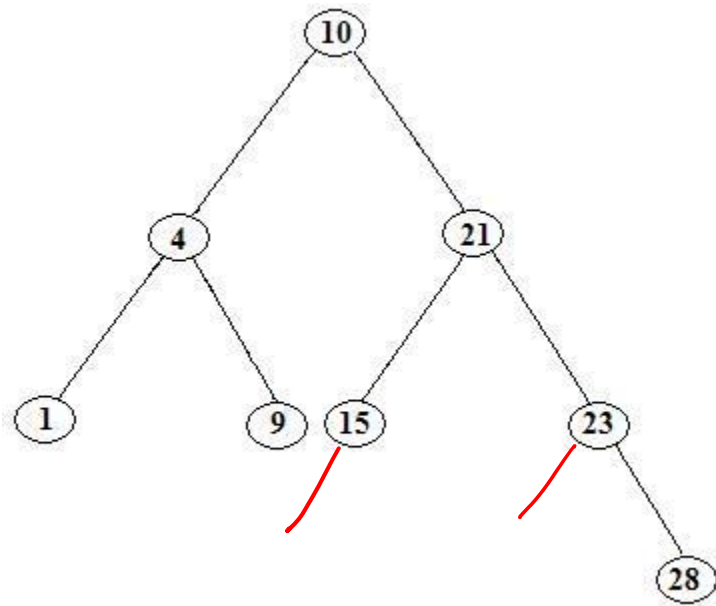
```
struct NodArb  
{  
    T info;  
    NodArb *stang;  
    NodArb *drept;  
};
```



unde

- *T* este presupus definit anterior (eventual printr-o definitie *typedef*),
 - *stang* este pointer la subarborele stang al nodului, iar
 - *drept* este pointer la subarborele drept al nodului.
- Pentru identificarea radacinii arborelui vom defini *NodArb *rad*; un pointer la radacina arborelui.
 - Daca unul din subarbori este vid, atunci pointerul la acel subarbore este NULL.

EXEMPLU



TRAVERSARE (PARCURGERE)

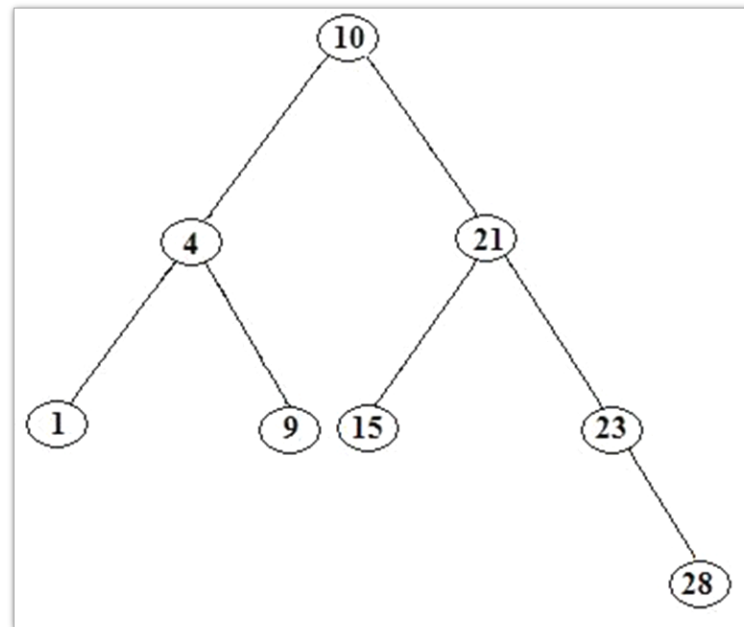
- ⊙ Traversare=examinare a tuturor nodurilor unui arbore
- ⊙ Pentru un arbore binar se poate face:
 - in **preordine (RSD)**: intai vizitam radacina arborelui, apoi subarborele stang urmat de subarborele drept
 - in **inordine (simetrica)(SRD)**: intai vizitam subarborele stang, , apoi radacina arborelui si apoi subarborele drept
 - in **postordine (SDR)**: intai vizitam subarborele stang si subarborele drept si ultima data radacina arborelui.

