Arbori binari

SD 2020/2021

Conținut

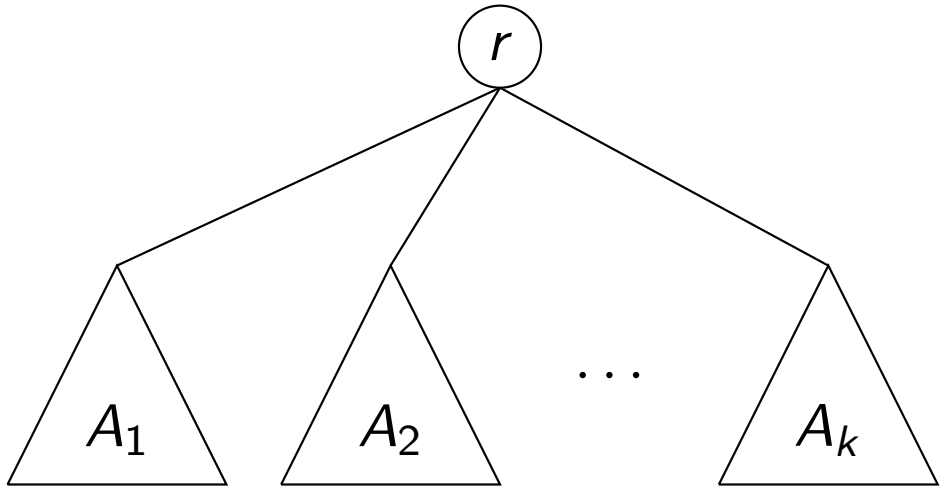
Arbori: definiție recursivă

$$A = \begin{cases} \Lambda - \text{arborele vid}, \\ (r, \{A_1, \cdots, A_k\}), \quad r \text{ element și } A_1, \cdots, A_k \text{ arbori.} \end{cases}$$

Arbori: definiție recursivă

$$A = \begin{cases} \Lambda - \text{arborele vid}, \\ (r, \{A_1, \cdots, A_k\}), \quad r \text{ element și } A_1, \cdots, A_k \text{ arbori.} \end{cases}$$

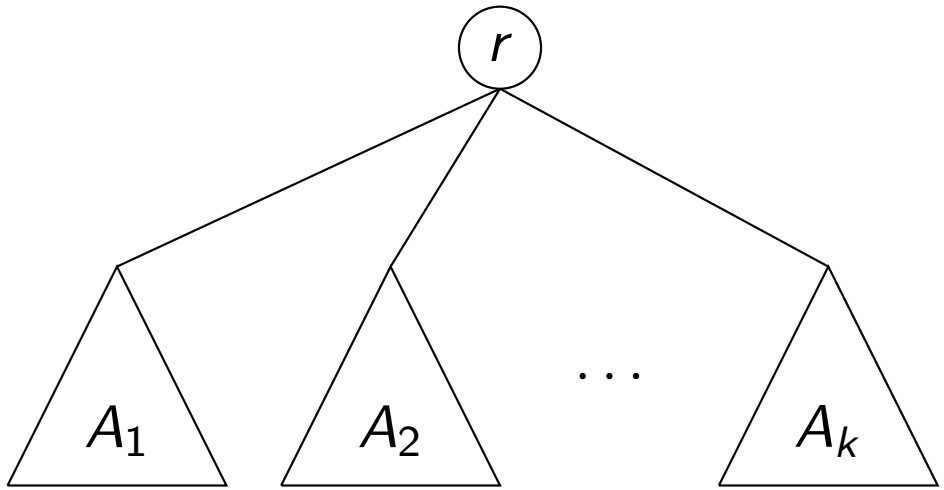
$A = \Lambda$ sau



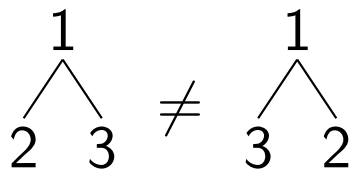
Arbori: definiție recursivă

$$A = \begin{cases} \Lambda - \text{arborele vid}, \\ (r, \{A_1, \cdots, A_k\}), \quad r \text{ element și } A_1, \cdots, A_k \text{ arbori.} \end{cases}$$

$A = \Lambda$ sau

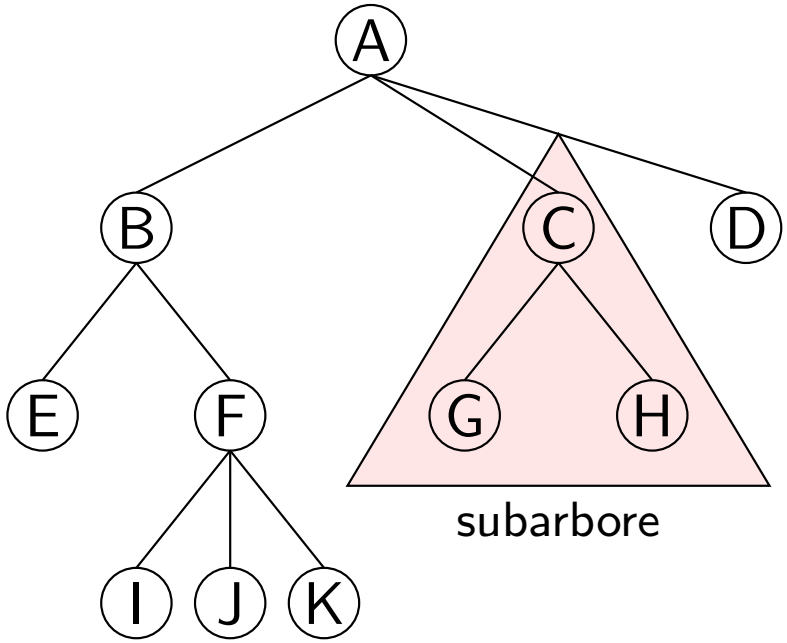


Dacă A este *ordonat (planar)*, atunci



Arbori: terminologie

- rădacina: nodul fără părinte.
- nod intern: nod cu cel puțin un fiu.
- nod extern (frunză): nod fără fii.
- descendenții unui nod: fii, nepoți, etc.
- frați: toate celelalte noduri având același părinte.
- subarbore: arborele format dintr-un nod și descendenții săi.

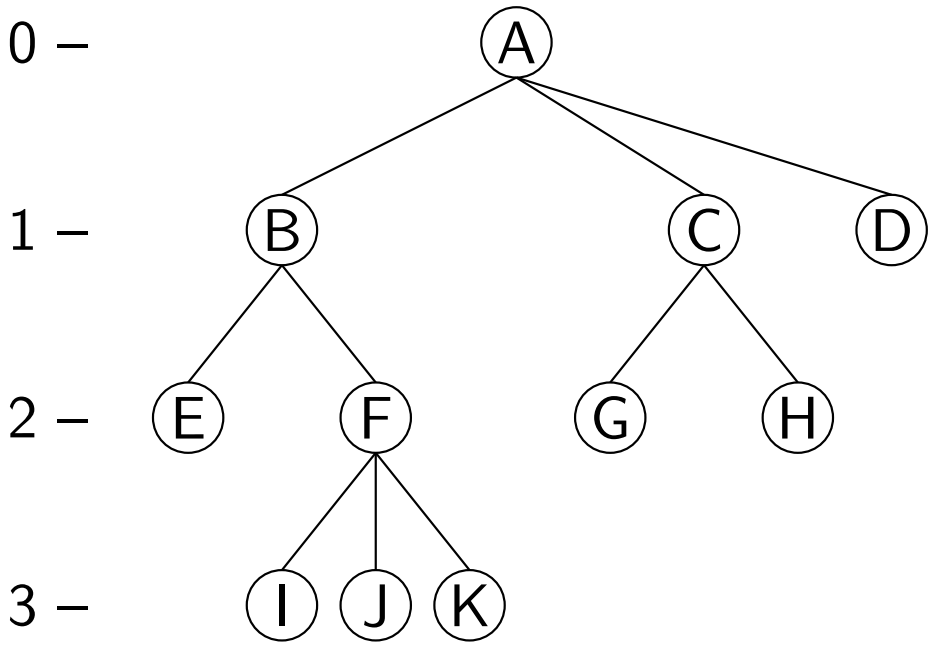


Arbori: terminologie

- adâncimea uni nod x:

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

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



Conținut

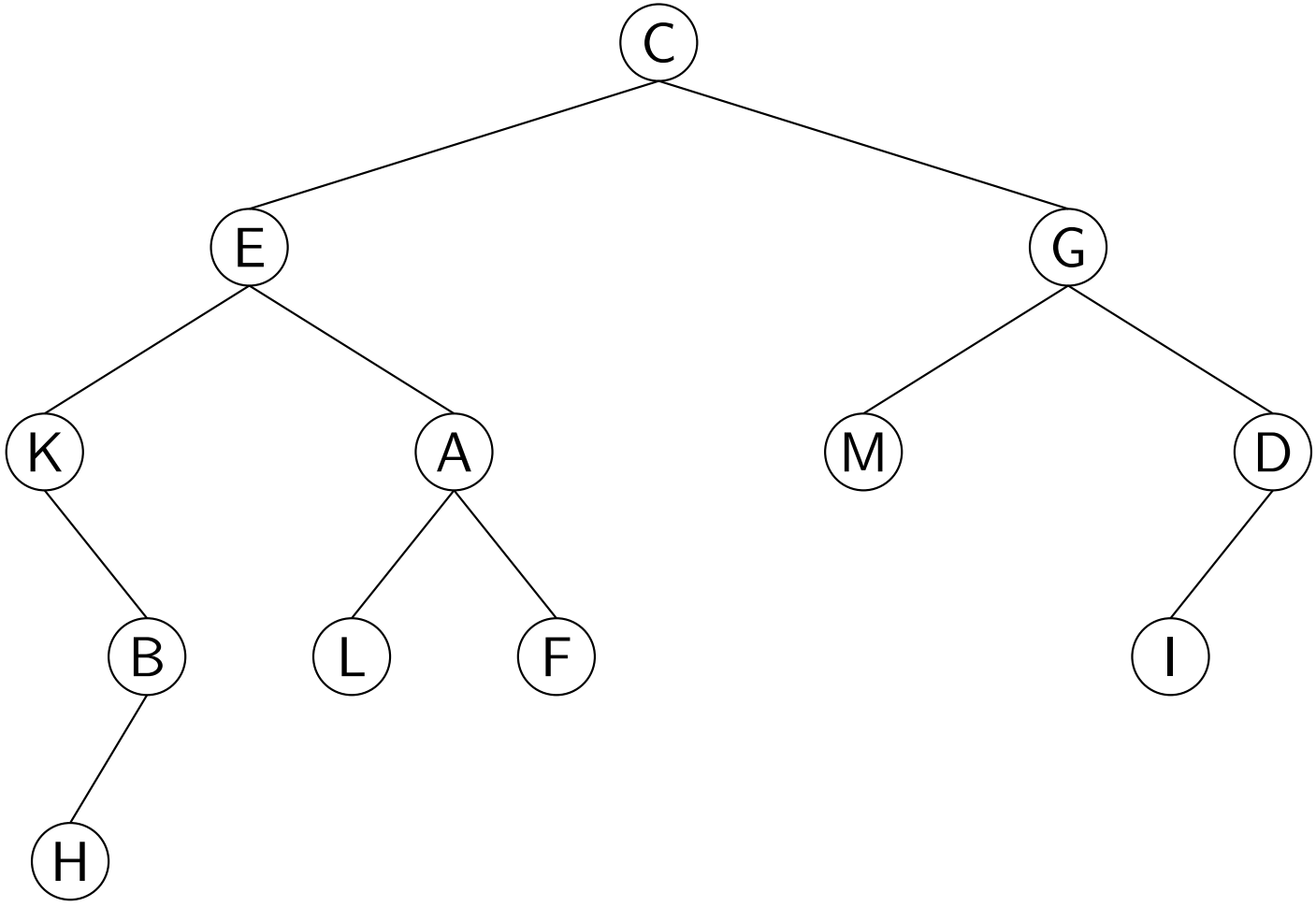
Tipul abstract ArbBin

OBIECTE: arbori binari.

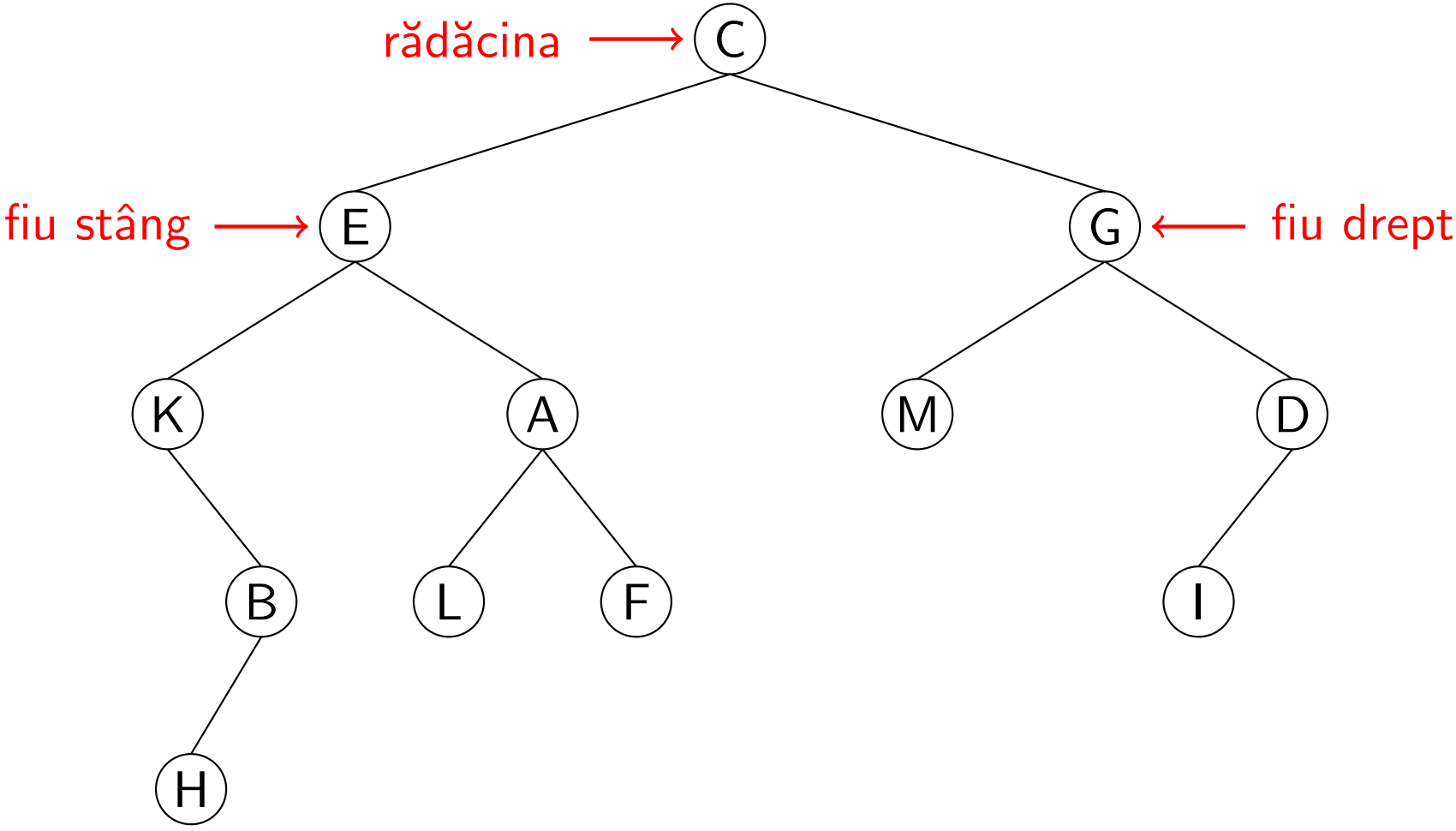
Un arbore binar este o colecție de noduri cu proprietățile:

- orice nod are 0, 1 sau 2 succesori (fii, copii).
- orice nod, exceptând unul singur — rădăcina — are un singur nod predecesor (tată, părinte).
- rădăcina nu are predecesori.
- fii sunt ordonați: fiul stâng, fiul drept. Dacă un nod are un singur fiu, atunci trebuie menționat care.
- nodurile fără fii formează frontiera arborelui.

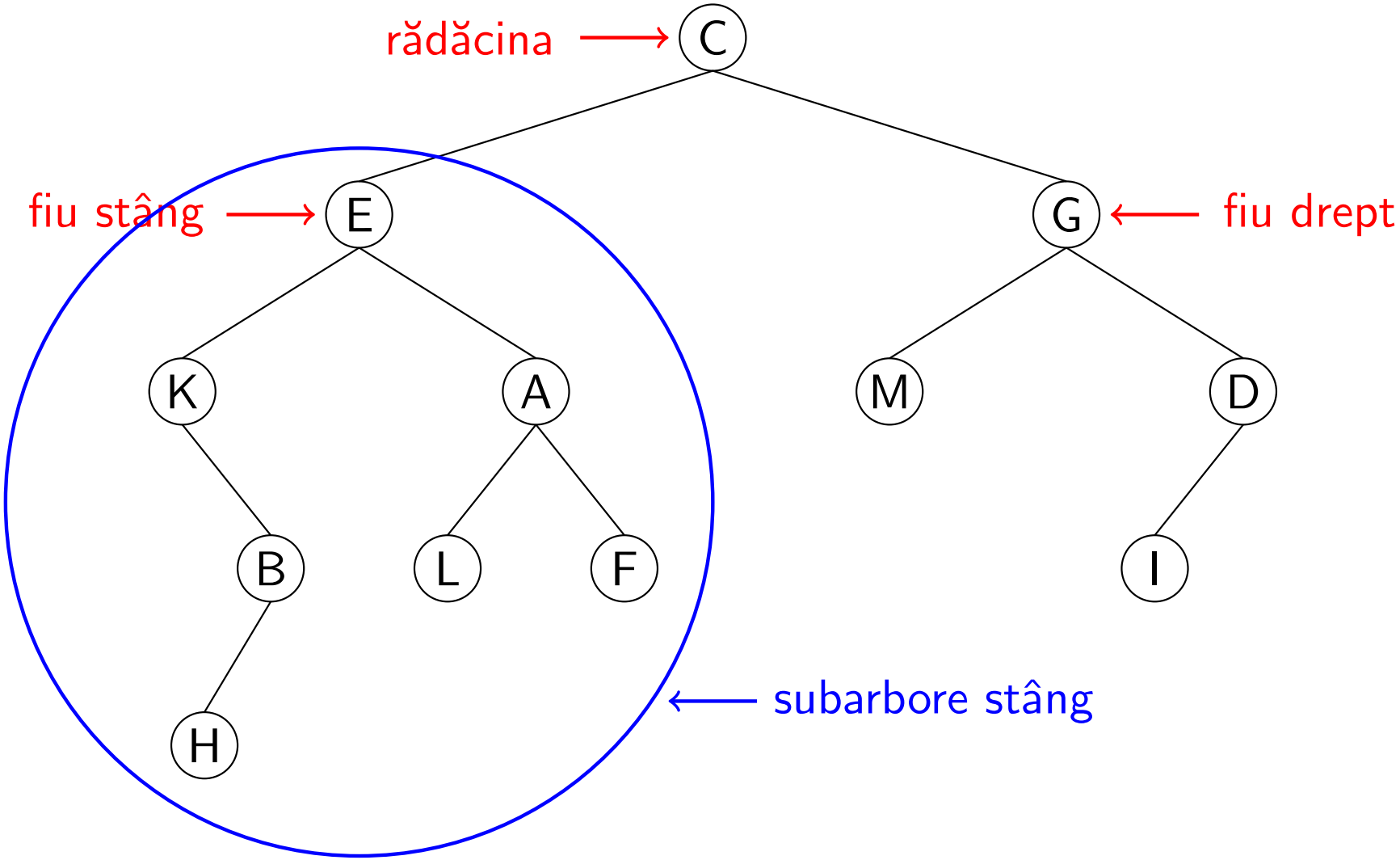
Arbori binari: exemplu



Arbori binari: exemplu

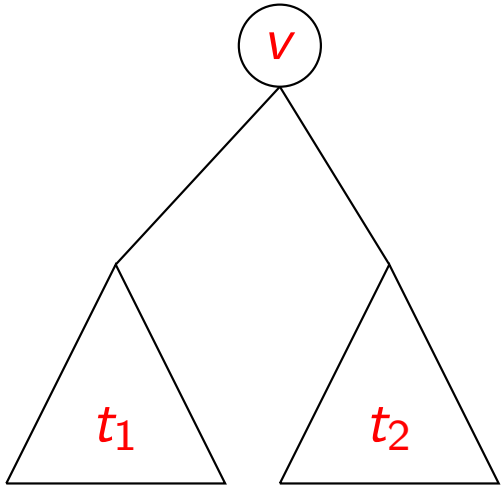


Arbori binari: exemplu



Arbori binari: definiție recursivă

- Arborele vid (fără nici un nod) este arbore binar.
- Daca v este un nod și t_1 și t_2 sunt arbori binari, atunci arborele care are pe v ca rădăcină, t_1 subarbore stâng al rădăcinii și t_2 subarbore drept al rădăcinii este arbore binar.



