

Arbori. Arbori binari

SD 2017/2018

Conținut

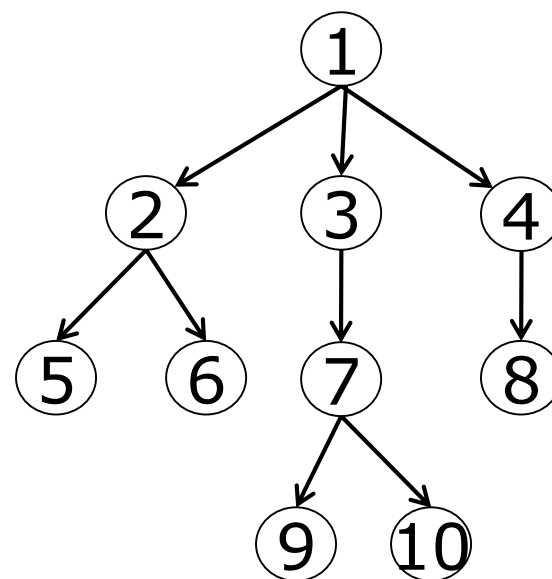
- Arbori
- Arbori binari (**ArbBin**)
- Aplicație: reprezentarea expresiilor ca arbori

Arbori [cu rădăcină]

- Model abstract pentru structuri ierarhice;
- Un arbore este format din noduri legate printr-o relație *părinte-copil*

– **$A = (N, P)$**

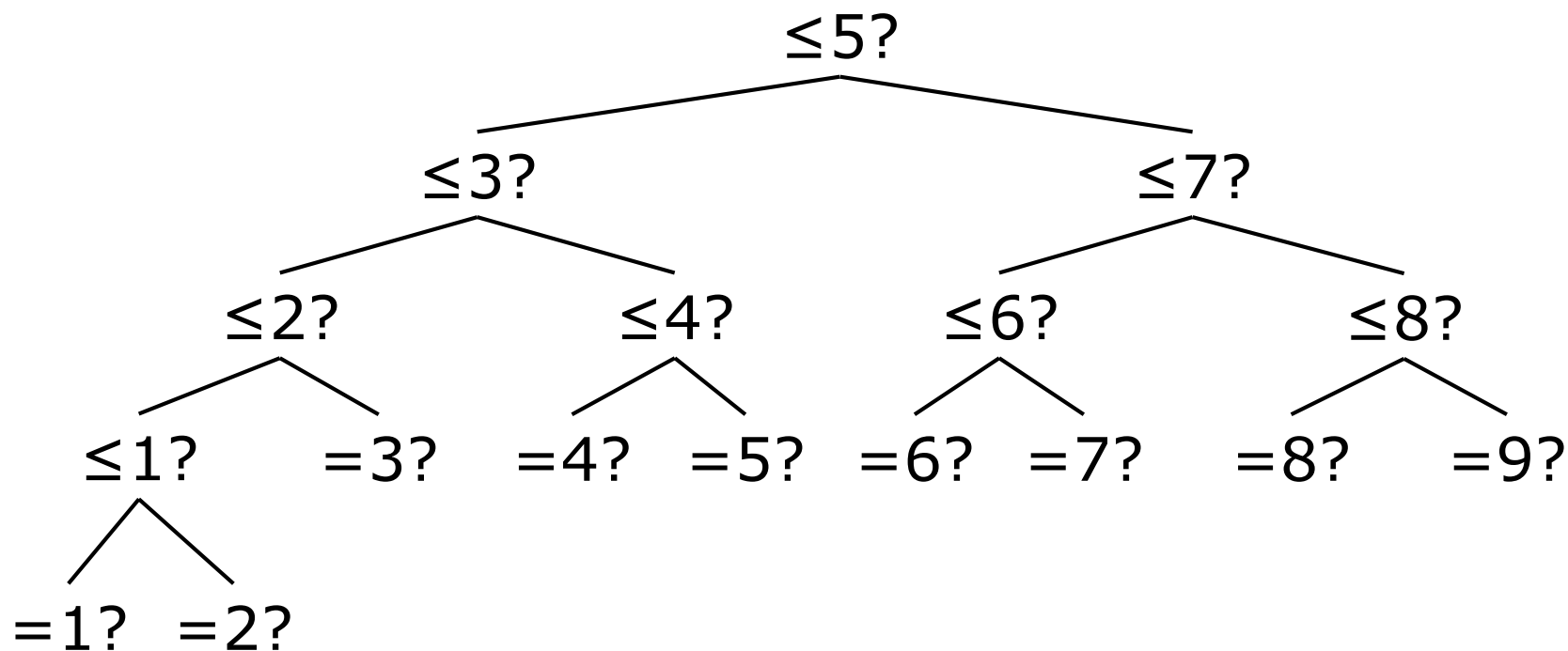
- N mulțimea de noduri;
 - P relație binară peste N , ("*părintele lui*");
 - $r \in N$, nod rădăcină.
- $\forall x \in N, \exists$ un singur drum de la x la r ;
- $\forall x \in N - \{r\}$, x are un singur părinte.



Arbori – aplicații

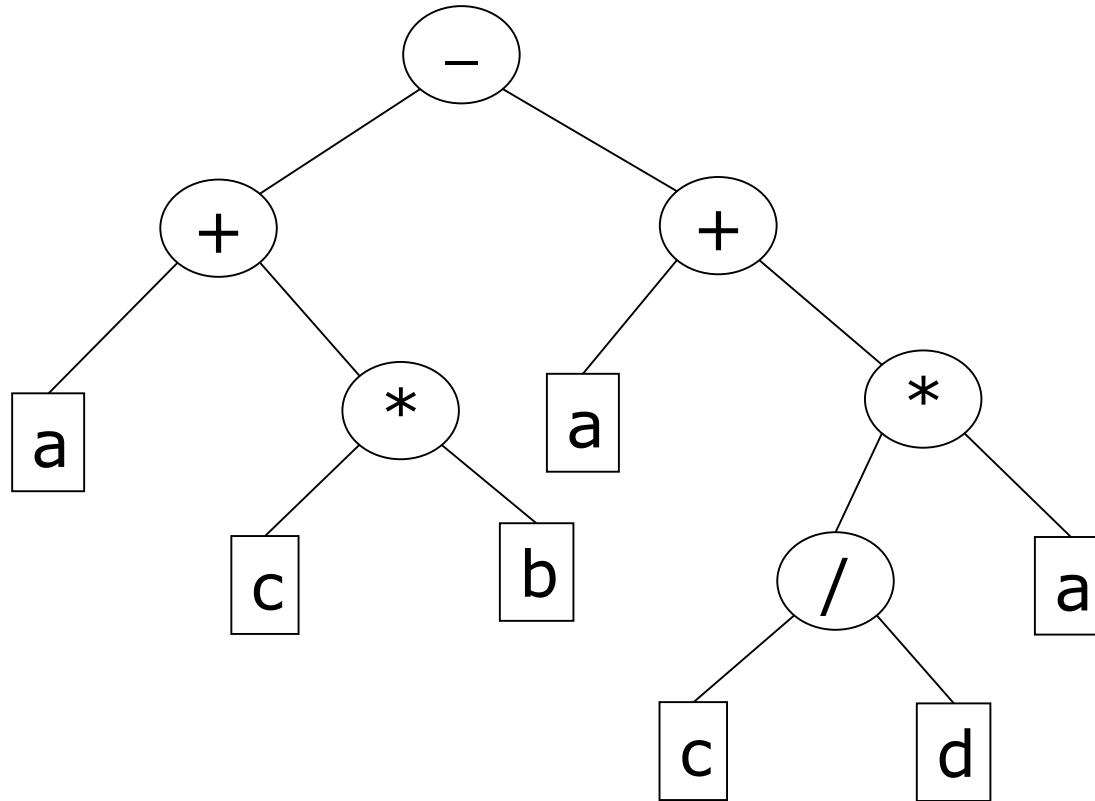
- Arbori genealogici;
- Colecții de cărți în biblioteci;
- Organizarea fișierelor;
- Medii de programare.