EXEMPLU

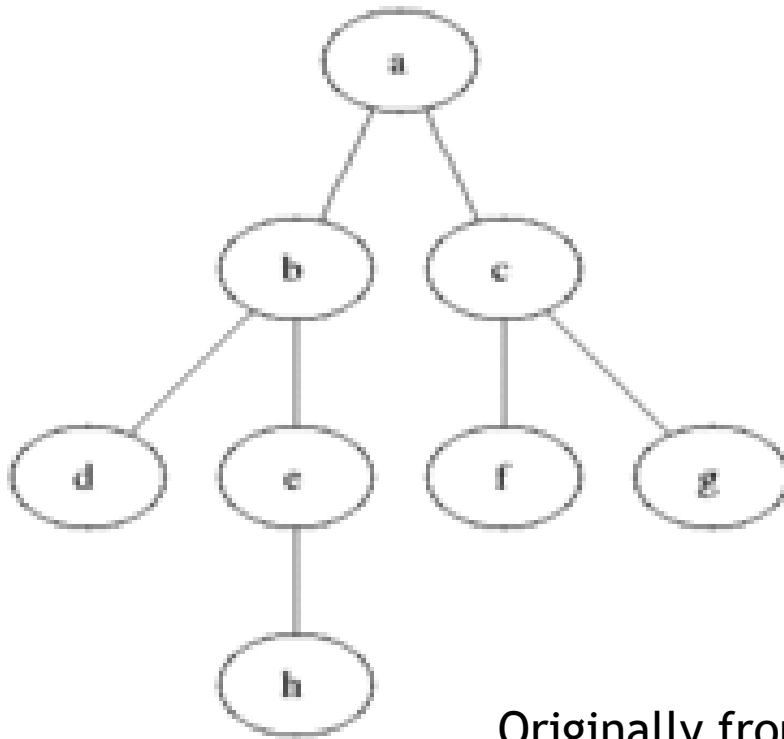
preordine: 10, 4, 1, 9, 21, 15, 23, 28

inordine (simetrica): 1, 4, 9, 10, 15, 21, 23, 28

postordine: 1, 9, 4, 15, 28, 23, 21.



TRAVERSARE IN ADANCIME (DEPTH-FIRST)



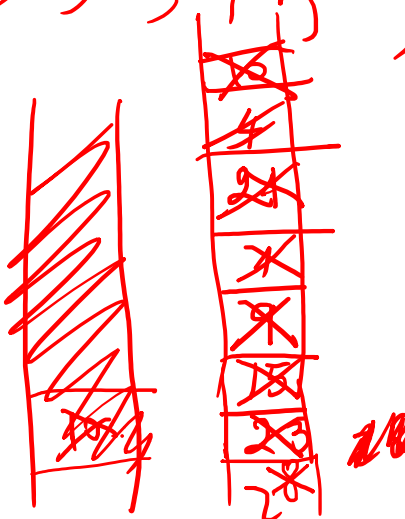
Originally from [en.wikipedia](https://en.wikipedia.org/wiki/Depth-first_search); description page is/was [here](#).

TRAVERSARE IN LATIME (BREADTH -FIRST)