Arbori binari: proprietăți

Notații:

- n – numărul de noduri din arbore.
- n_e – numărul de noduri externe.
- n_i – numărul de noduri interne.
- h – înălțimea arborelui.

$$h + 1 \leq n \leq 2^{h+1} - 1; \qquad \log_2(n+1) - 1 \leq h \leq n - 1$$

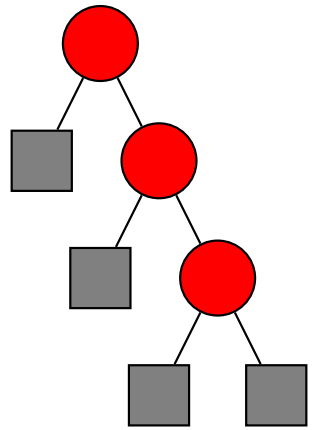
$$1 \leq n_e \leq 2^h; \qquad h \leq n_i \leq 2^h - 1$$

Arbori binari: proprietăți

- Arbore propriu: fiecare nod intern are exact doi fii.

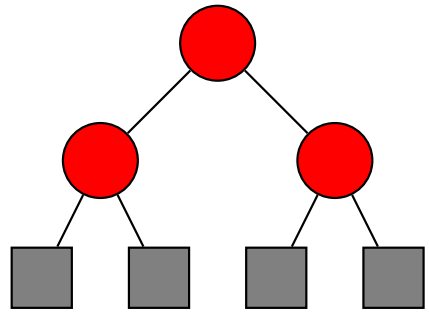
$2h + 1 \leq n \leq 2^{h+1} - 1;$
$\log_2(n+1) - 1 \leq h \leq (n-1)/2$

$h + 1 \leq n_e \leq 2^h;$
$h \leq n_i \leq 2^h - 1$
$n_e = n_i + 1$



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

nivelul i are 2^i noduri;
$n = 2^{h+1} - 1 = 2n_e - 1$



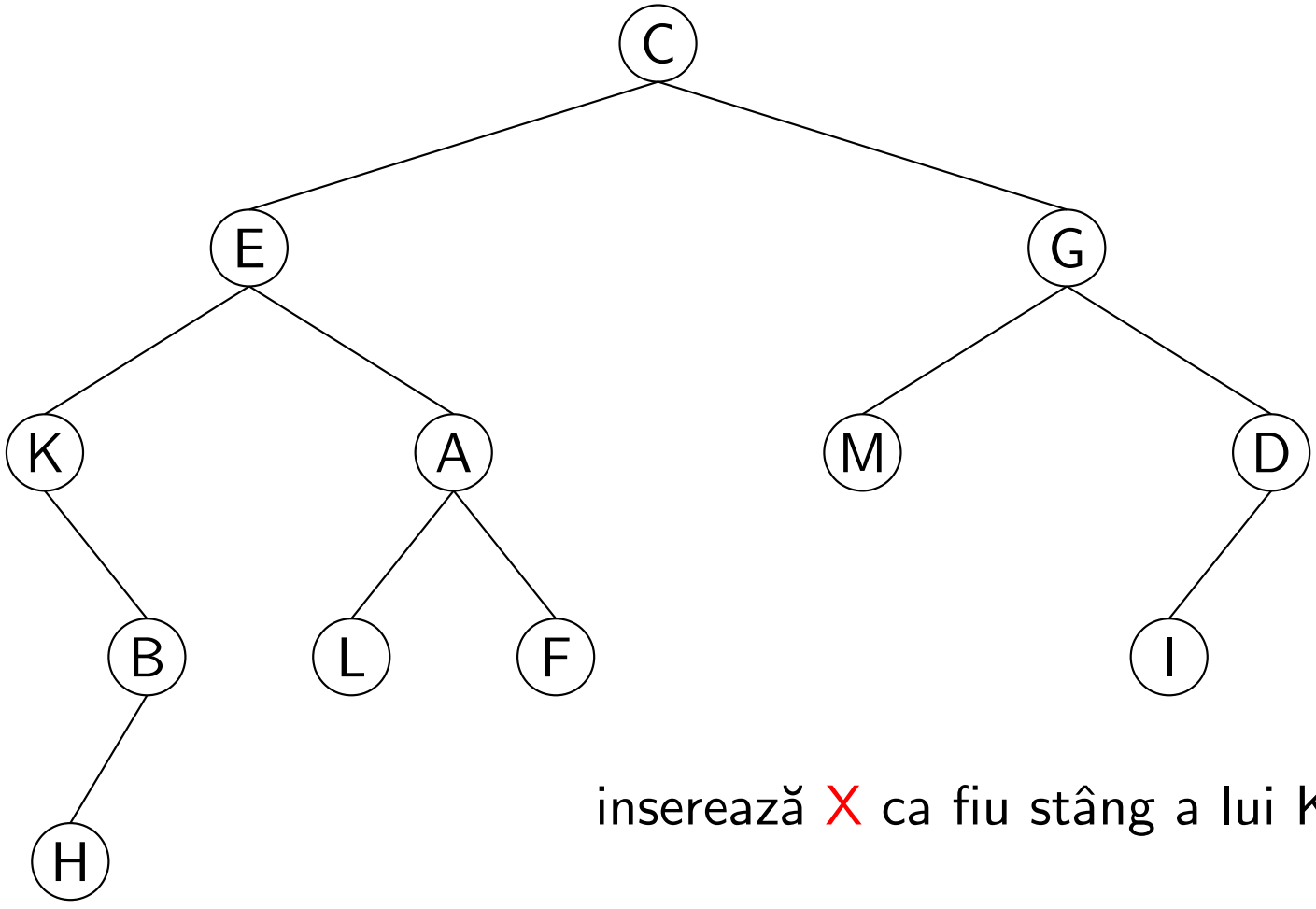
ArbBin – operații

`insereaza()`

- intrare:
  - un arbore binar `t`;
  - adresa unui nod cu cel mult un fiu (tatăl noului nod);
  - tipul fiului adăgat (stânga, dreapta);
  - informația `e` din noul nod.

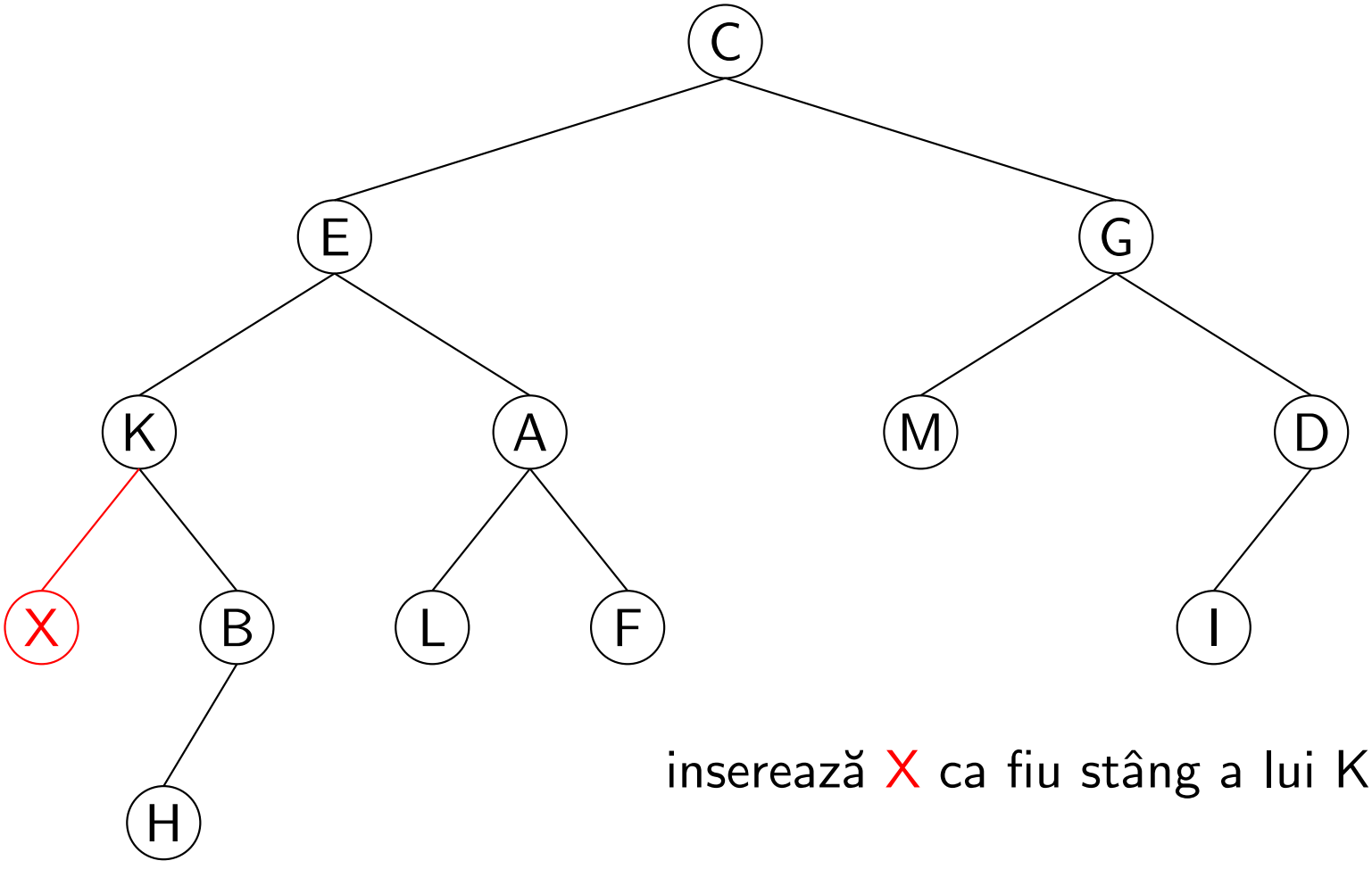
- ieșire:
  - arborele `t` la care s-a adăugat un nod ce memorează `e`; noul nod nu are fii.