Arbori de decizie



1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

Arbori sintactici



$(a + c * b) - (a + c / d * a)$

Arbori pătratici

toată imaginea neagră: ●

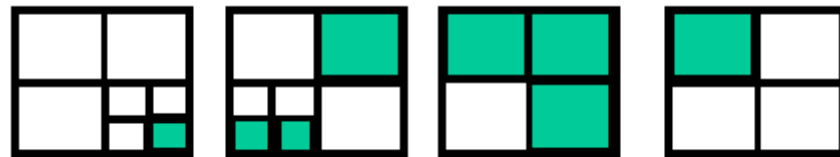
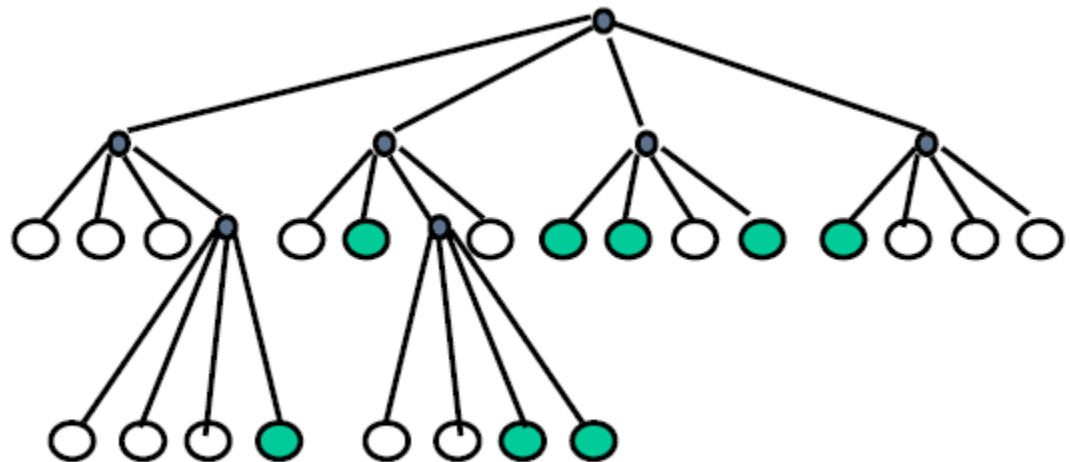
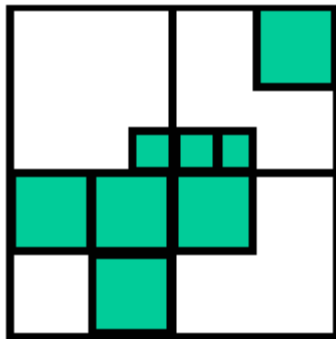
toată imaginea albă: ○

altfel:

1	2
3	4

 =

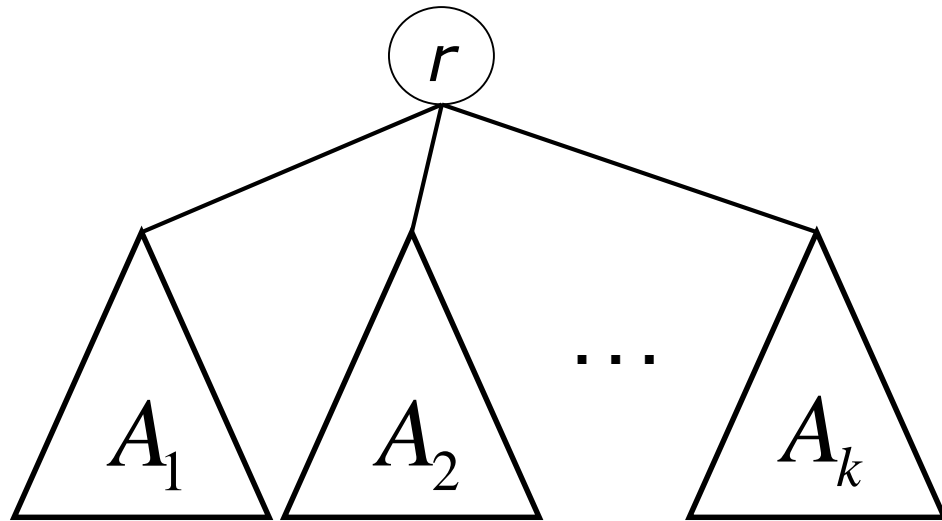
1	2	3	4
---	---	---	---



Arbori: definiție recursivă

$$A = \begin{cases} \Lambda, & \text{arborele vid,} \\ (r, \{A_1, \dots, A_k\}), & r \text{ element, } A_1, \dots, A_k \text{ arbori} \end{cases}$$

$A = \Lambda$ sau

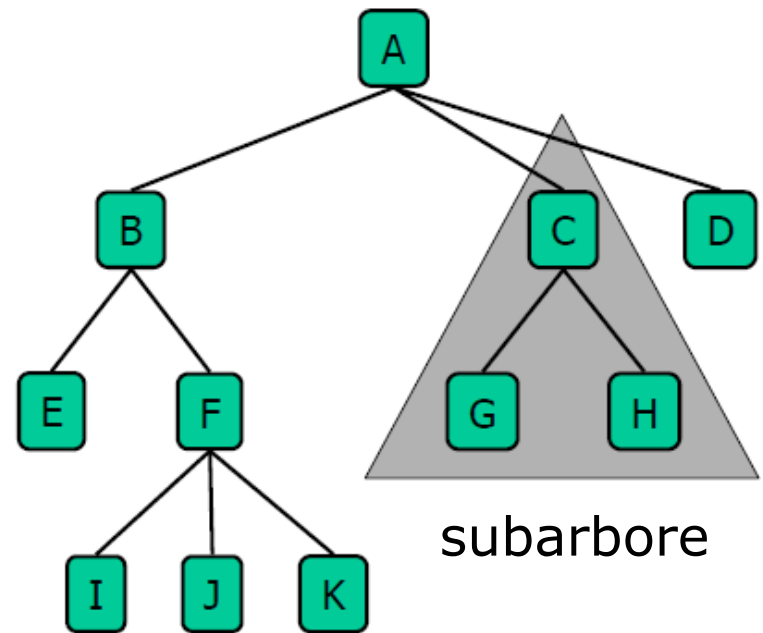


Dacă A este ordonat (planar), atunci



Arbori: terminologie

- Rădăcina: nodul fără părinte
- Nod intern: nod cu cel puțin un fiu
- Nod extern (frunză): nod fără fii
- Descendenții unui nod: fii, nepoți, etc
- Frații unui nod: toate celelalte noduri având același părinte
- Subarbore: arbore format dintr-un nod și descendenții săi

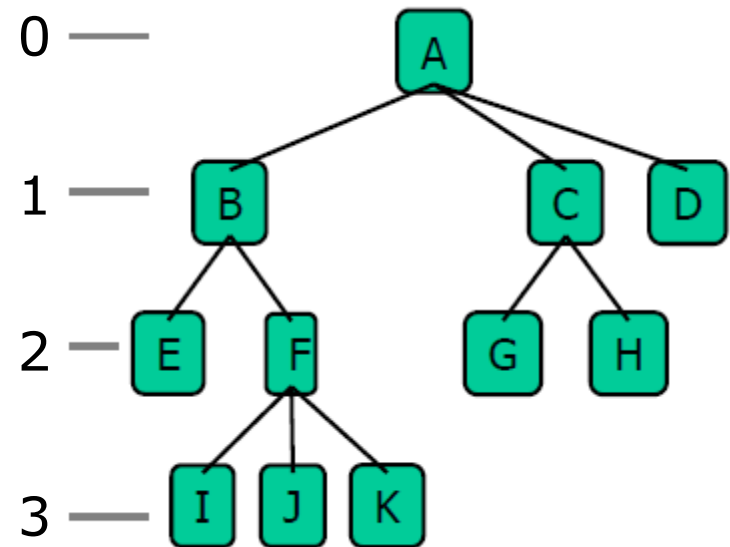


Arbori: terminologie

- Adâncimea unui nod x : numărul de noduri de la rădăcină la x

$$\text{adâncime}(x) = \begin{cases} 0, & \text{dacă } x \text{ este radacina} \\ 1 + \text{adâncime}(\text{părinte}(x)), & \text{în caz contrar} \end{cases}$$

- Înălțimea unui arbore: adâncimea maximă a nodurilor arborelui.
- Înălțimea unui nod x : distanța de la x la cel mai depărtat descendent al său.