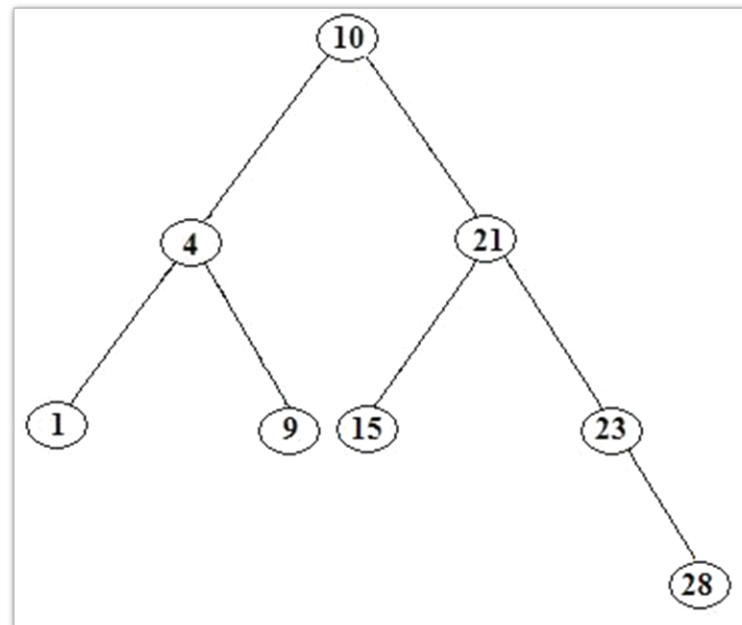
Se viziteaza nodurile pe nivele de la nivelul 0 pana la ultimul nivel, iar pe fiecare nivel de la stanga la dreapta.

10, 4, 21, 1, 9, 15, 23, 28

10, 4, 21, 1, 9, 15, 23, 28



The image shows two vertical columns of boxes representing a queue. The first column has 8 boxes, with the top 7 crossed out and the bottom one containing '4'. The second column has 8 boxes, with the top 7 crossed out and the bottom one containing '28'. To the right of the second column is the handwritten number '28'.



ALGORITHM - TRAVERSARE IN LATIME -NERECURSIV

Se foloseste o coada Q.

Rad->Q

while Q nu este vida

Q-> a, vizitam nodul a

if a->stang #NULL then a->stang -> Q

endif

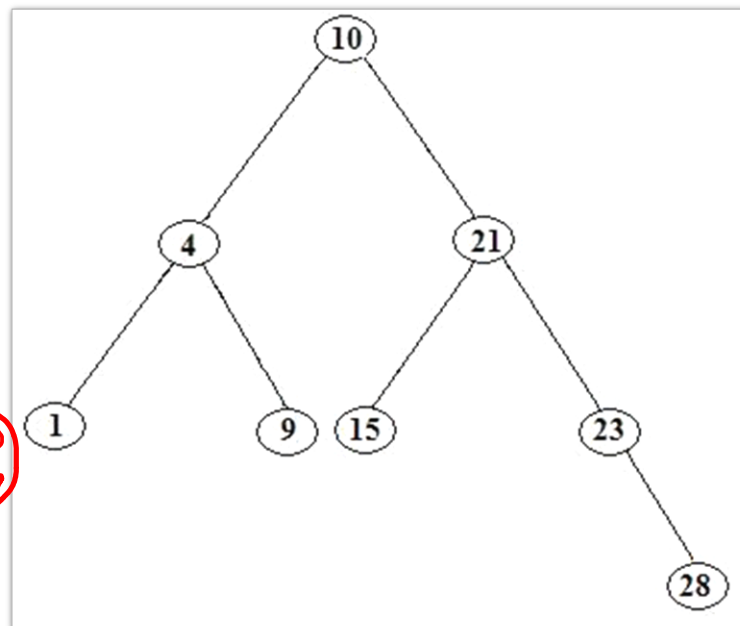
if a->drept #NULL then a->drept -> Q

endif

endwhile

TRAVERSARE IN ADANCIME (DEPTH-FIRST) = PREORDINE

- Nodurile arborelui le vizitam in ordine incepand cu radacina apoi celelalte noduri cat mai adanc in arbore pana cand ajungem la un nod terminal, caz in care ne reintoarcem la tatal acestui nod terminal si incercam sa alegem alta cale si sa vizitam noduri care nu au fost vizitate pana atunci.



10, 4, 1, 9, 21, 15, 23, 28

"

preordinea

10, 4, 1, 9, 21, 15, 23, 28

R

ALGORITHM-PARCURGERE ADANCIME-NERECURSIV

Se foloseste o stiva S.

Rad -> S

While S nu este vida

 S-> a, vizitam nodul a

 if a->drept #NULL then a->drept -> S

 endif

 if a->stang #NULL then a->stang -> S

 endif

endwhile