ArbBin: inserare - exemplu



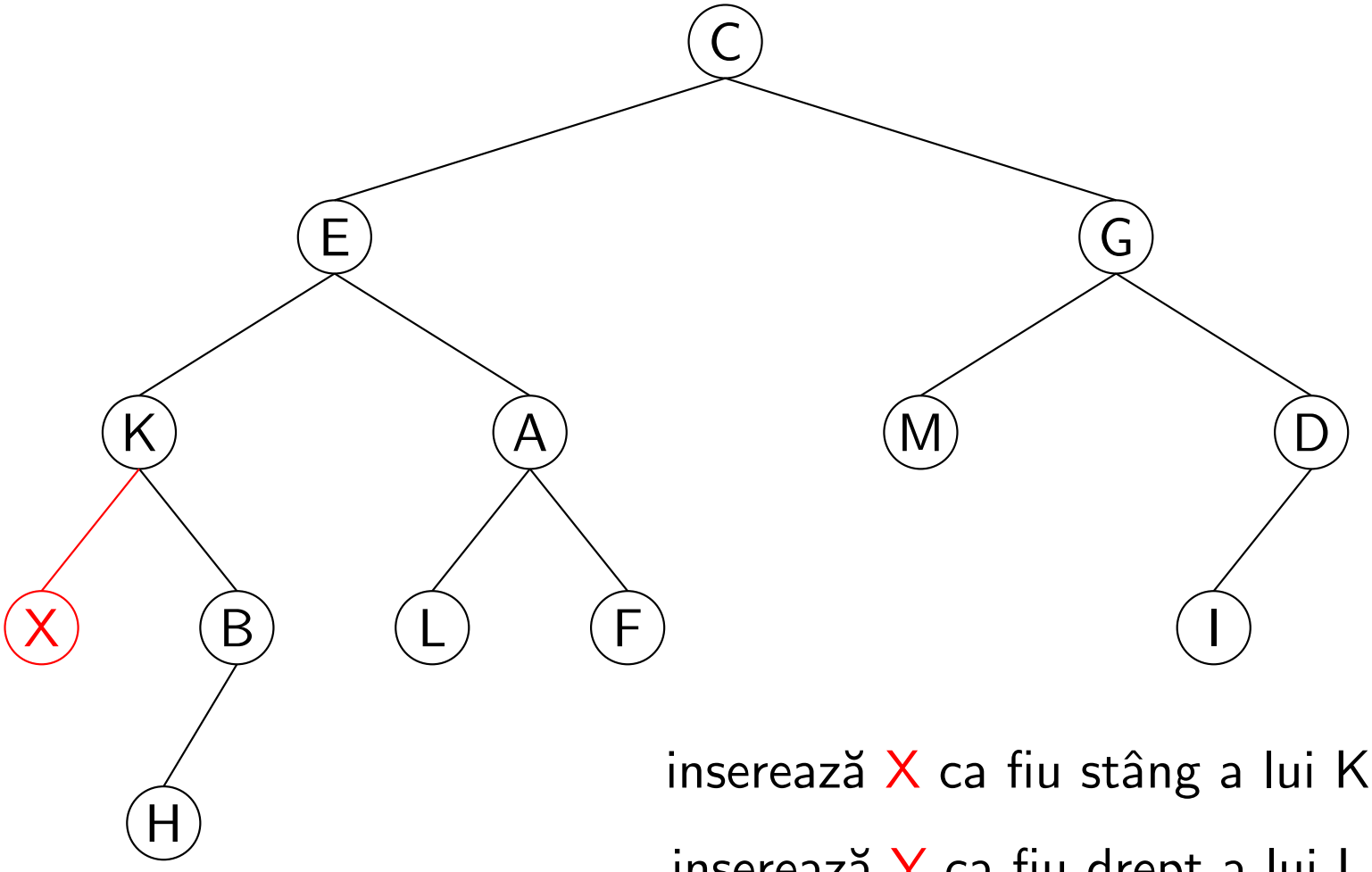
inserează X ca fiu stâng a lui K

ArbBin: inserare - exemplu



inserează X ca fiu stâng a lui K

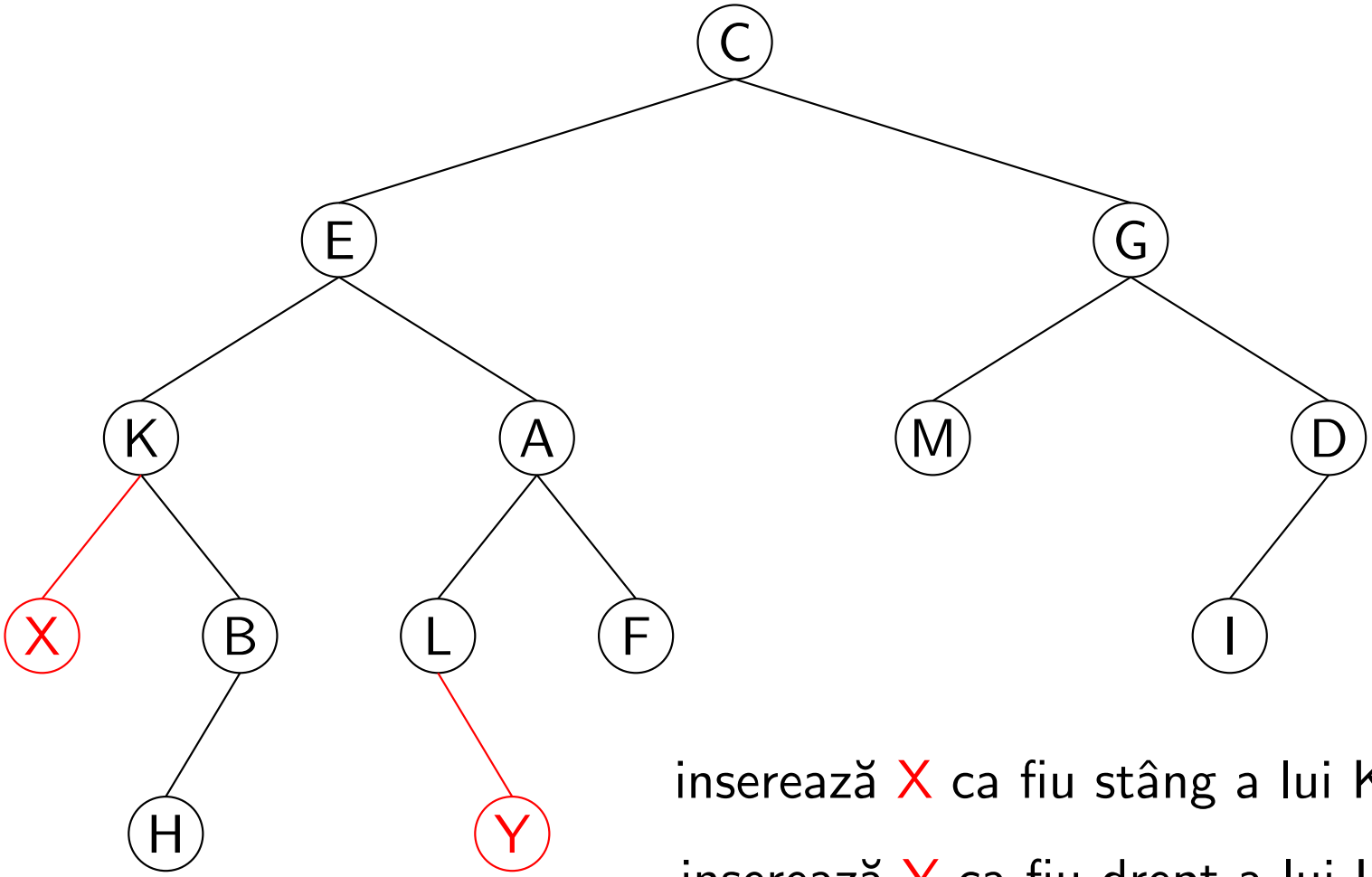
ArbBin: inserare - exemplu



inserează X ca fiu stâng a lui K

inserează Y ca fiu drept a lui L

ArbBin: inserare - exemplu



inserează X ca fiu stâng a lui K

inserează Y ca fiu drept a lui L

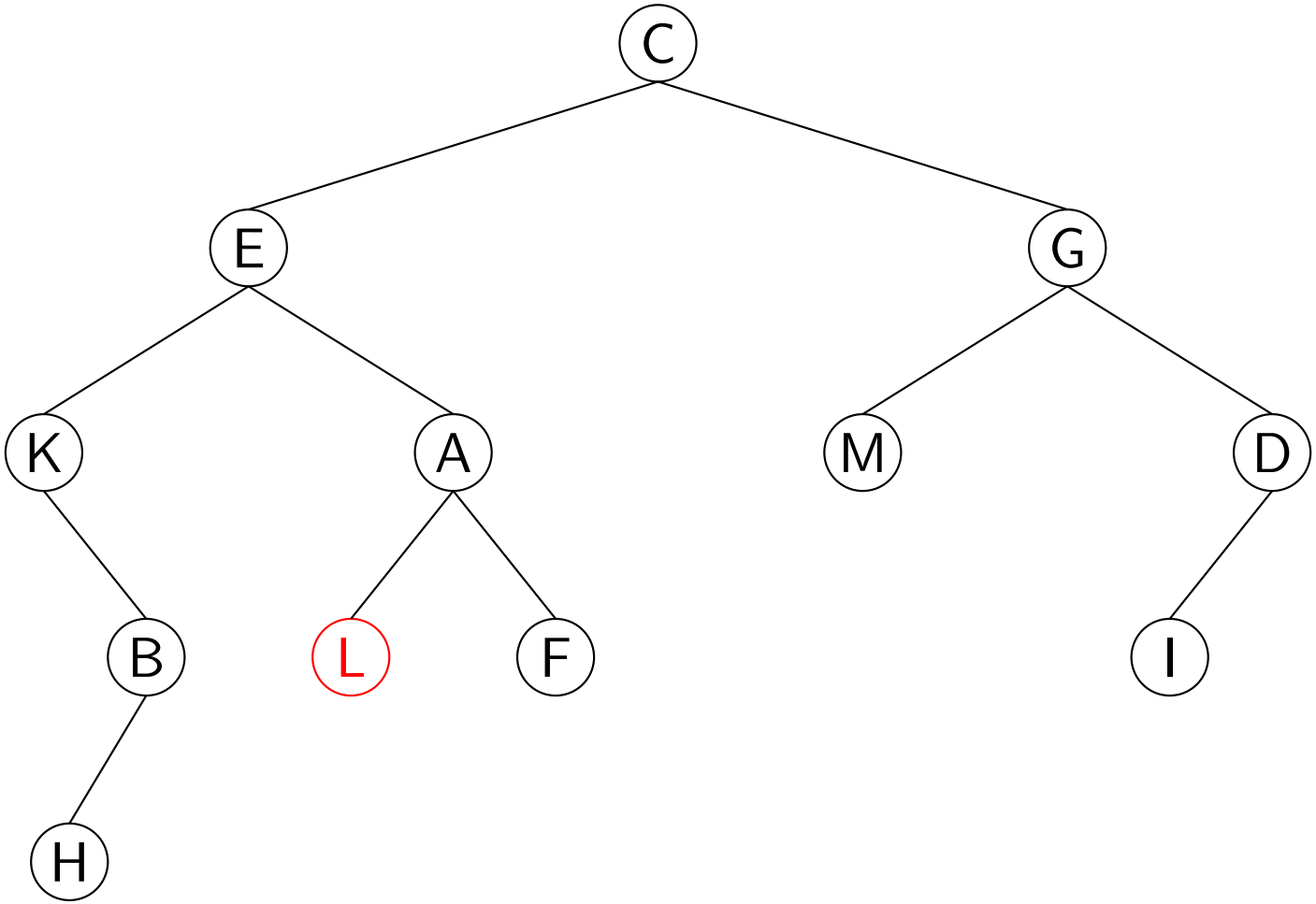
ArbBin – operații

`elimina()`

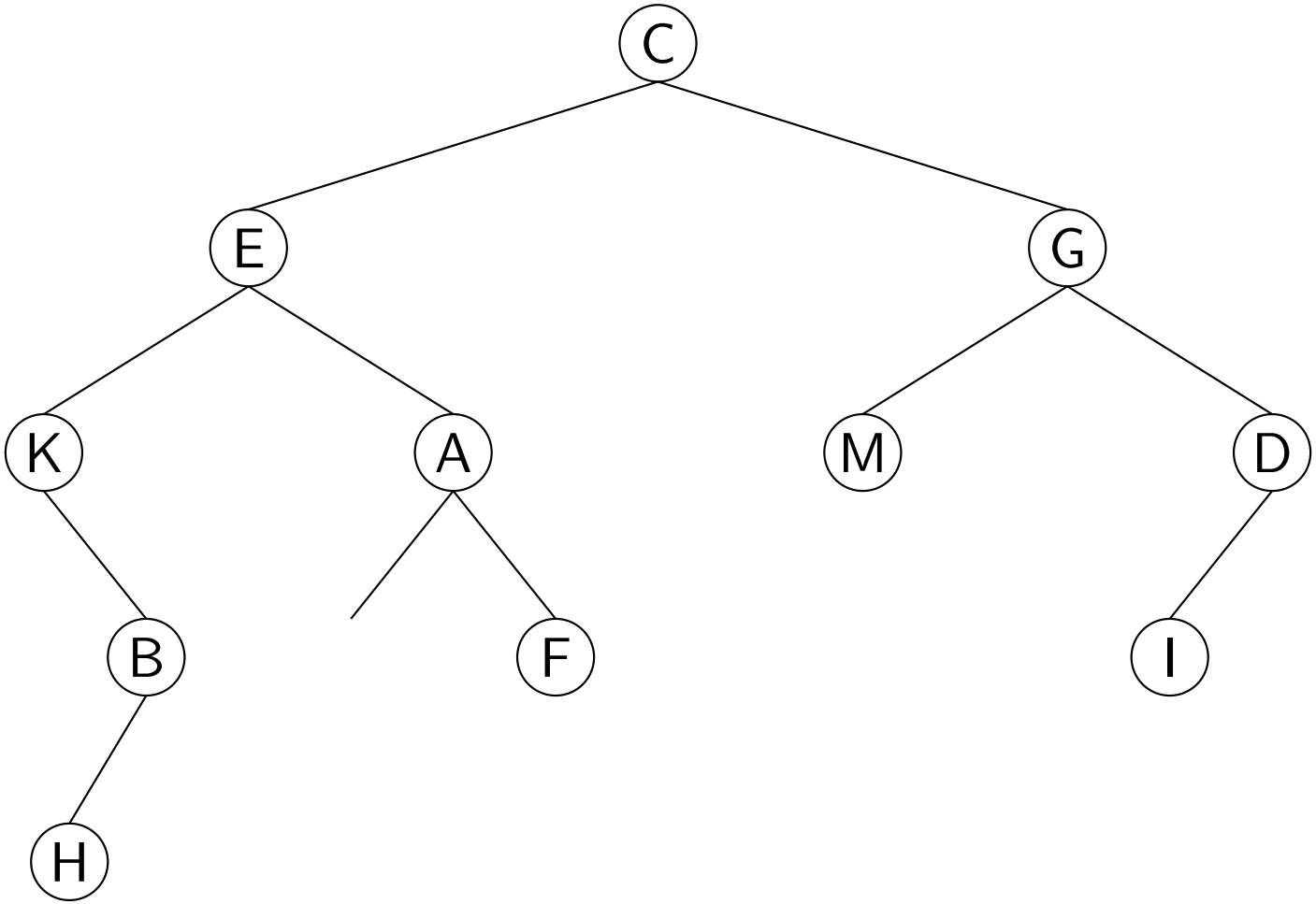
- intrare:
  - un arbore binar `t`;
  - adresa unui nod fără fii și adresa nodului părinte.

- ieșire:
  - arborele `t` din care s-a eliminat nodul dat (de pe frontieră).

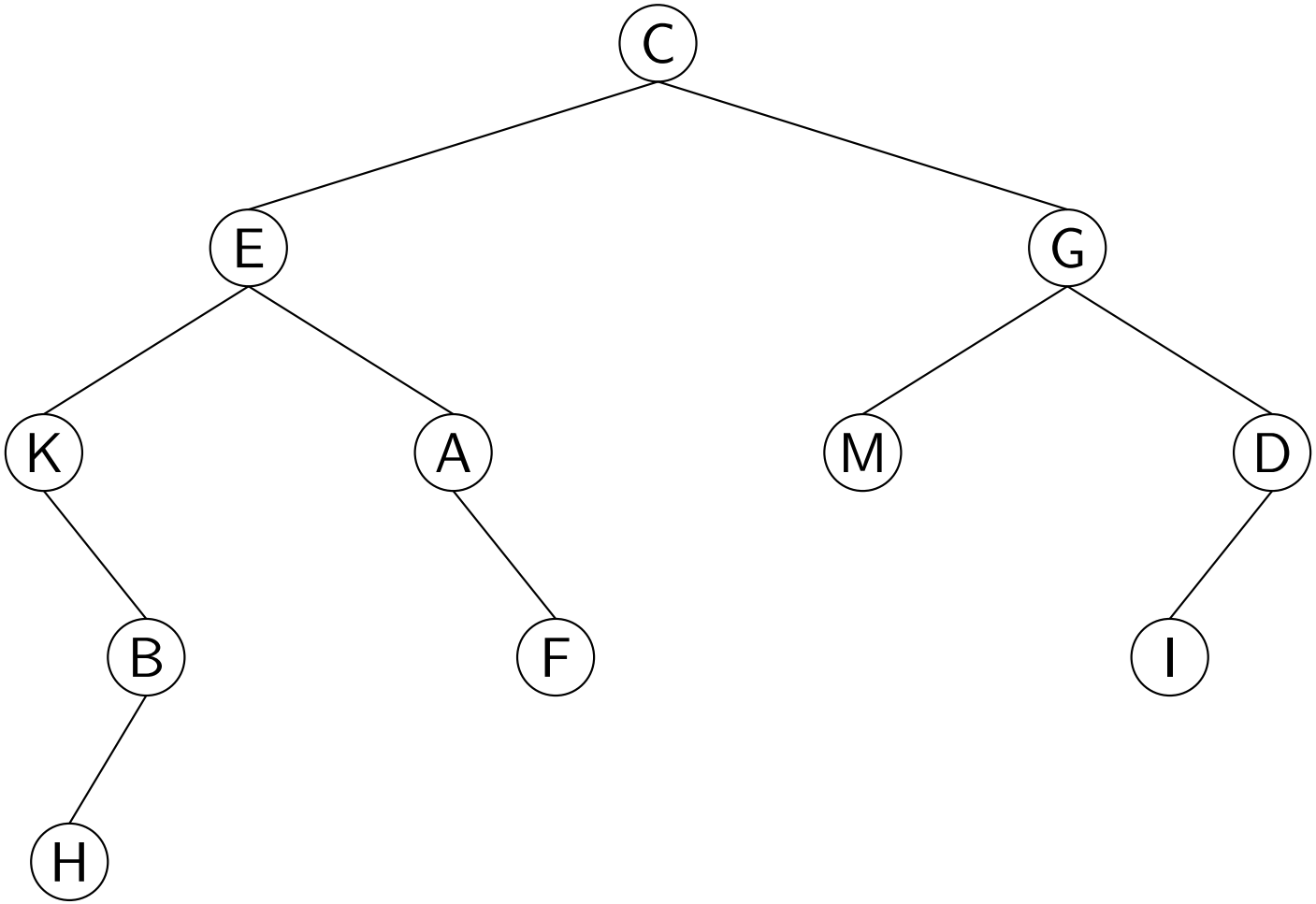
ArbBin: eliminare - exemplu



ArbBin: eliminare - exemplu



ArbBin: eliminare - exemplu



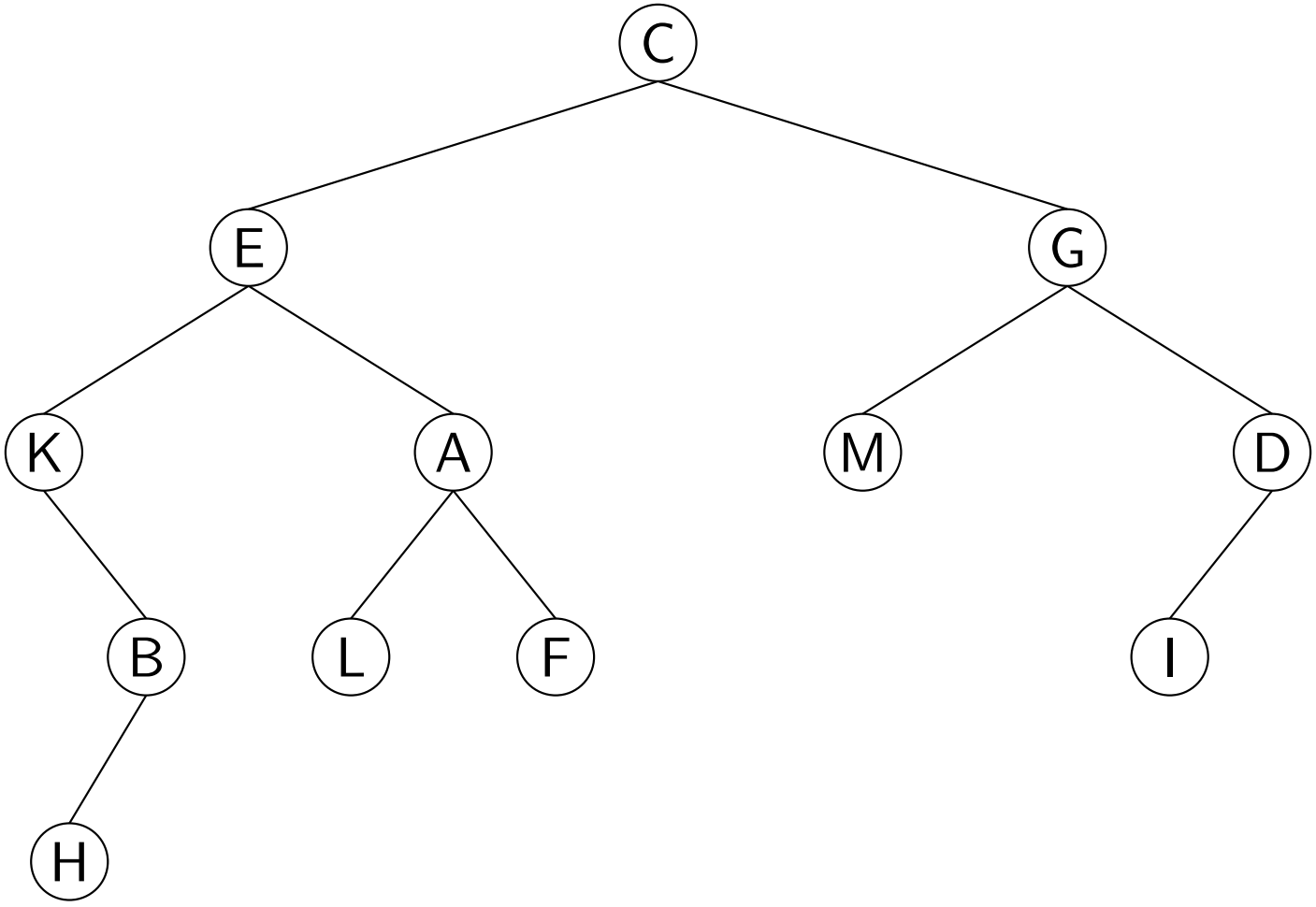
ArbBin – parcurgerea preordine

`parcurgerePreordine()`

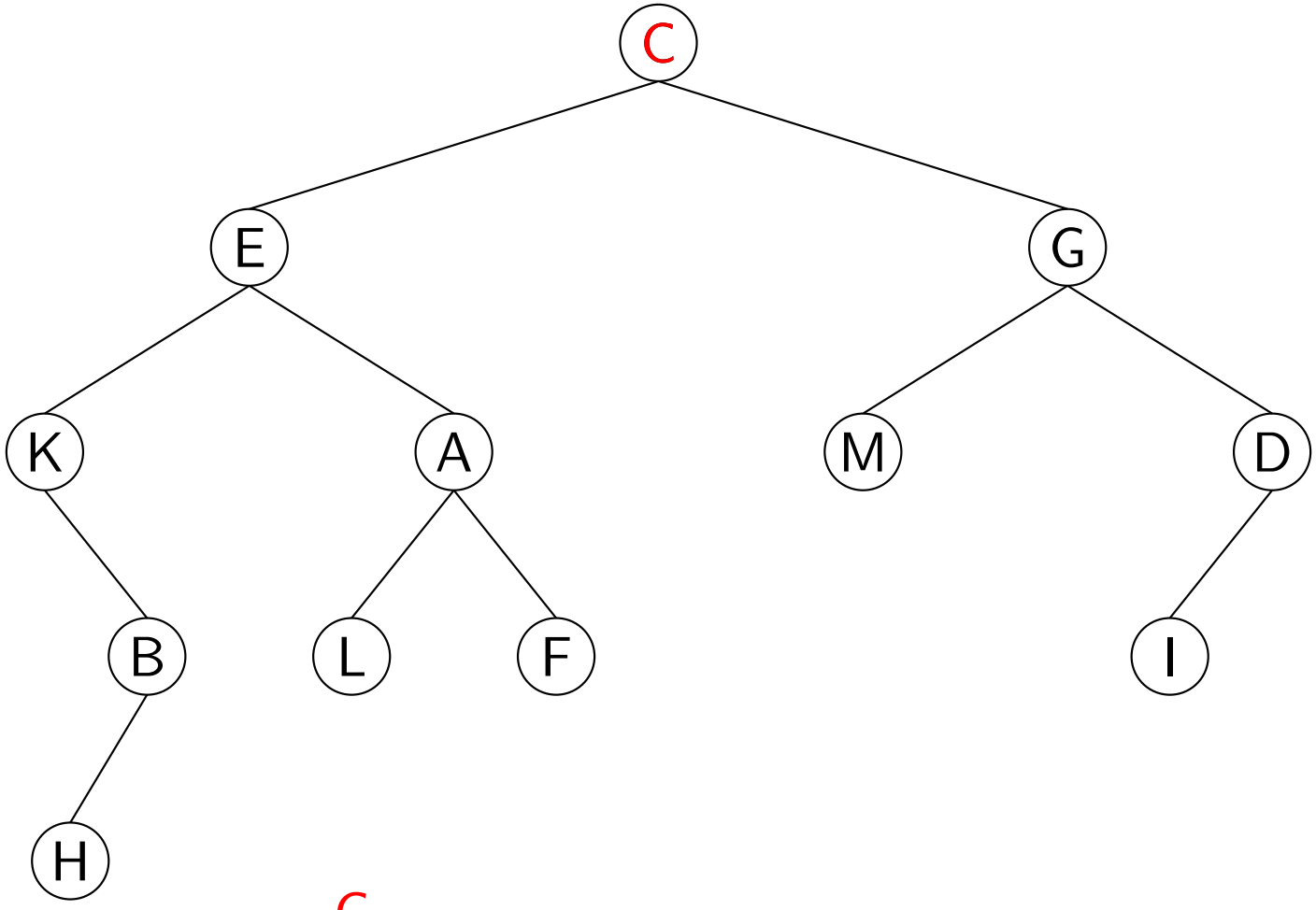
- intrare:
  - un arbore binar `t`;
  - o procedură `viziteaza()`.