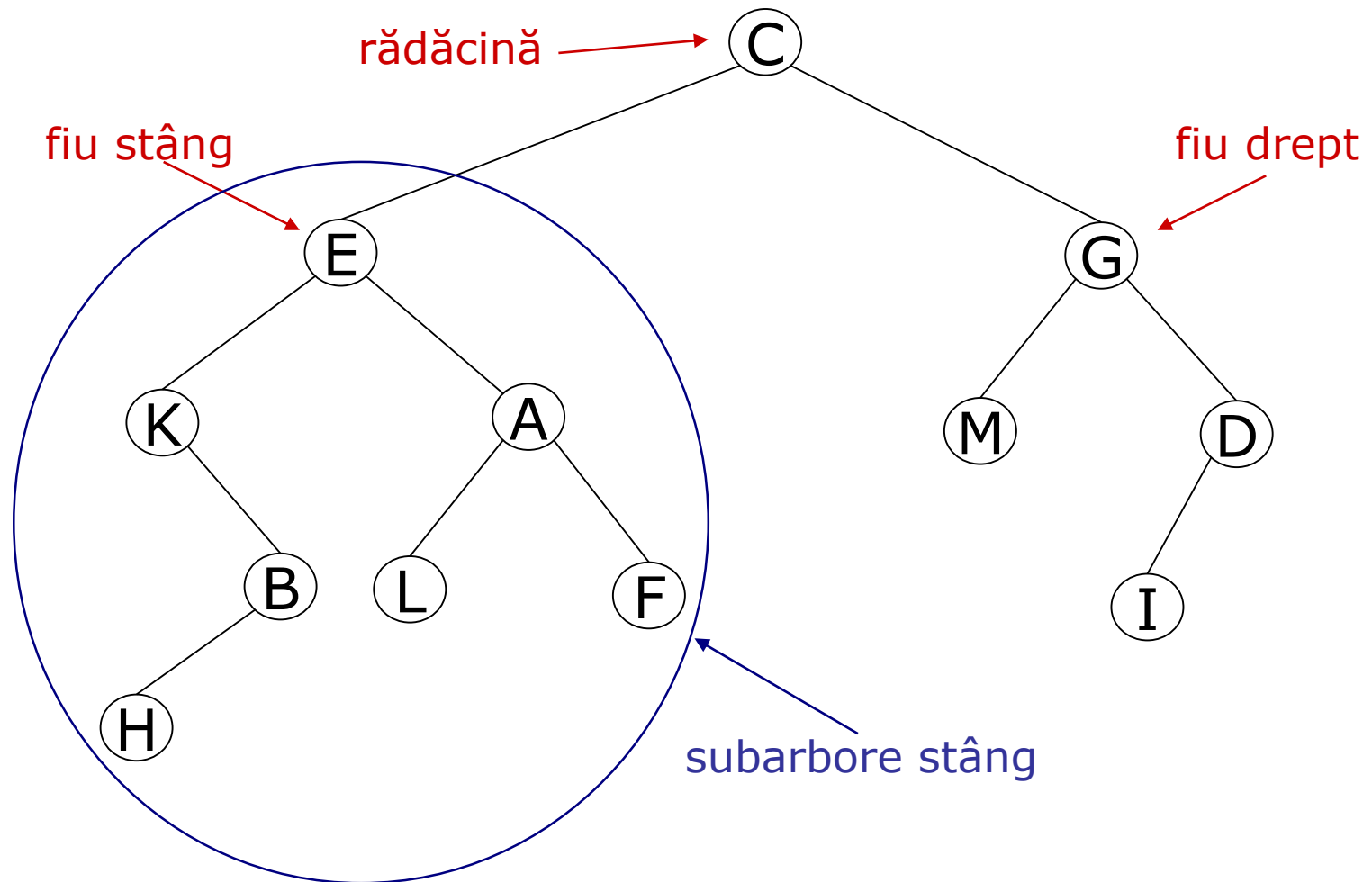
Tipul de date abstract

ArbBin

➤ obiecte : arbori binari.

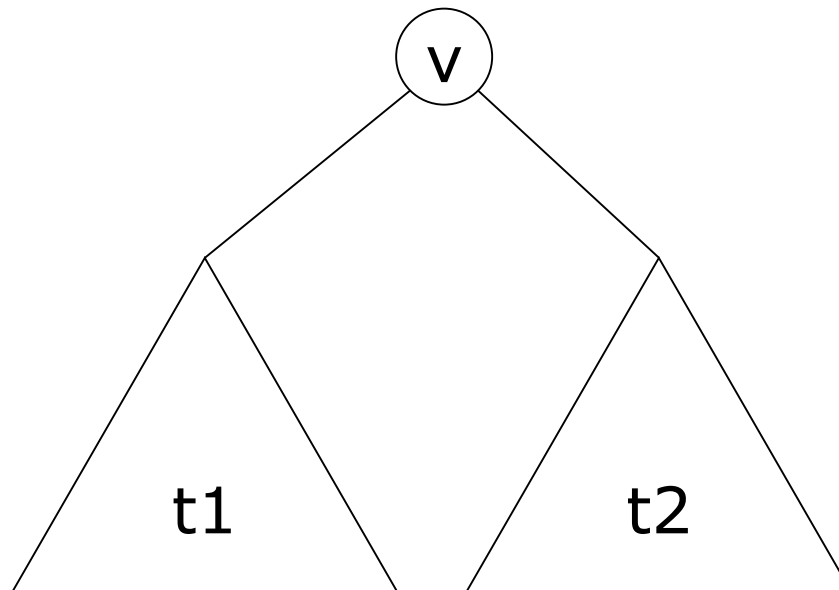
- un arbore binar este o colecție de noduri cu proprietățile:
 1. orice nod are 0, 1 sau 2 succesori (**fii, copii**);
 2. orice nod, exceptând unul singur – **rădăcina**, are un singur nod predecesor (**tatăl, părintele**);
 3. rădăcina nu are predecesori;
 4. fiii sunt ordonați: fiul stâng, fiul drept (daca un nod are un singur fiu, trebuie menționat care);
 5. nodurile fără fii formează frontiera arborelui.

Arbori binari: exemplu



Arbori binari: definiția recursivă

- Arborele cu nici un nod (vid) este arbore binar.
- Dacă **v** este un nod și **t1**, **t2** sunt arbori binari atunci arborele care are pe **v** ca rădăcină, **t1** subarbore stâng al rădăcinii și **t2** subarbore drept al rădăcinii, este arbore binar.



Arbori binari: proprietăți

- Notății

- n numărul de noduri
- n_e numărul de noduri externe
- n_i numărul de noduri interne
- h înălțimea