- ieșire:
  - arborele `t`, dar cu nodurile procesate cu `viziteaza()` în ordinea
    * (R) – rădăcina
    * (S) – subarborele stânga
    * (D) – subarborele dreapta

Parcurgere preordine - exemplu

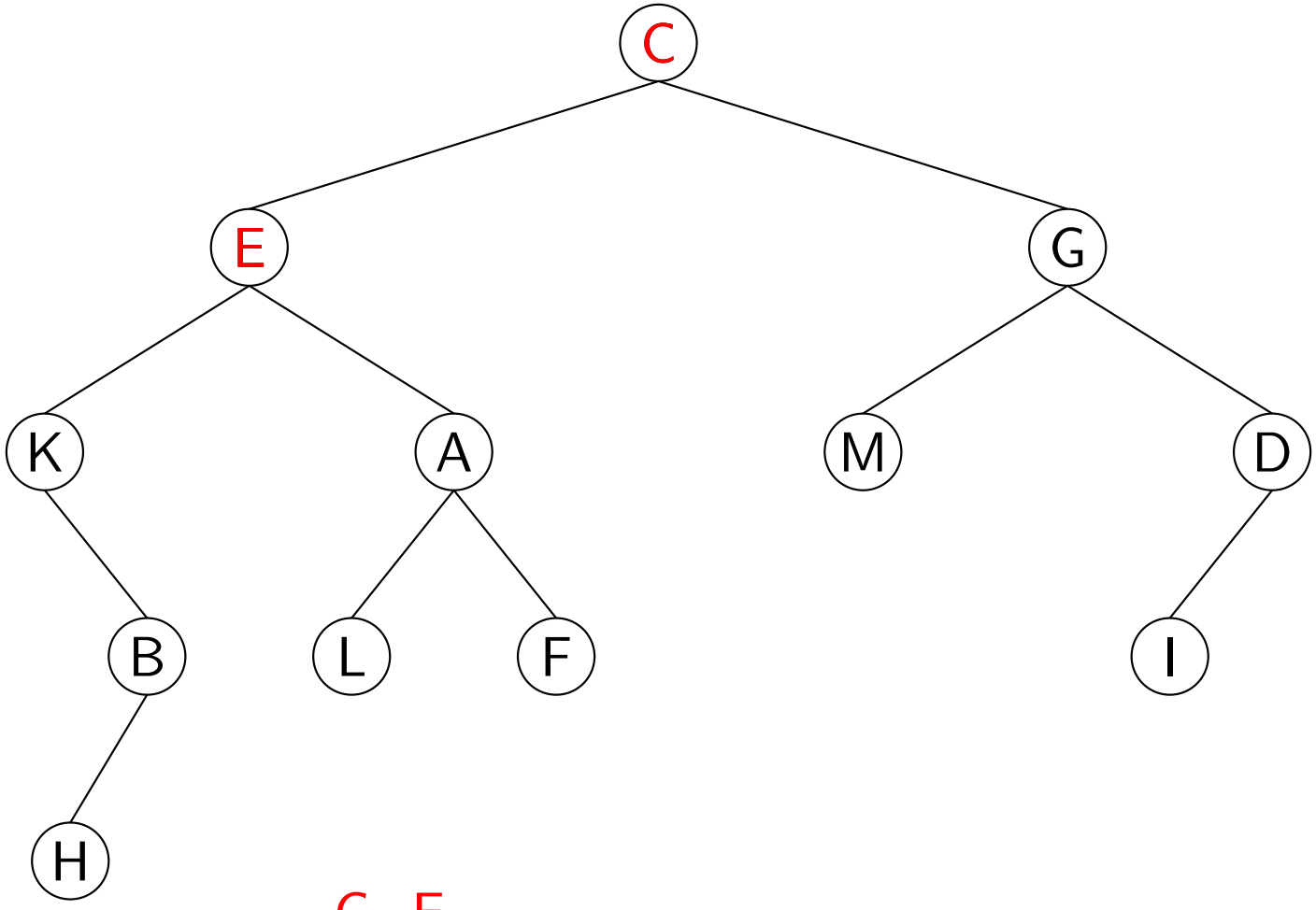


Parcurgere preordine - exemplu



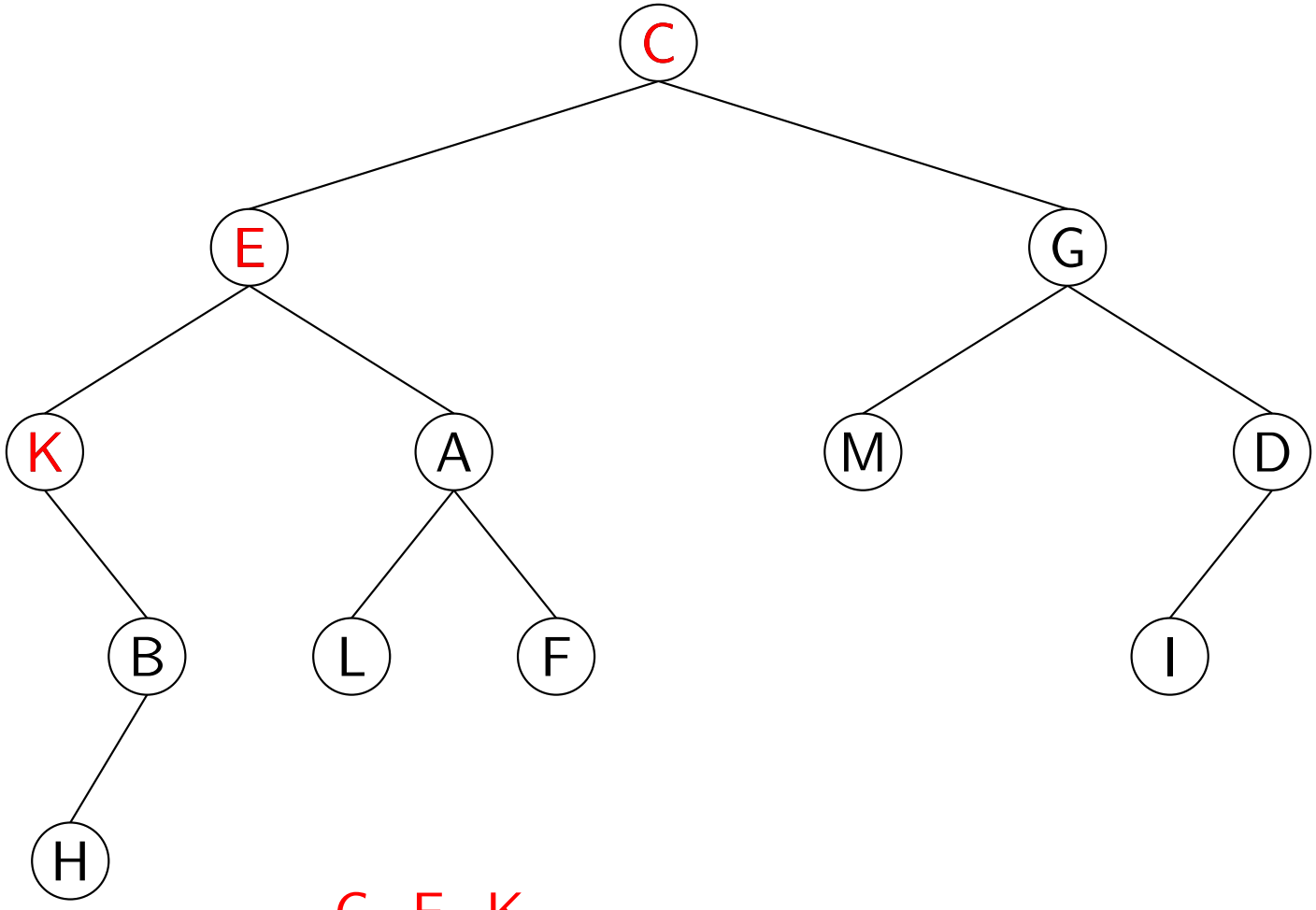
C

Parcurgere preordine - exemplu



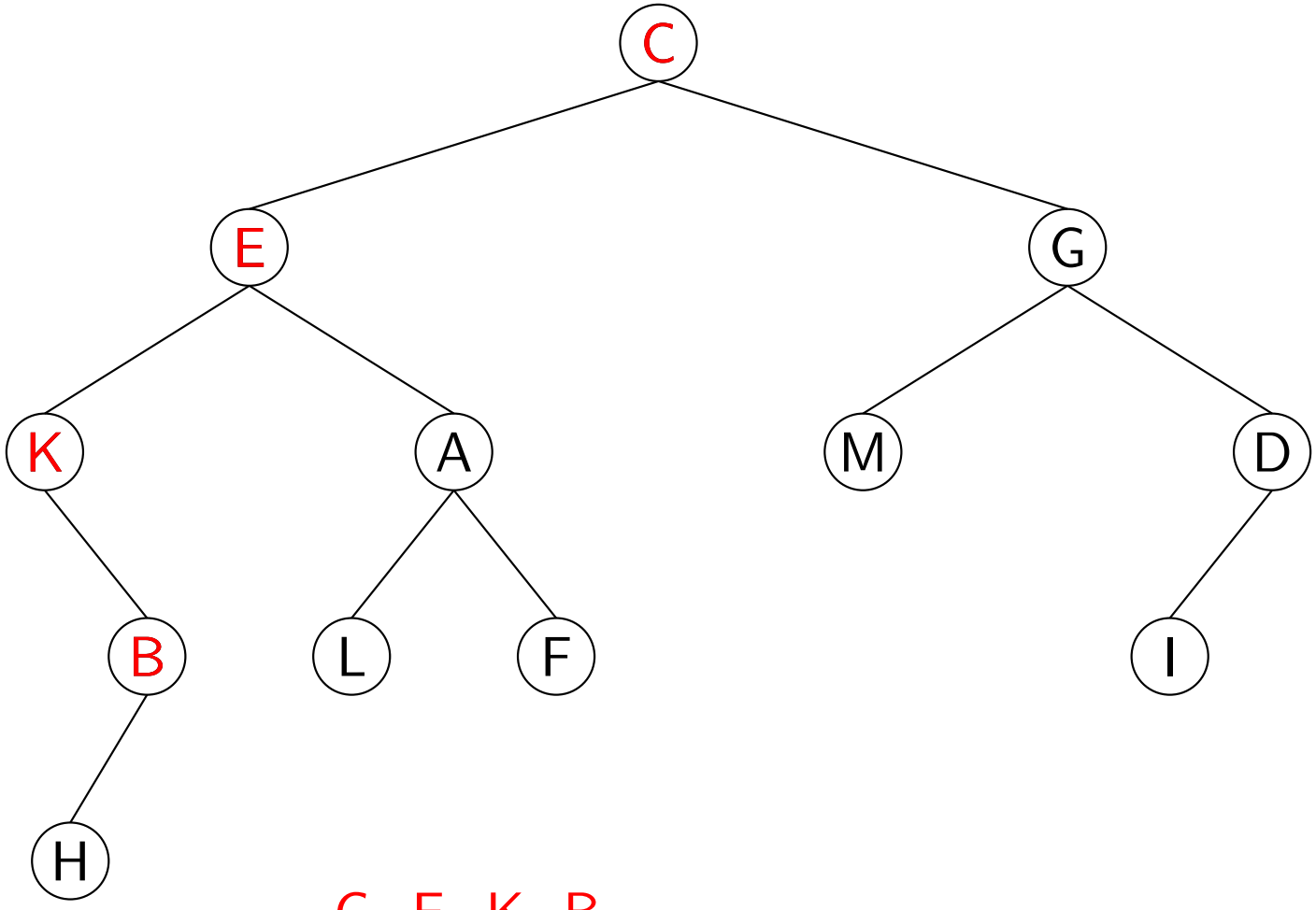
C E

Parcurgere preordine - exemplu



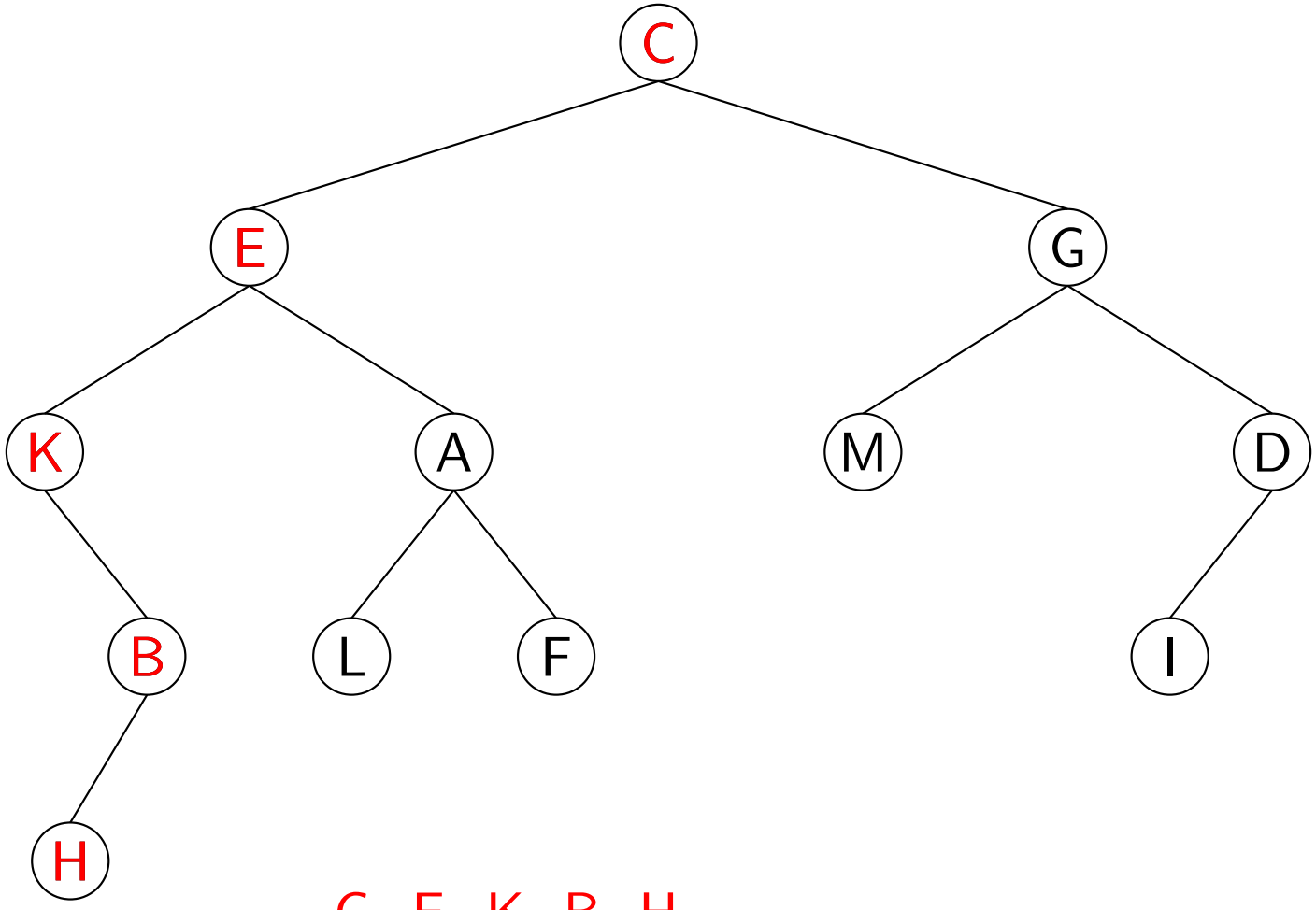
C E K

Parcurgere preordine - exemplu



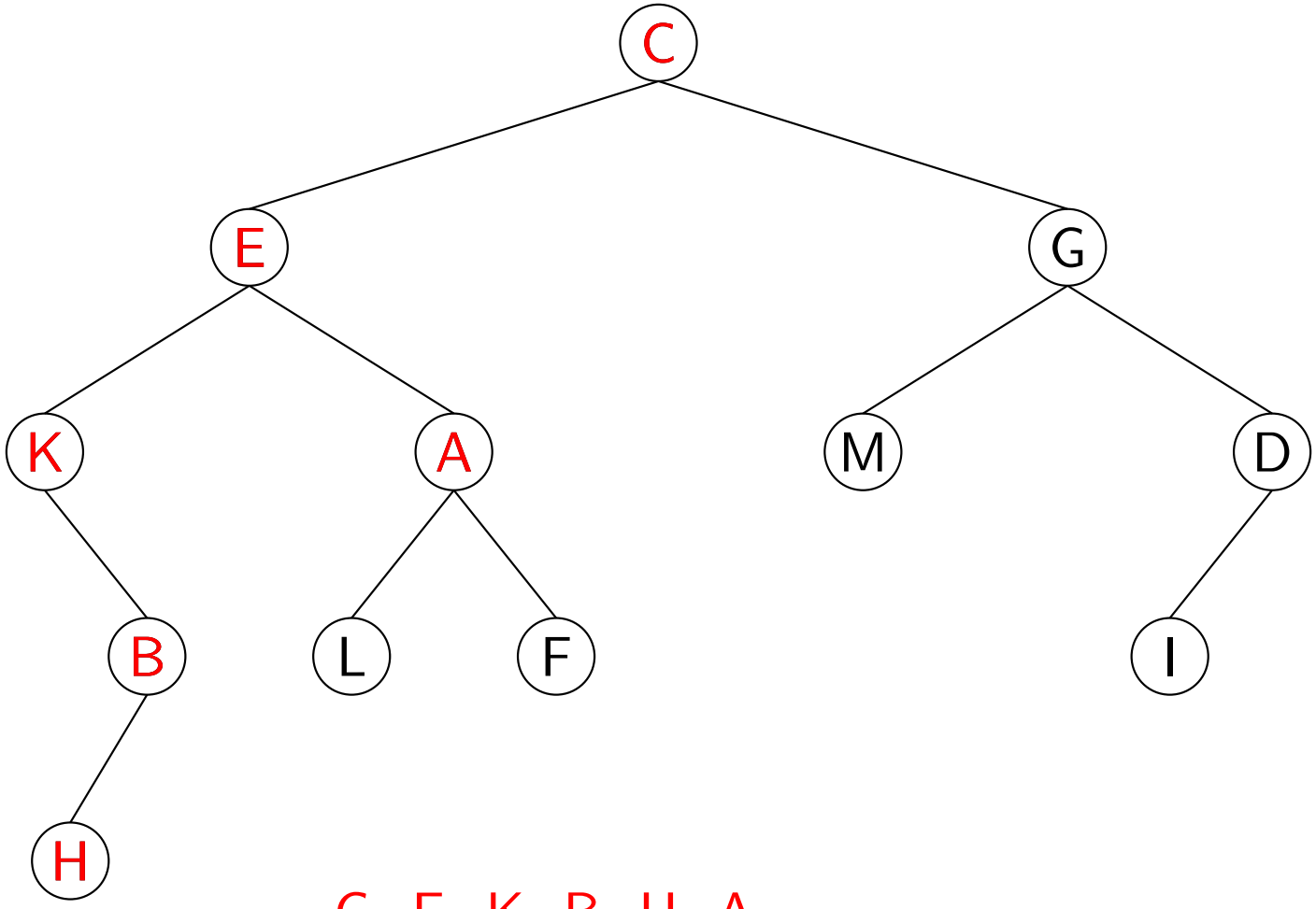