$$h+1 \leq n \leq 2^{h+1} - 1$$

$$1 \leq n_e \leq 2^h$$

$$\log_2(n+1) - 1 \leq h \leq n-1$$

$$h \leq n_i \leq 2^h - 1$$

Arbori binari: proprietăți

- Arbore propriu: fiecare nod intern are exact doi fii

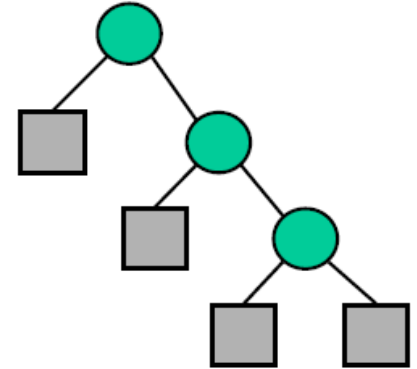
$$2h+1 \leq n \leq 2^{h+1} - 1$$

$$\log_2(n+1) - 1 \leq h \leq (n-1)/2$$

$$h+1 \leq n_e \leq 2^h$$

$$h \leq n_i \leq 2^h - 1$$

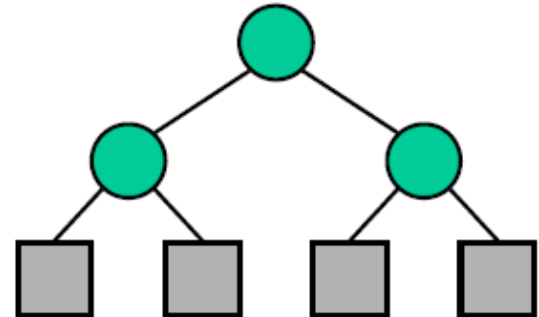
$$n_e = n_i + 1$$



- Arbore complet: arbore propriu în care frunzele au aceeași adâncime

nivelul i are 2^i noduri

$$n = 2^{h+1} - 1 = 2n_e - 1$$



ArbBin: operații

- **insereaza()**

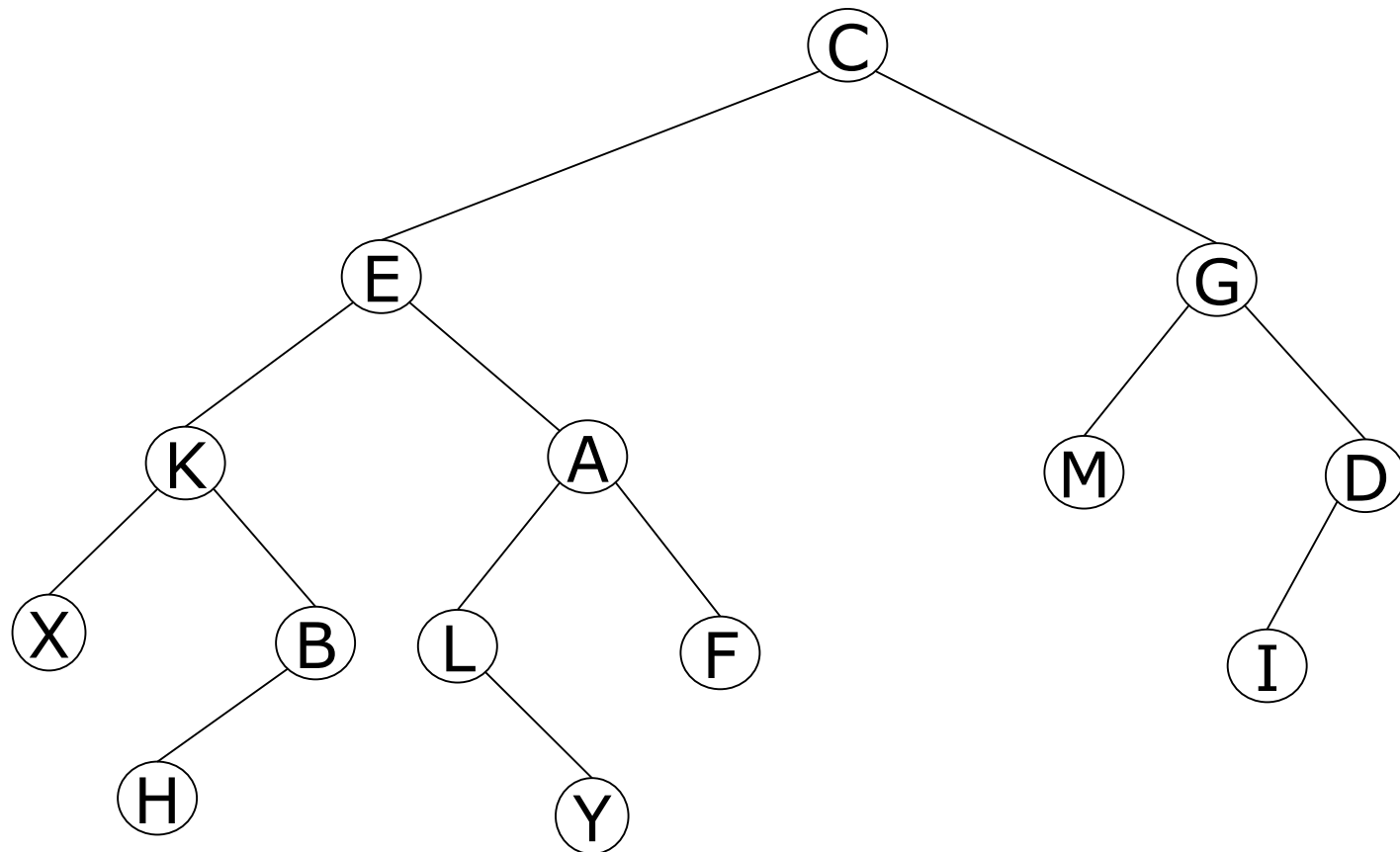
- intrare:

- un arbore binar **t**
 - adresa unui nod cu cel mult un fiu (tatăl noului nod)
 - tipul fiului adăugat (stânga, dreapta)
 - informația **e** din noul nod

- ieșire

- arborele la care s-a adăugat un nod ce memorează **e**; noul nod nu are fii

ArbBin: inserire



ArbBin: eliminare

- elimina()

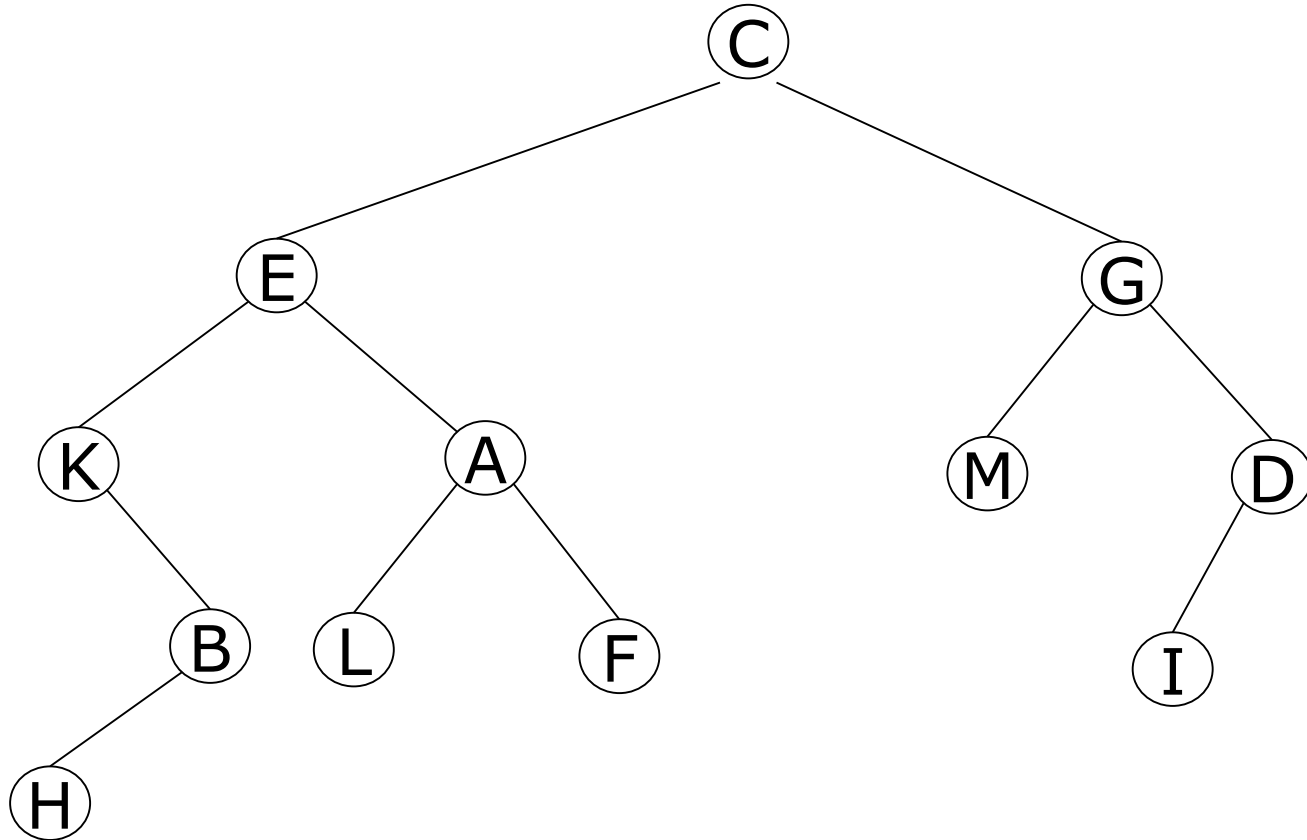
- intrare:

- un arbore binar **t**
 - adresa unui nod fără fii și adresa nodului-tată

- ieșire

- arborele din care s-a eliminat nodul dat (de pe frontieră)

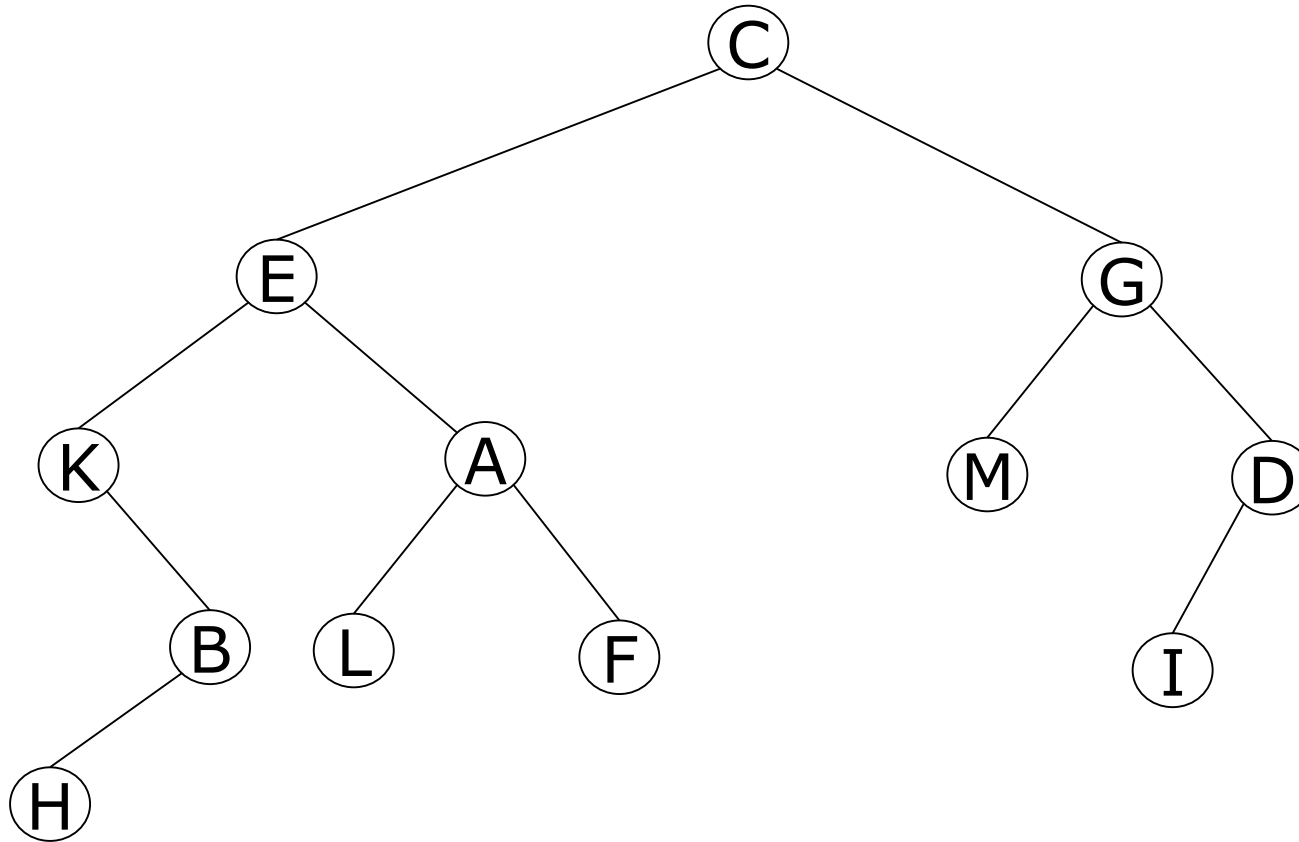
ArbBin: eliminare



ArbBin: parcurgere preordine

- `parcurePreordine()`
 - intrare
 - un arbore binar **t**
 - o procedură **viziteaza()**
 - ieşire
 - arborele binar **t** dar cu nodurile procesate cu **viziteaza()** în ordinea:
 - rădăcina (R)
 - subarborele stânga (S)
 - subarborele dreapta (D)

Parcurgere preordine - exemplu



C, E, K, B, H, A, L, F, G, M, D, I

ArbBin : parcurgere inordine

- `parcureInordine()`

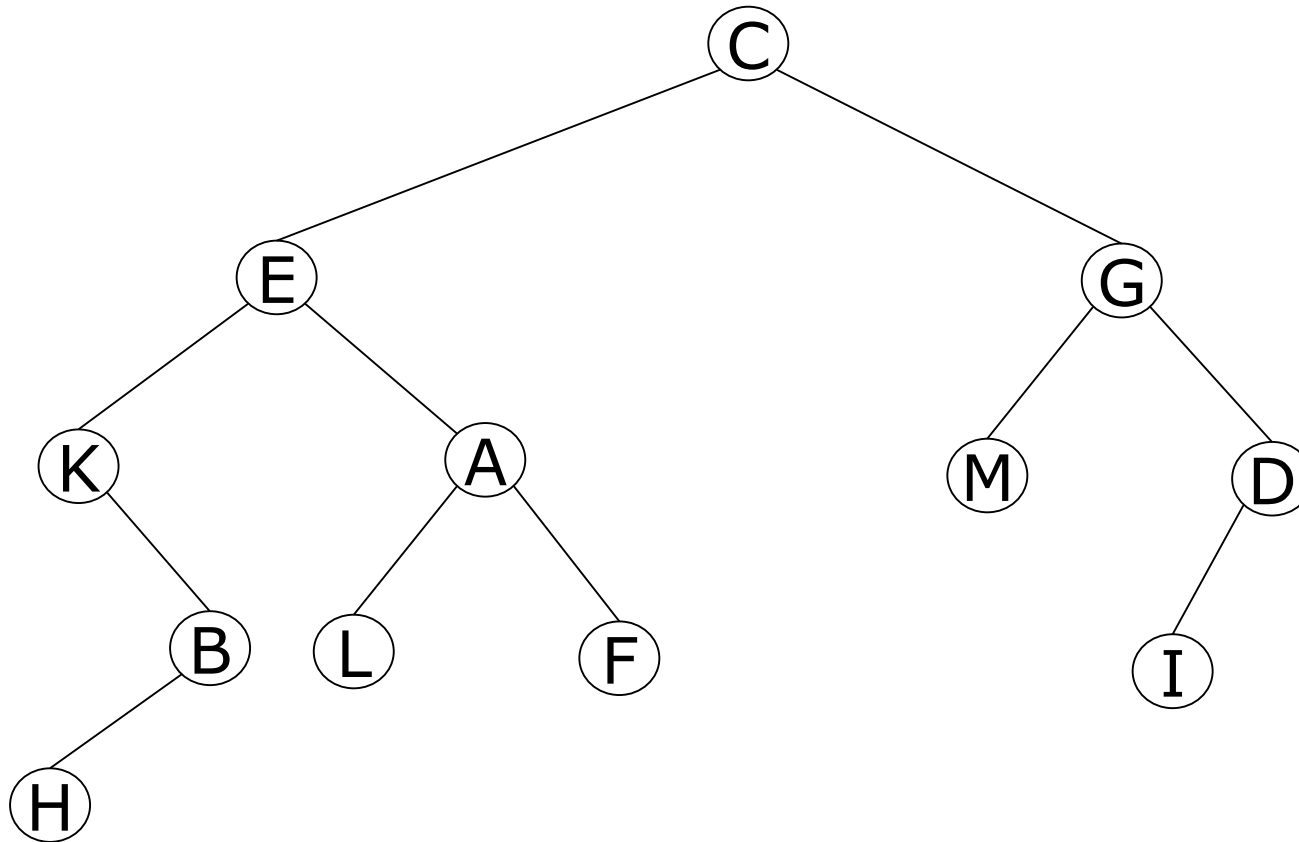
- intrare

- un arbore binar **t**
 - o procedură `viziteaza()`

- ieşire

- arborele binar **t** dar cu nodurile procesate cu `viziteaza()` în ordinea S R D

Parcurgere inordine - exemplu

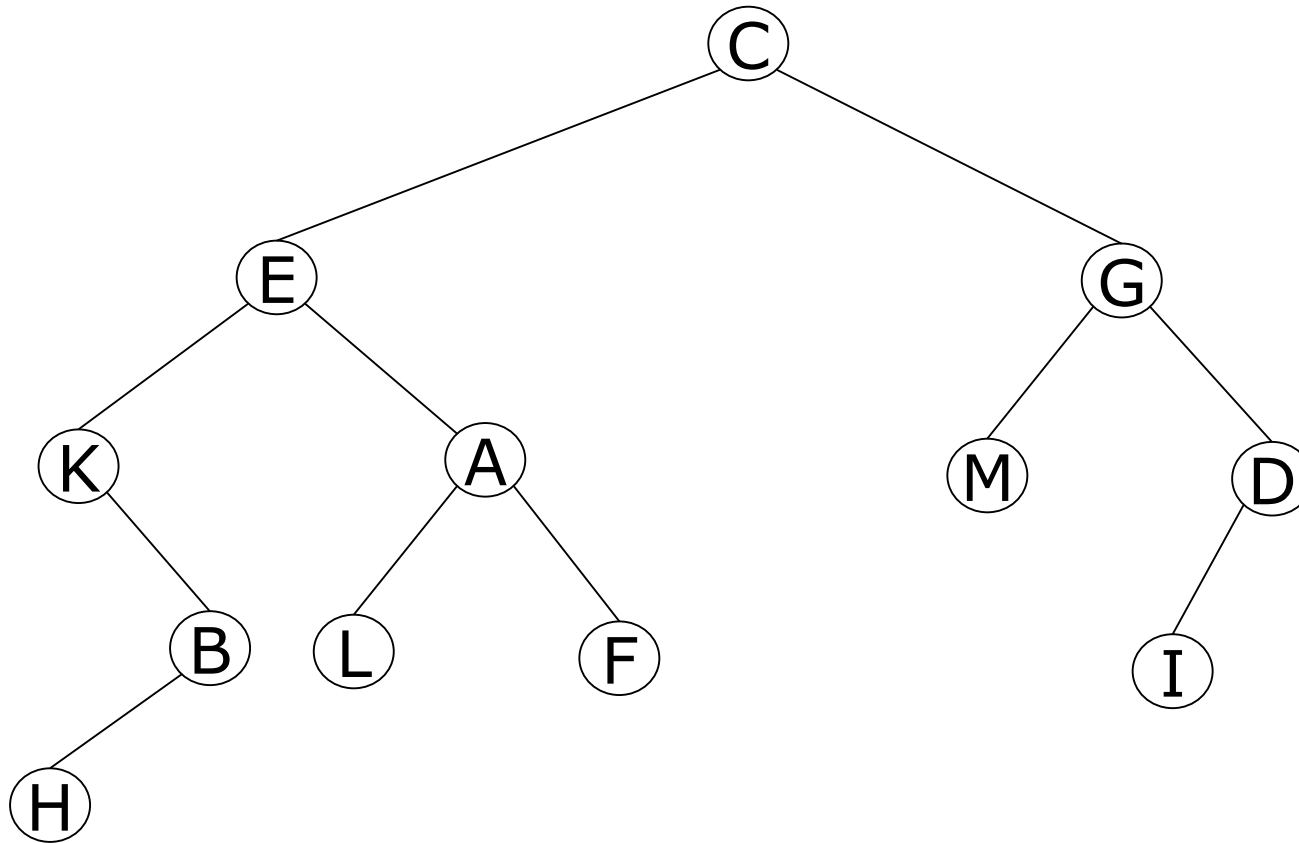


K, H, B, E, L, A, F, C, M, G, I, D

ArbBin: parcurgere postordine

- `parcurePostordine()`
 - intrare
 - un arbore binar **t**
 - o procedură `viziteaza()`
 - ieșire
 - arborele binar **t** dar cu nodurile procesate cu `viziteaza()` în ordinea S D R

Parcurgere postordine - exemplu

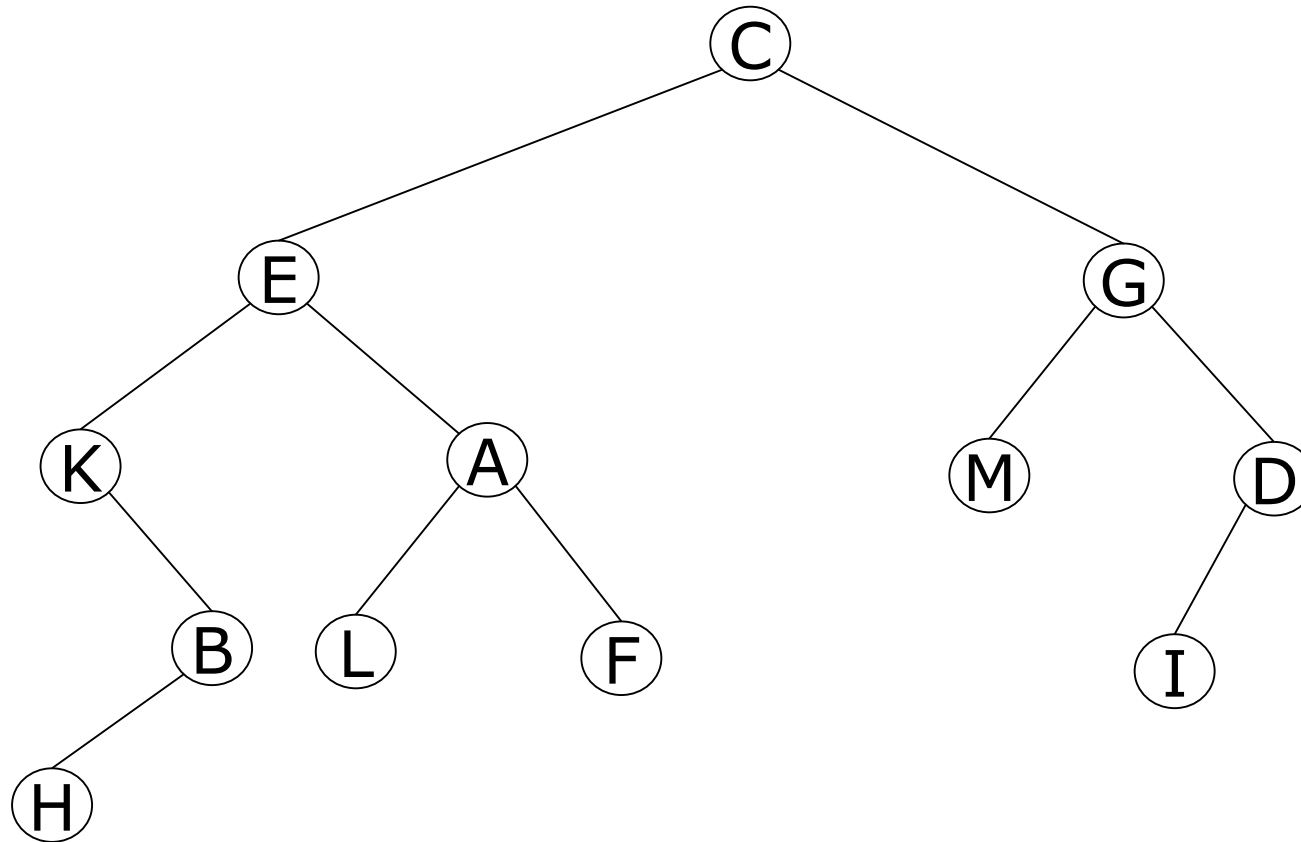


H, B, K, L, F, A, E, M, I, D, G, C

ArbBin: parcurgere BFS

- **parcureBFS()** - Breadth-first search
 - intrare
 - un arbore binar **t**
 - o procedură **viziteaza()**
 - ieşire
 - arborele binar **t** dar cu nodurile procesate cu **viziteaza()** în ordinea BFS (pe niveluri)