C E K B

Parcurgere preordine - exemplu



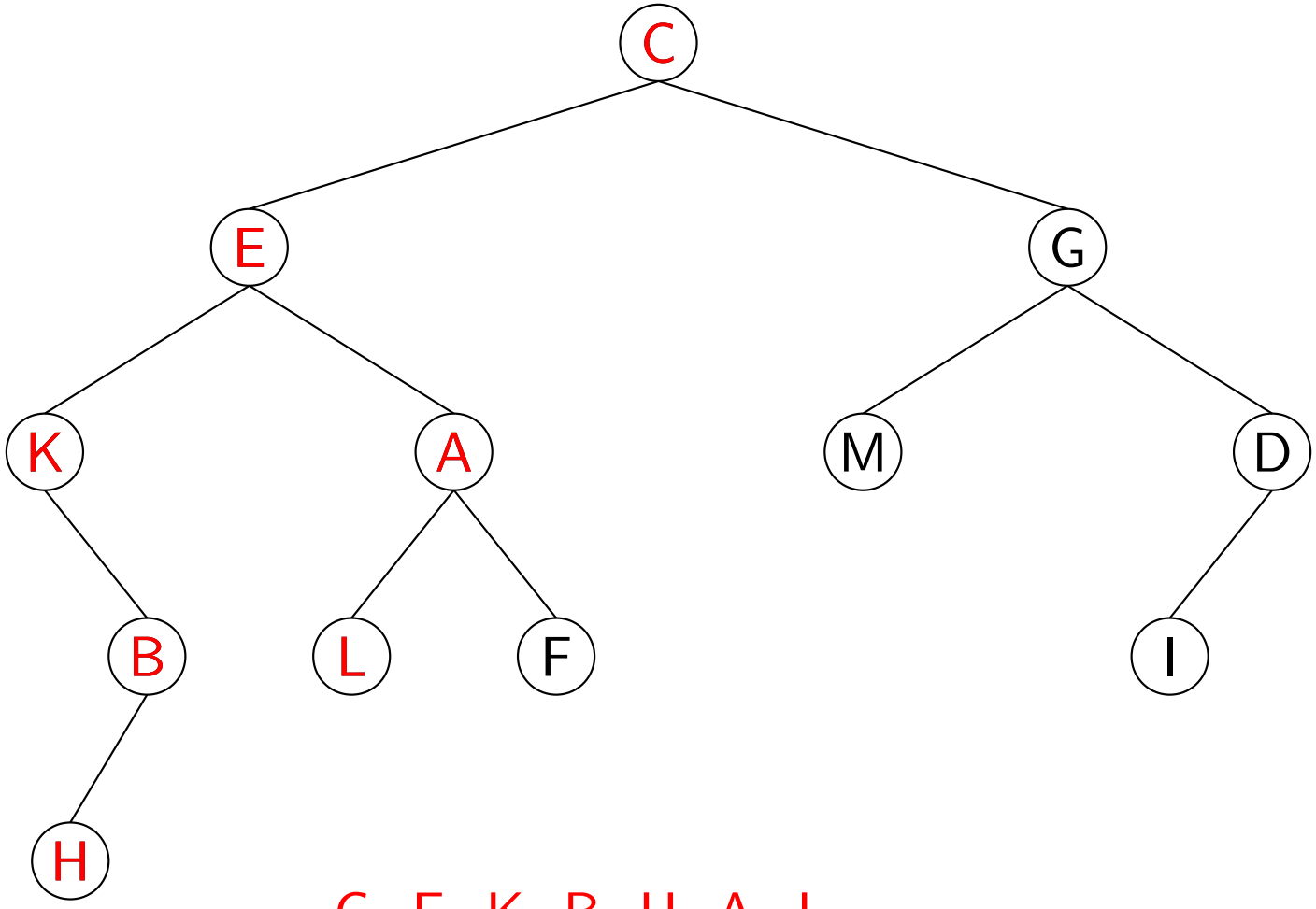
C E K B H

Parcurgere preordine - exemplu



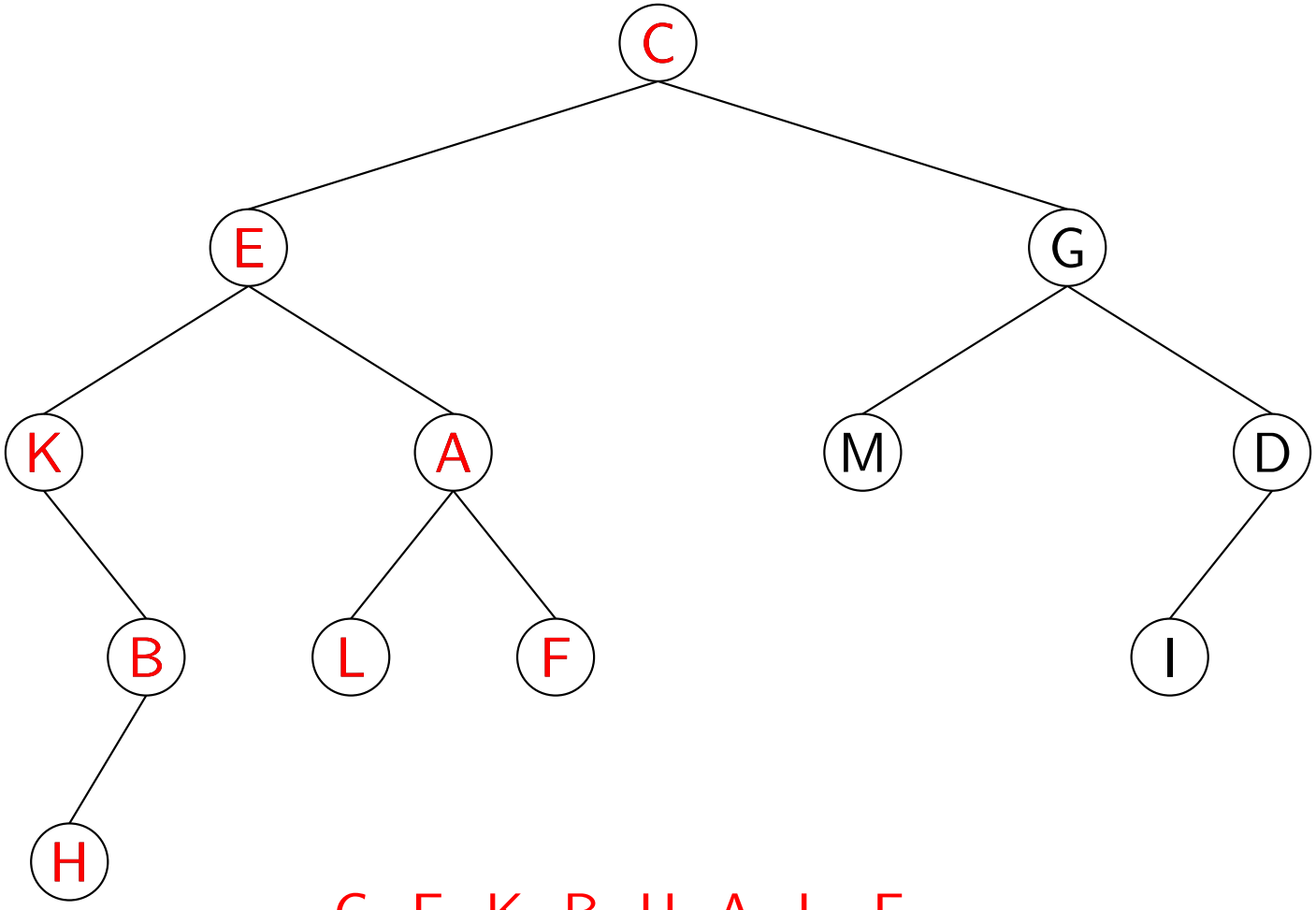
C E K B H A

Parcurgere preordine - exemplu



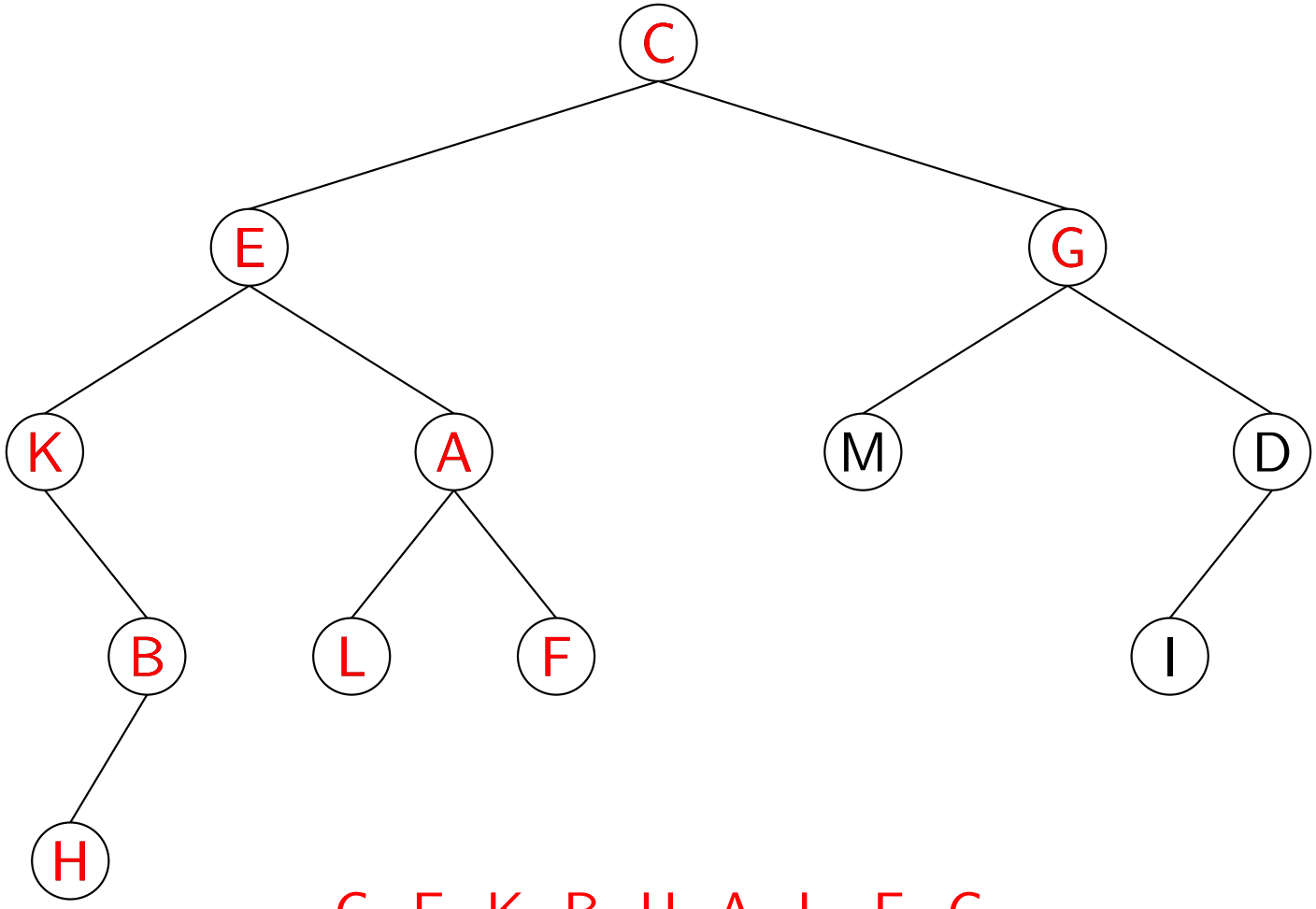
C E K B H A L

Parcurgere preordine - exemplu



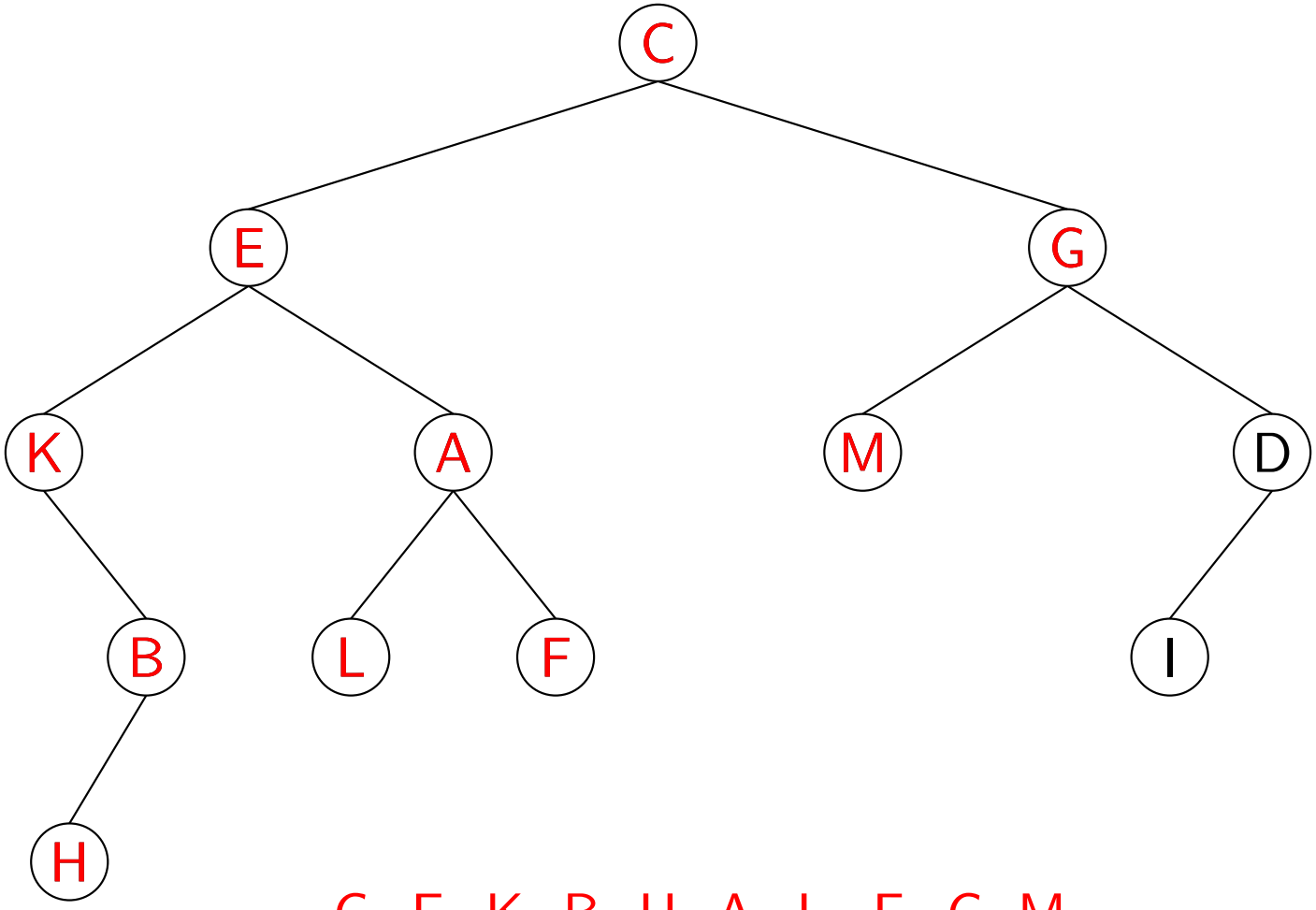
C E K B H A L F

Parcurgere preordine - exemplu



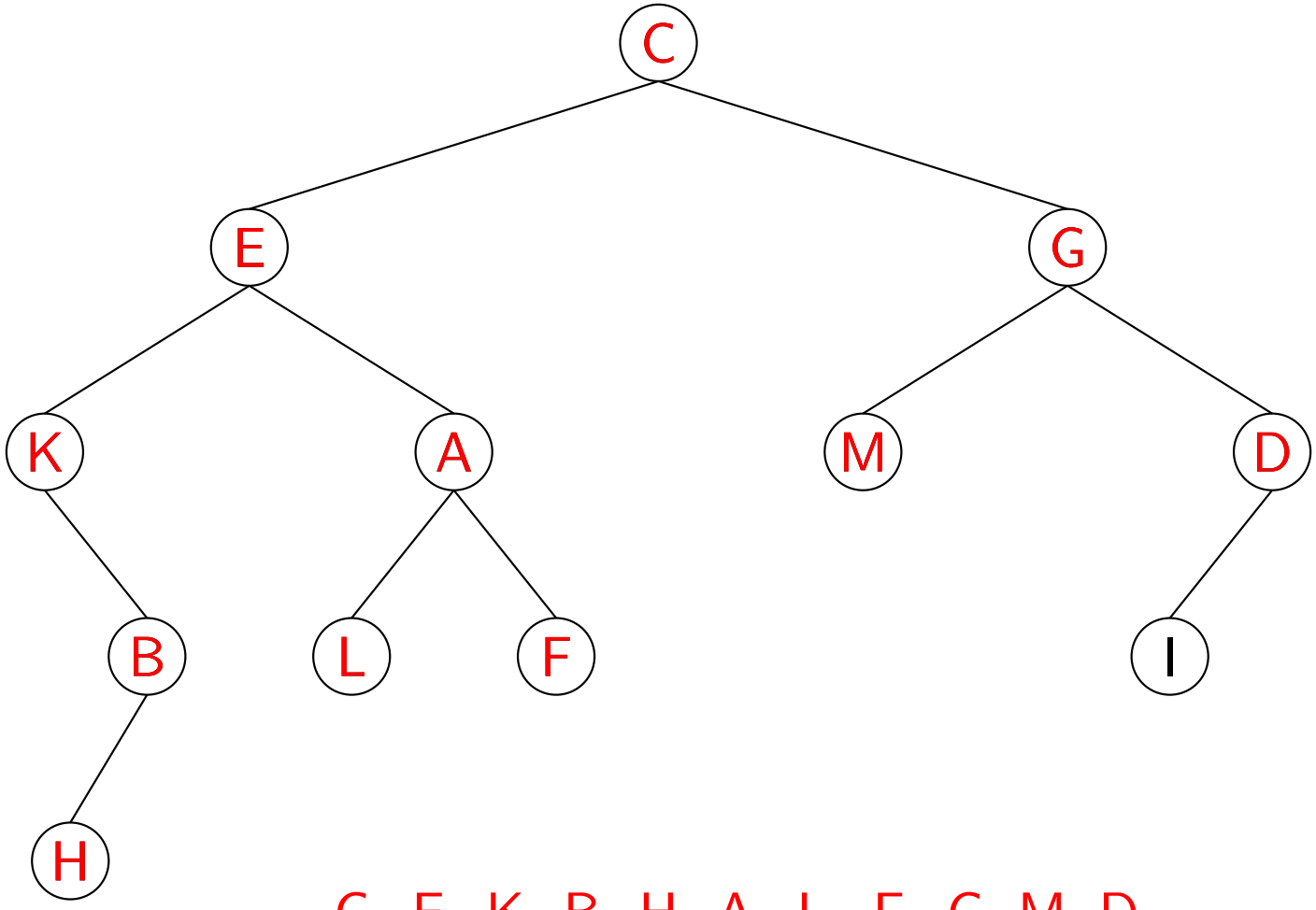
C E K B H A L F G

Parcurgere preordine - exemplu



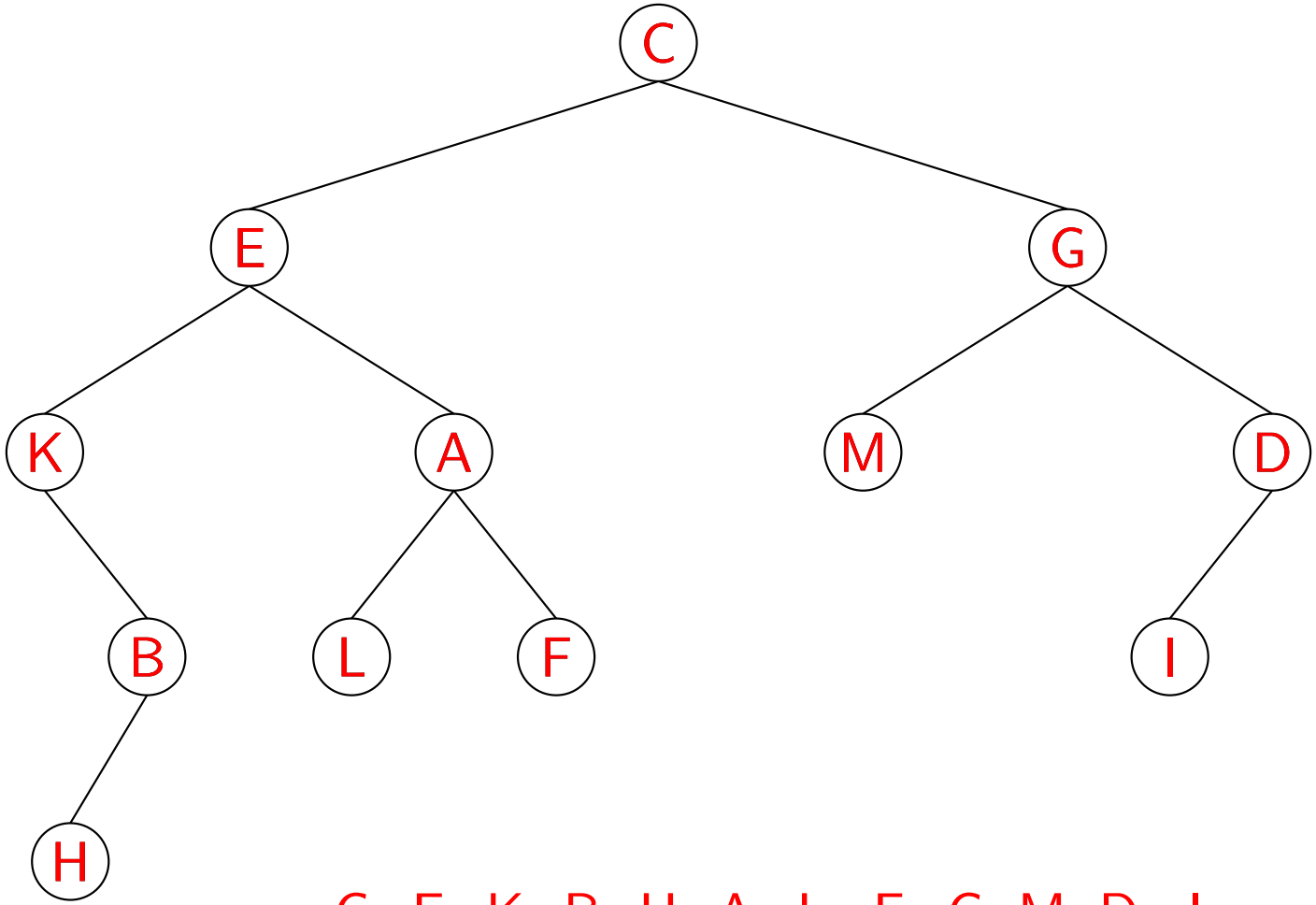