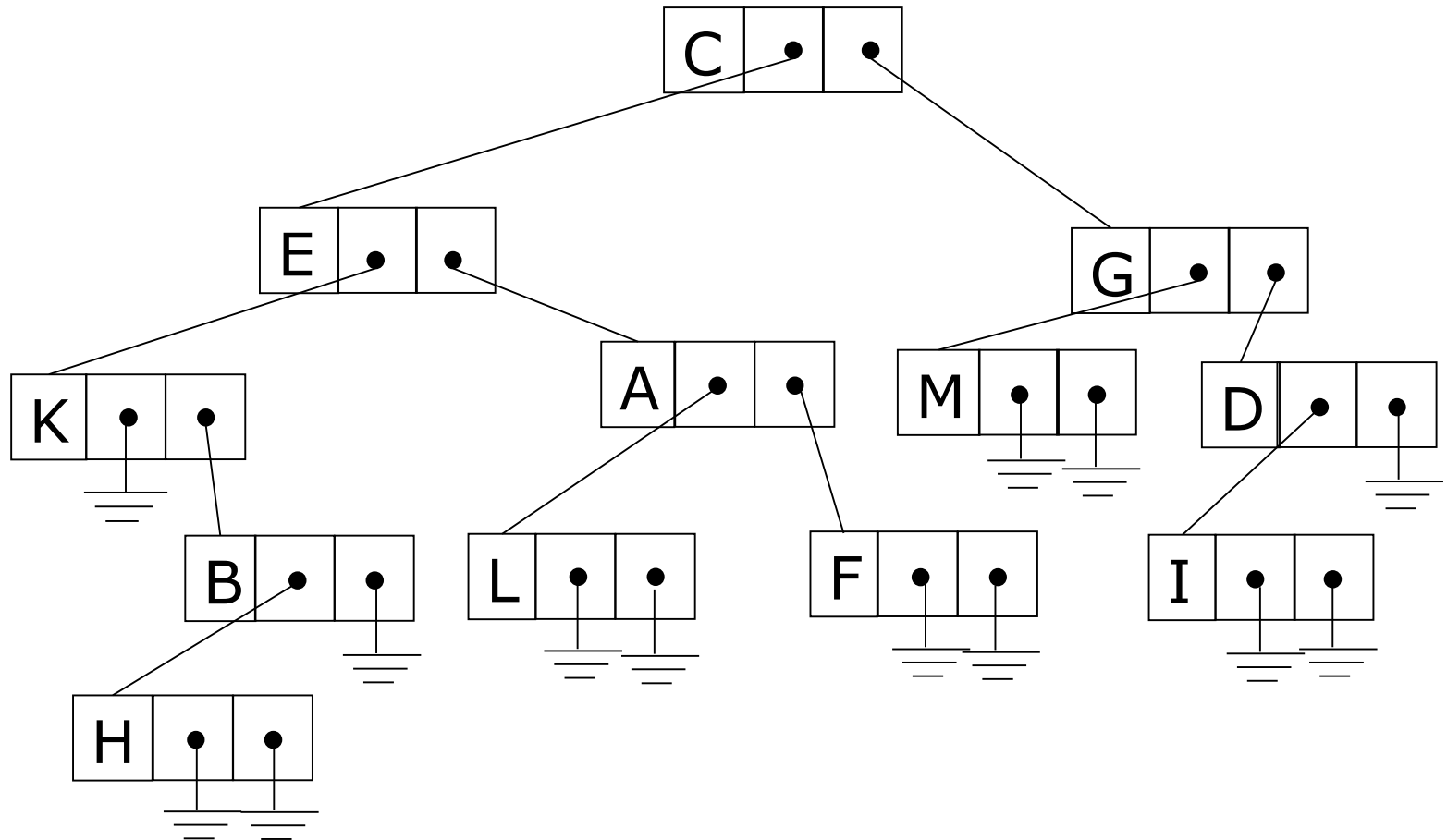
Parcurgere BFS - exemplu



C, E, G, K, A, M, D, B, L, F, I, H

ArbBin: implementare cu structuri înlanțuite

- reprezentarea obiectelor



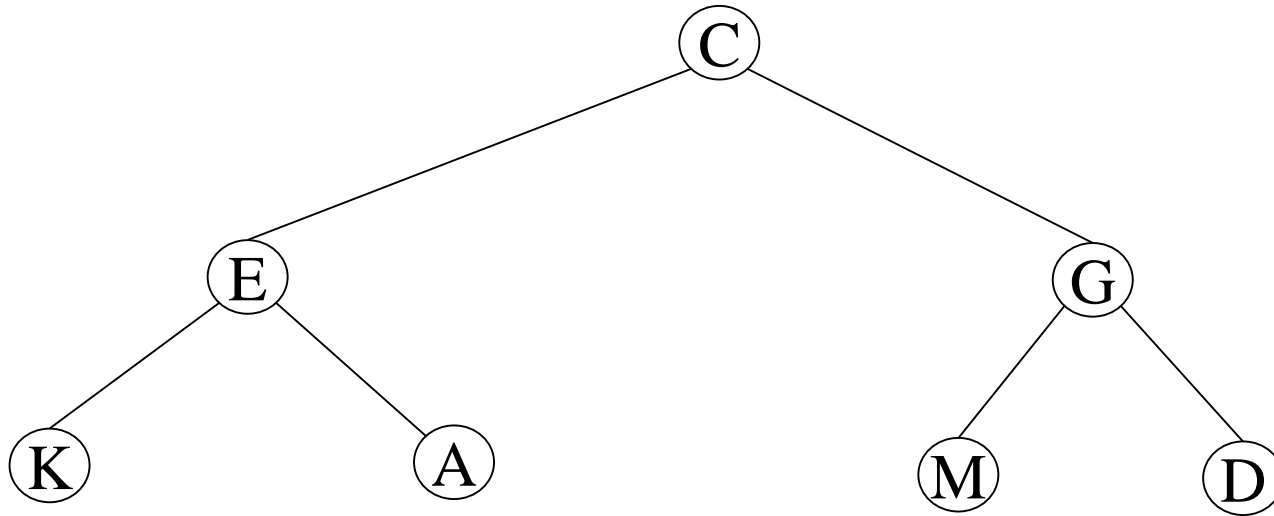
ArbBin: structura unui nod

- un nod **v** (aflat la adresa v) este o structură cu trei câmpuri:
 - **v->inf** /*informația memorată în nod*/
 - **v->stg** /*adresa fiului stânga*/
 - **v->drp** /*adresa fiului dreapta*/

ArbBin: parcurgePreordine()

```
procedure parcurgePreordine(v, viziteaza)
begin
    if (v == NULL) then return
    else
        viziteaza(v)
        parcurgePreordine(v->stg, viziteaza)
        parcurgePreordine(v->drp, viziteaza)
    end
```

Implementarea parcurgerii BFS



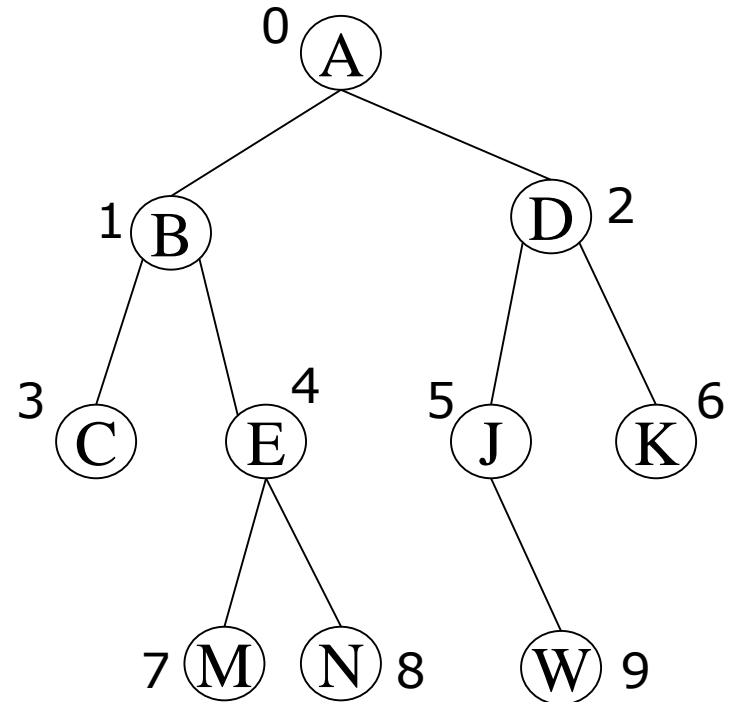
Coadă = (C E G K A M D)