C E K B H A L F G M

Parcurgere preordine - exemplu



C E K B H A L F G M D

Parcurgere preordine - exemplu



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

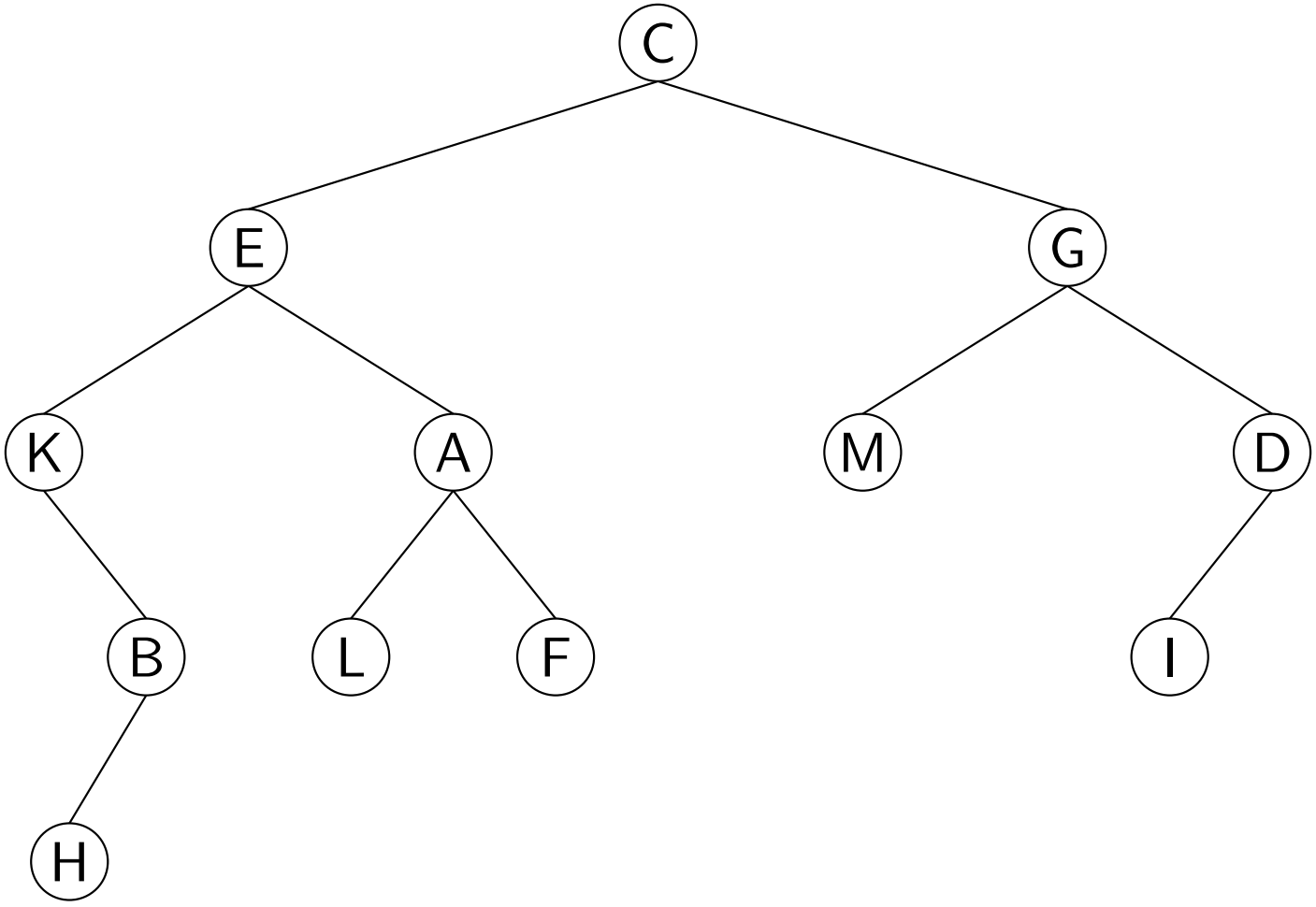
ArbBin – parcurgerea inordine

`parcurgereInordine()`

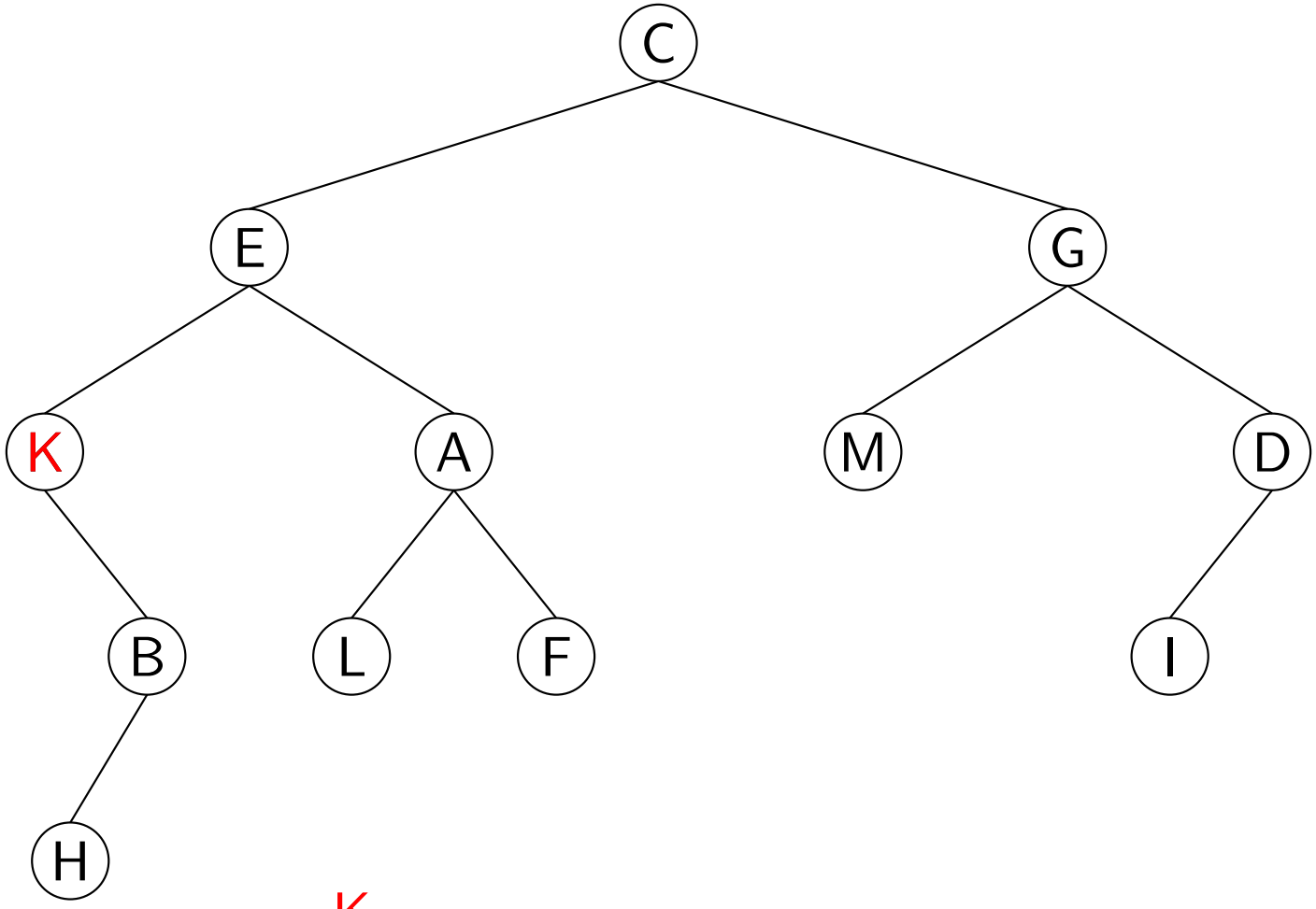
- intrare:
  - un arbore binar `t`;
  - o procedură `viziteaza()`.

- ieșire:
  - arborele `t`, dar cu nodurile procesate cu `viziteaza()` în ordinea
    * (S) – subarborele stânga
    * (R) – rădăcina
    * (D) – subarborele dreapta

Parcurgere inordine - exemplu

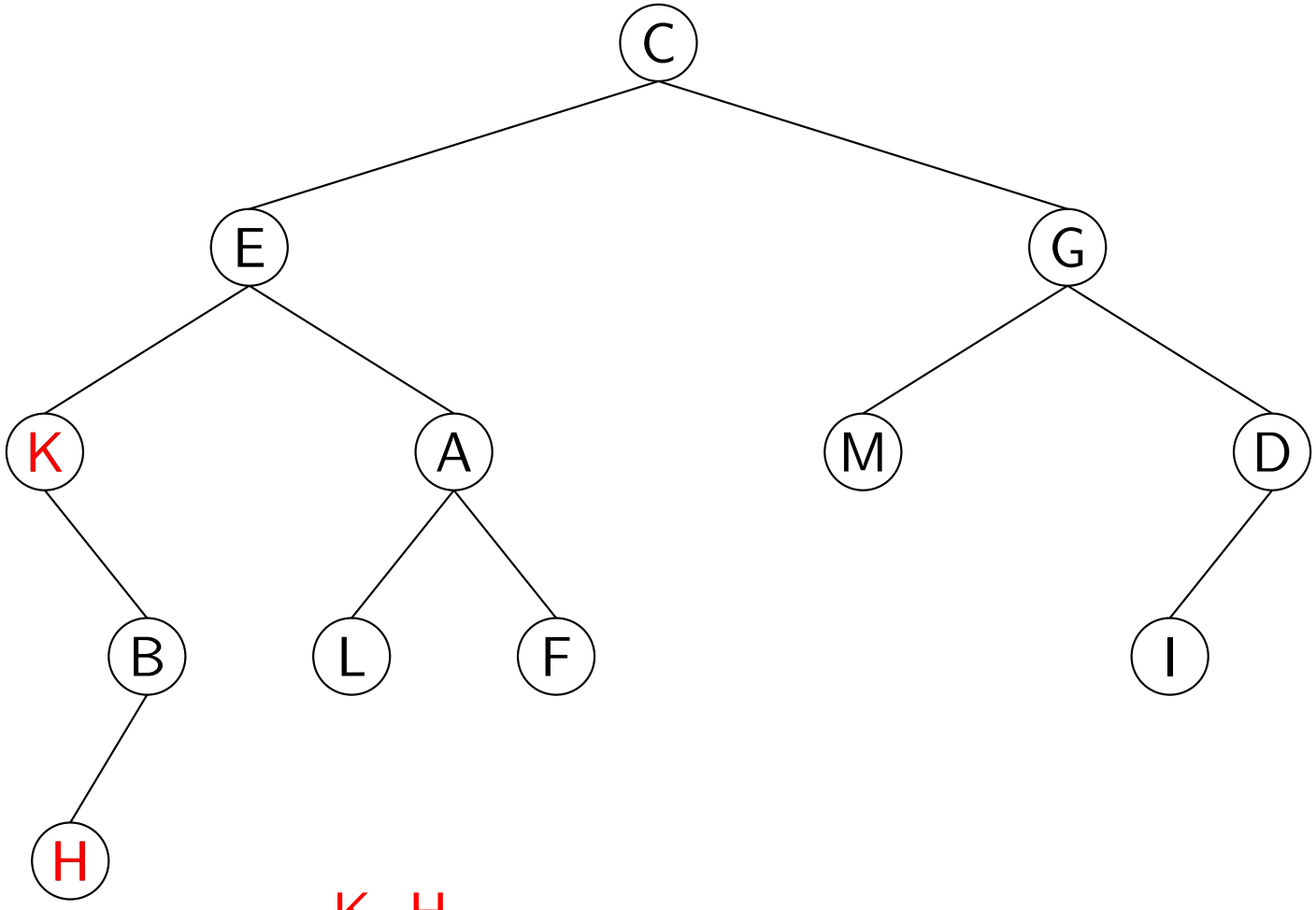


Parcurgere inordine - exemplu



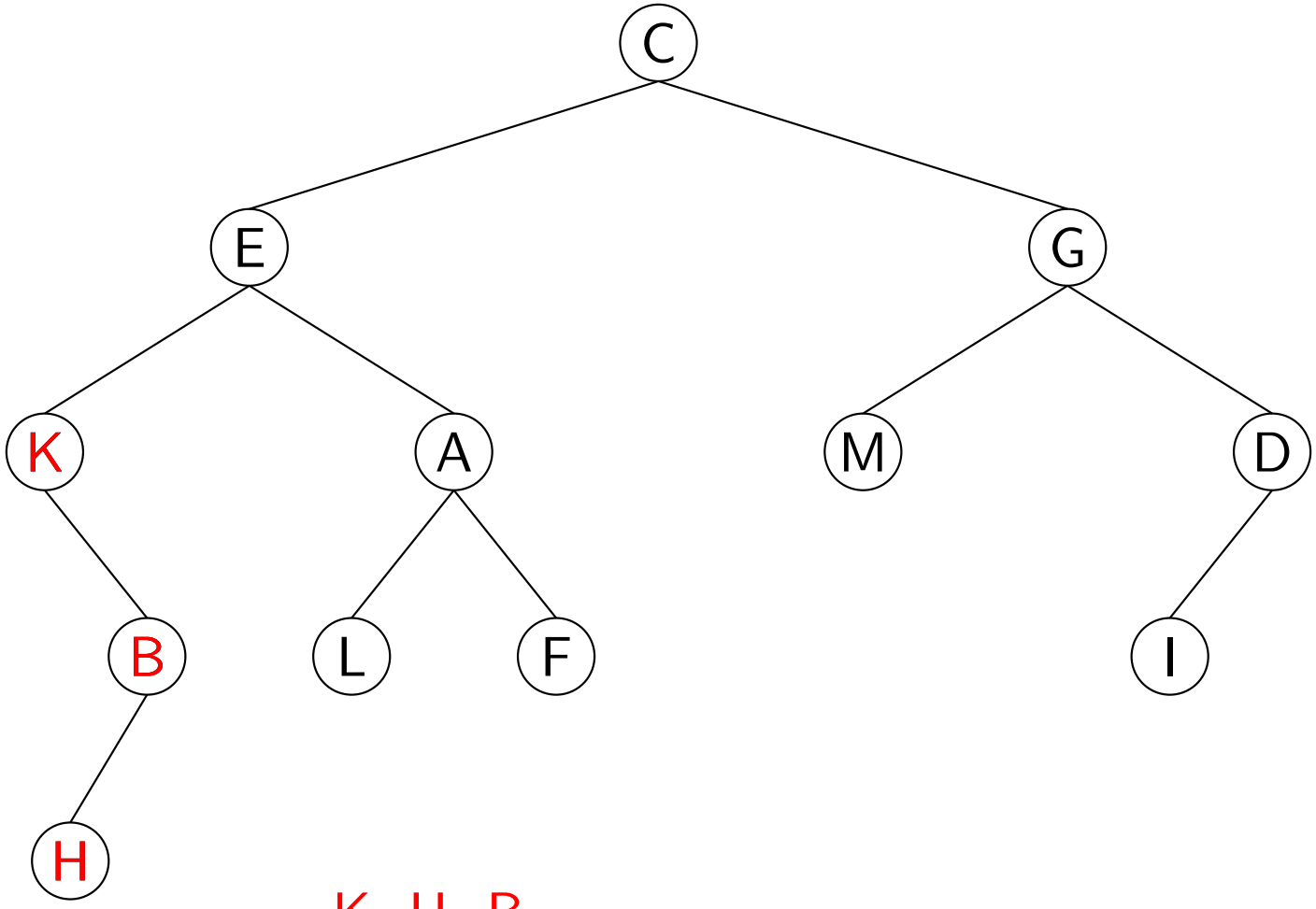
K

Parcurgere inordine - exemplu



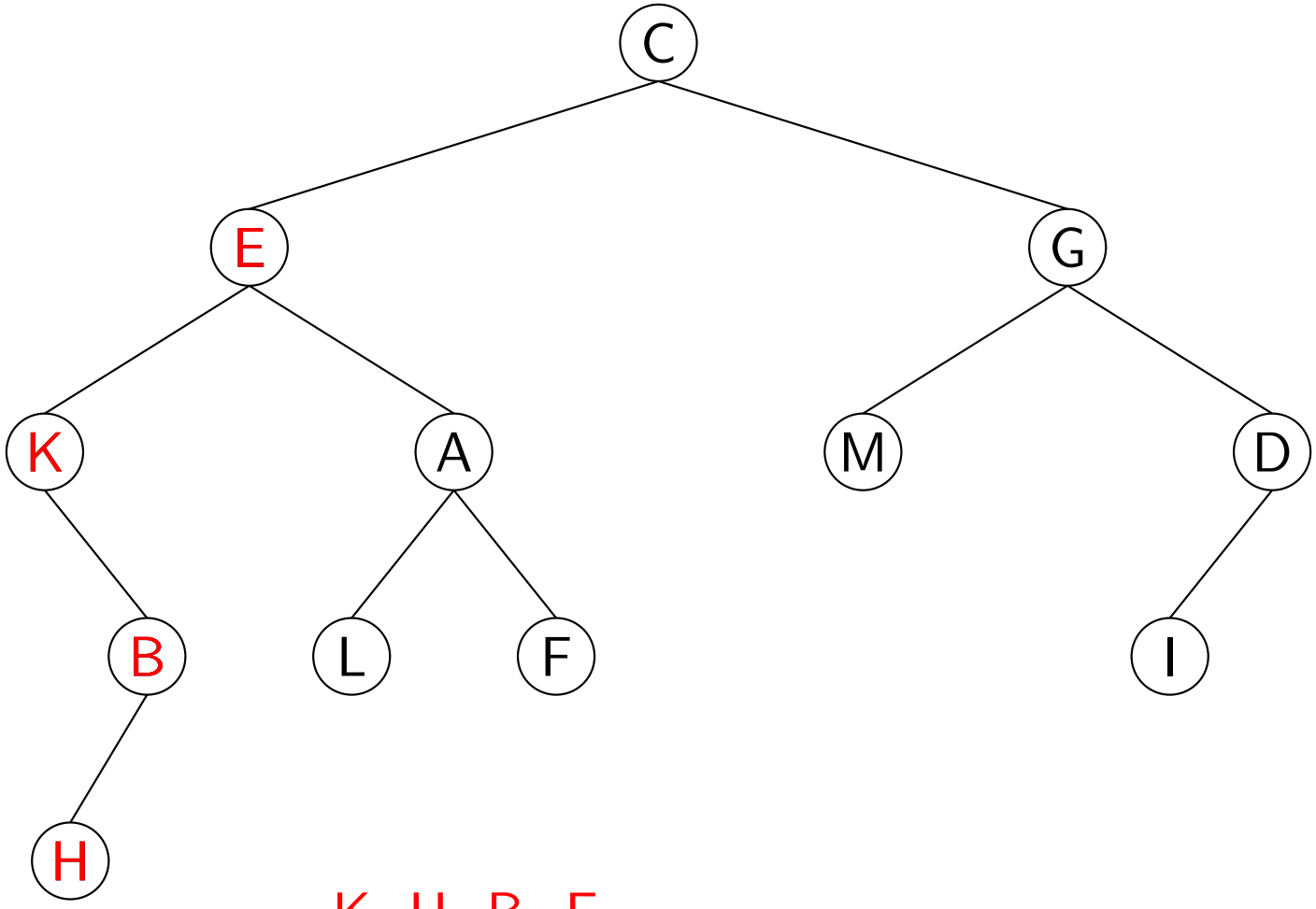
K H

Parcurgere inordine - exemplu



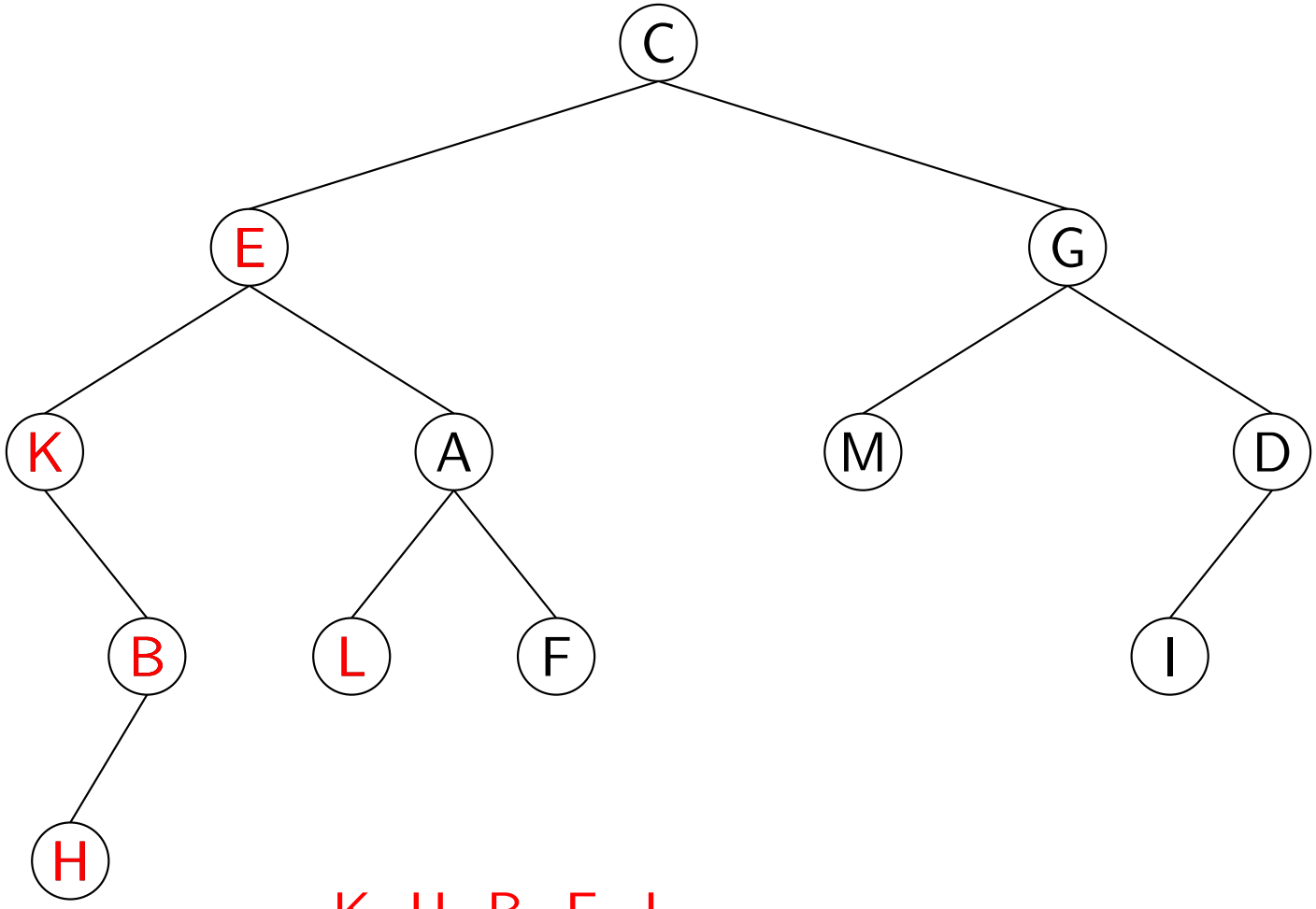
K H B

Parcurgere inordine - exemplu



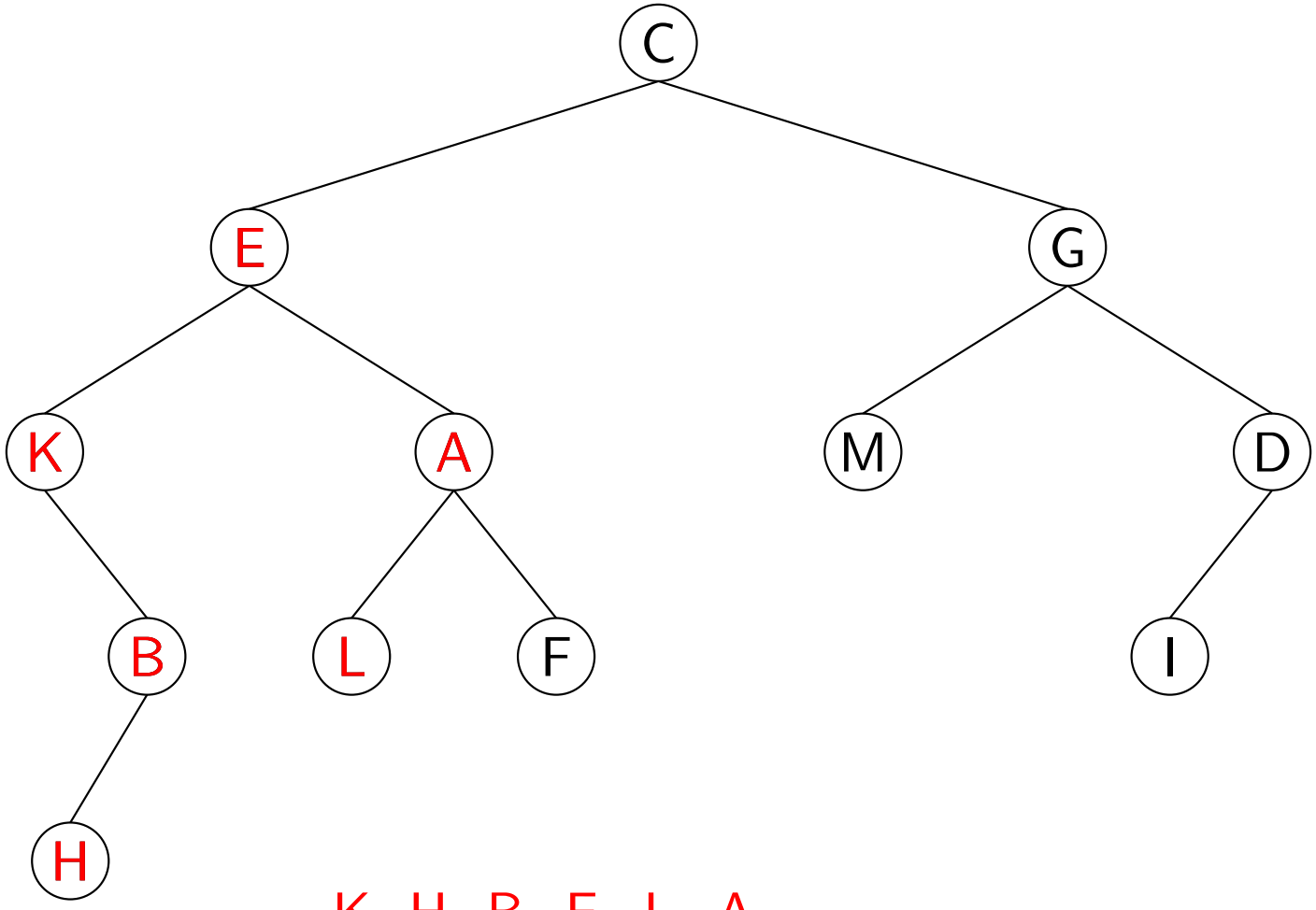
K H B E

Parcurgere inordine - exemplu



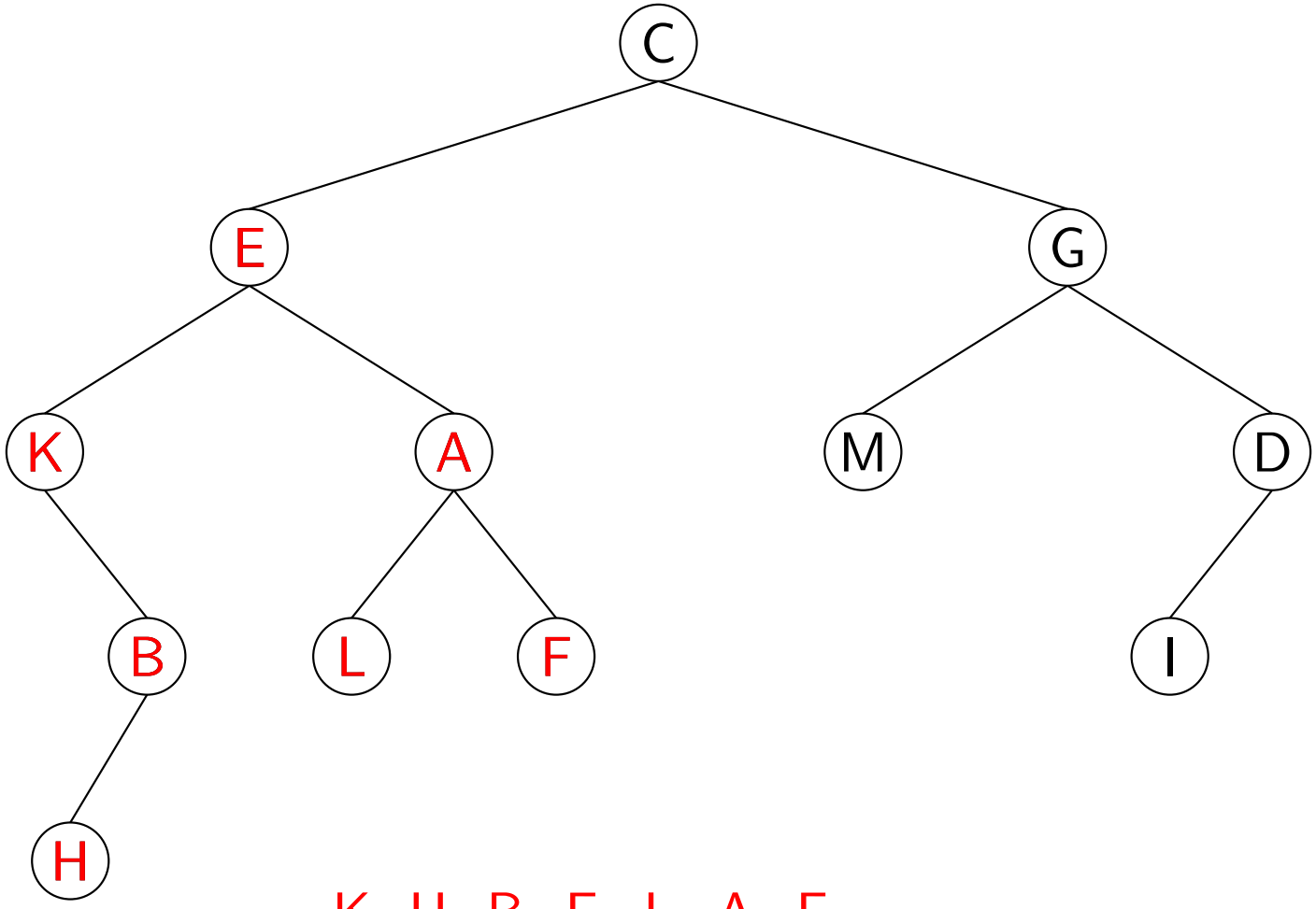
K H B E L

Parcurgere inordine - exemplu



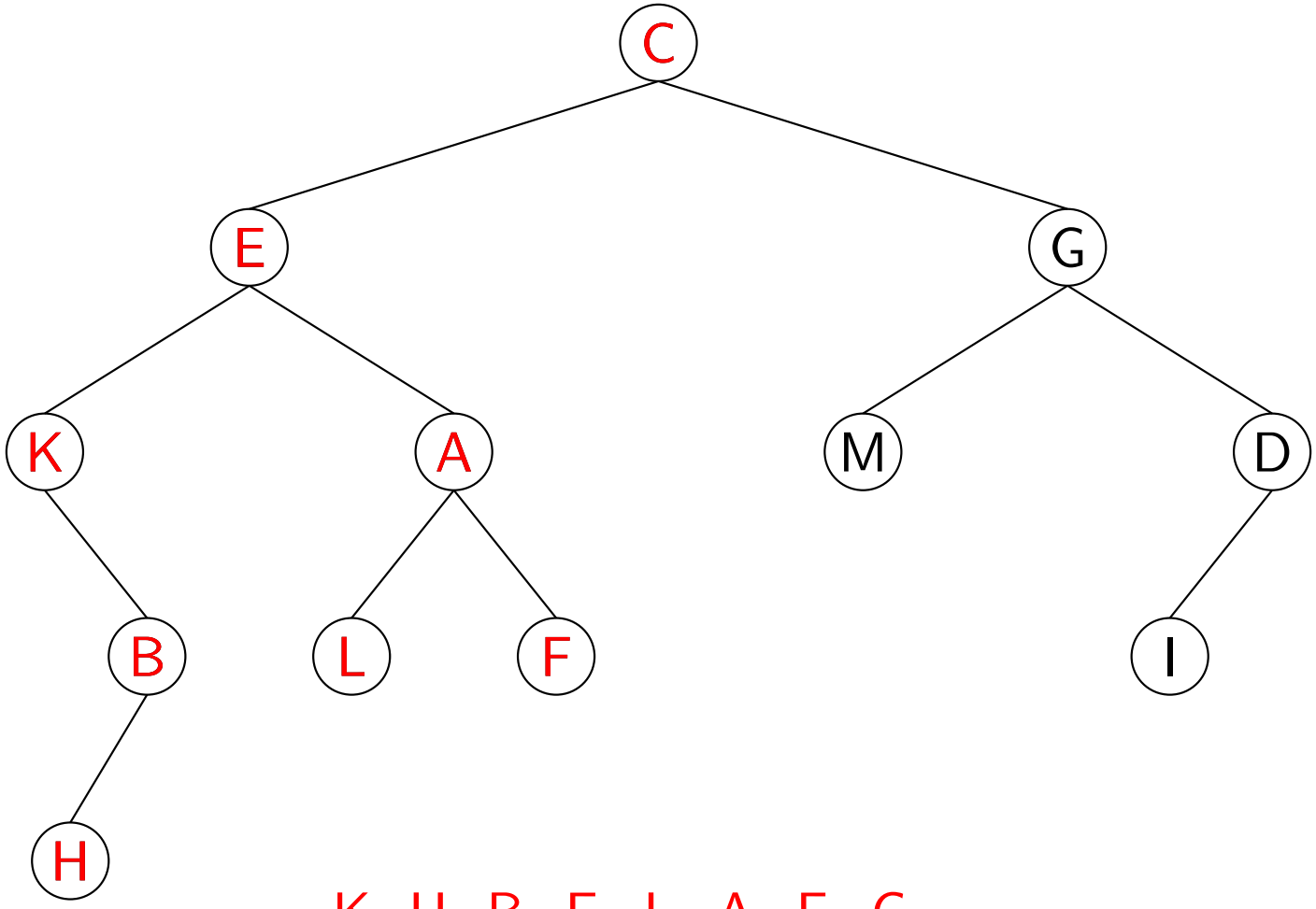