Implementarea parcurgerii BFS

```
procedure parcurgeBFS(t, viziteza)
begin
    if (t == NULL) then return
    else
        Coadă ← coadaVida()
        insereaza(Coadă, t)
        while (not esteVida(Coadă))
            citește(Coadă, v)
            viziteaza(v)
            if (v->stg != NULL)
                then insereaza(Coadă, v->stg)
            if (v->drp != NULL)
                then insereaza(Coadă, v->drp)
            elimina(Coadă)
    end
```


ArbBin: implementarea cu liste

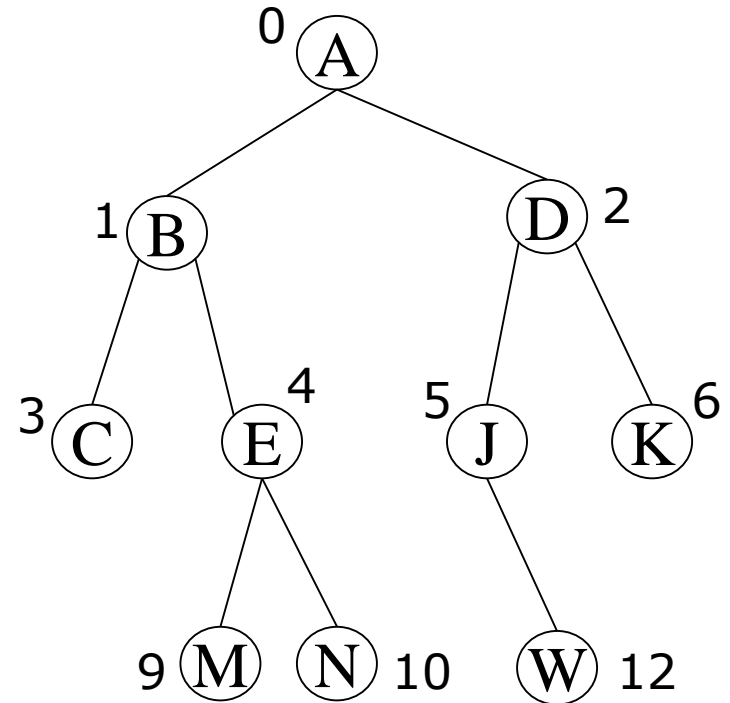
- Reprezentarea relației “părinte”: tablou de părinți
- **Avantaje:**
 - Simplitate;
 - Acces ușor de la un nod spre rădăcină;
 - Economie de memorie.
- **Inconveniente:**
 - Acces dificil de la rădăcină spre noduri.



-1	0	0	1	1	2	2	4	4	5
0	1	2	3	4	5	6	7	8	9

ArbBin: implementare cu tablouri

- Nodurile sunt memorate într-un tablou.
- Indexul unui nod este
 - $\text{index}(\text{rădăcină}) = 0$
 - $\text{index}(x) = 2 * \text{index}(\text{părinte}(x)) + 1$,
dacă x este fiu stâng
 - $\text{index}(x) = 2 * \text{index}(\text{părinte}(x)) + 2$,
dacă x este fiu drept



A	B	D	C	E	J	K			M	N		w		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Aplicație: expresii întregi

- Expresii întregi
 - definiție;
 - exemple.
- Reprezentarea expresiilor ca arbori:
 - similarități între cele două definiții;
 - arborele asociat unei expresii;
 - notațiile prefixate, infixate și postfixate și parcurgeri ale arborilor.

Definiția expresiilor întregi

$\langle \text{int} \rangle ::= \dots -2 \mid -1 \mid 0 \mid 1 \mid 2 \dots$

$\langle \text{op_bin} \rangle ::= + \mid - \mid * \mid / \mid \%$

$\langle \text{expr_int} \rangle ::= \langle \text{int} \rangle$

$\mid \langle \text{exp_int} \rangle \langle \text{op_bin} \rangle \langle \text{exp_int} \rangle$

$\mid (\langle \text{exp_int} \rangle)$

- reguli de precedență

$12-5*2$ este $(12-5)*2$ sau $12-(5*2)?$

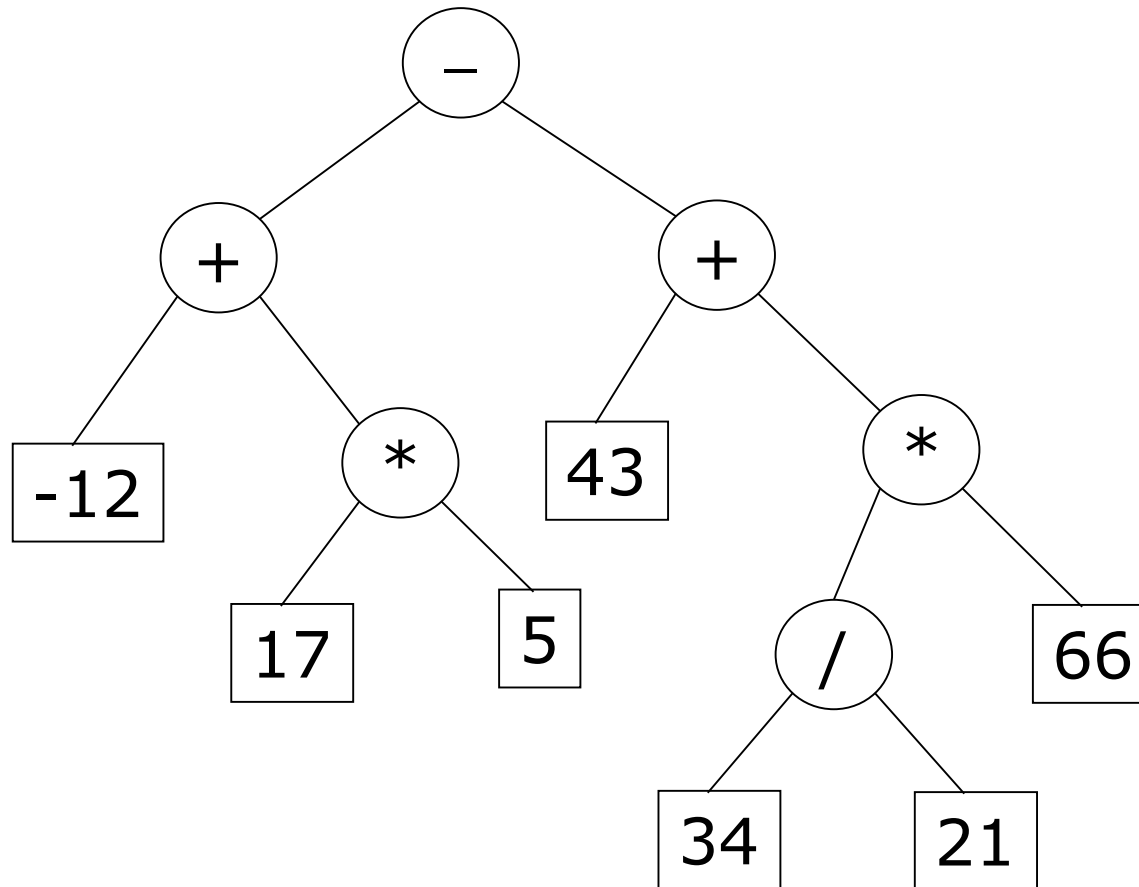
- reguli de asociere

$15/4/2$ este $(15/4)/2$ sau $15/(4/2)?$

$15/4*2$ este $(15/4)*2$ sau $15/(4*2)?$

Expresiile reprezentate ca arbori

-12 + 17 * 5 - (43 + 34 / 21 * 66)



Notațiile postfixate și prefixate

- notația postfixată se obține prin parcurge postordine
-12, 17, 5, *, +, 43, 34, 21, /, 66, *, +, -
- notația prefixată se obține prin parcurge preordine
-, +, -12, *, 17, 5, +, 43, *, /, 34, 21, 66