K H B E L A

Parcurgere inordine - exemplu



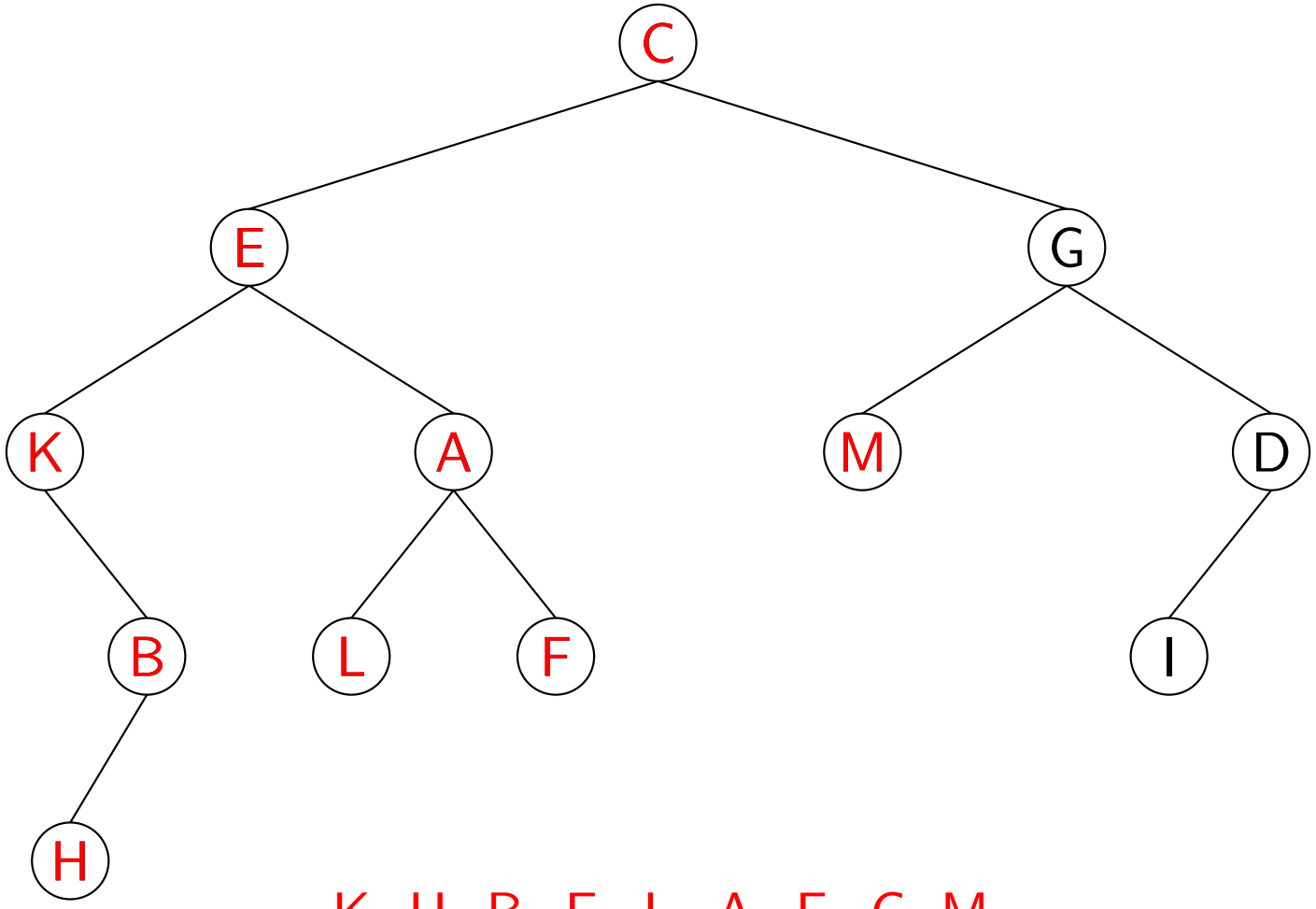
K H B E L A F

Parcurgere inordine - exemplu



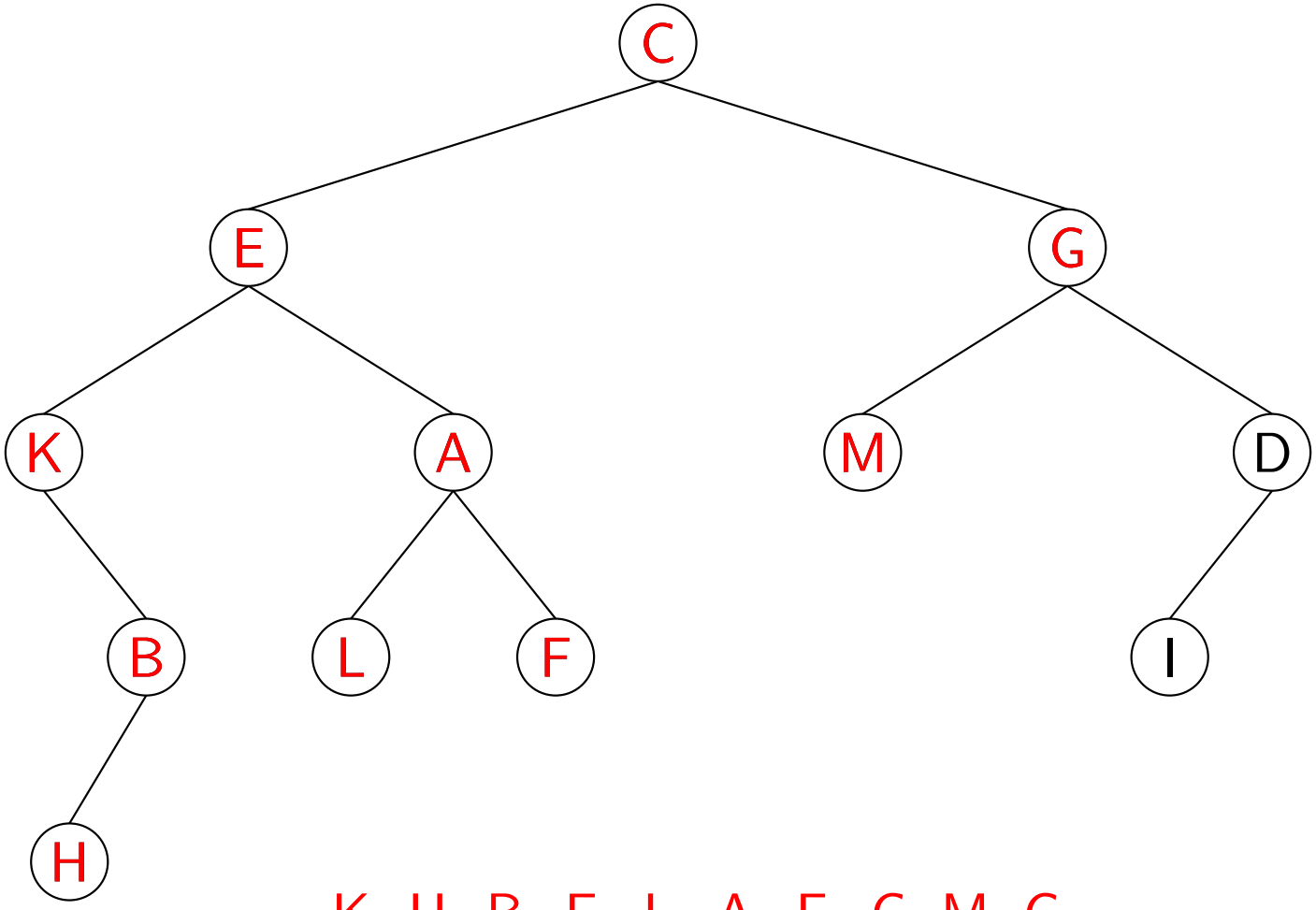
K H B E L A F C

Parcurgere inordine - exemplu



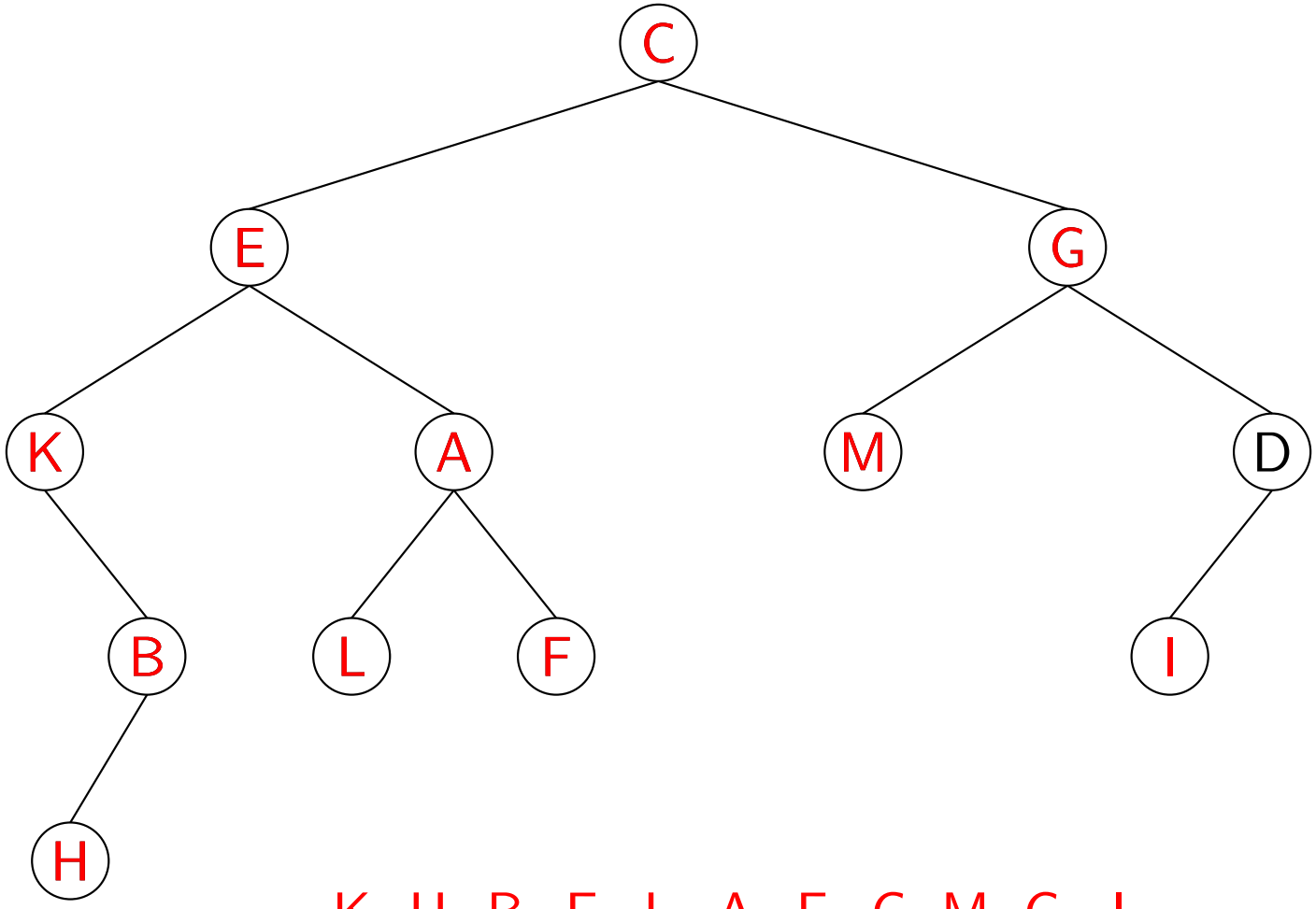
K H B E L A F C M

Parcurgere inordine - exemplu



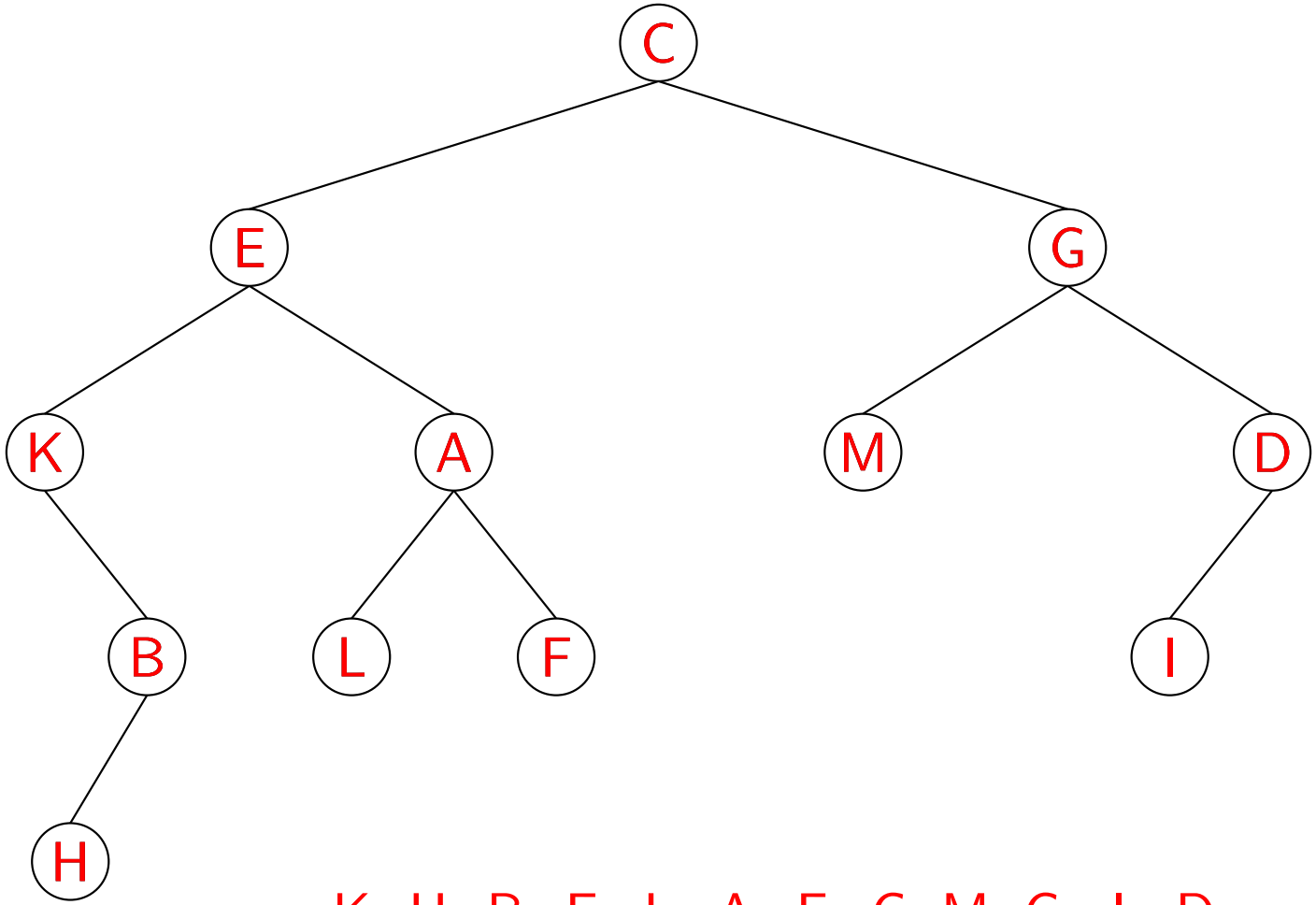
K H B E L A F C M G

Parcurgere inordine - exemplu



K H B E L A F C M G I

Parcurgere inordine - exemplu



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

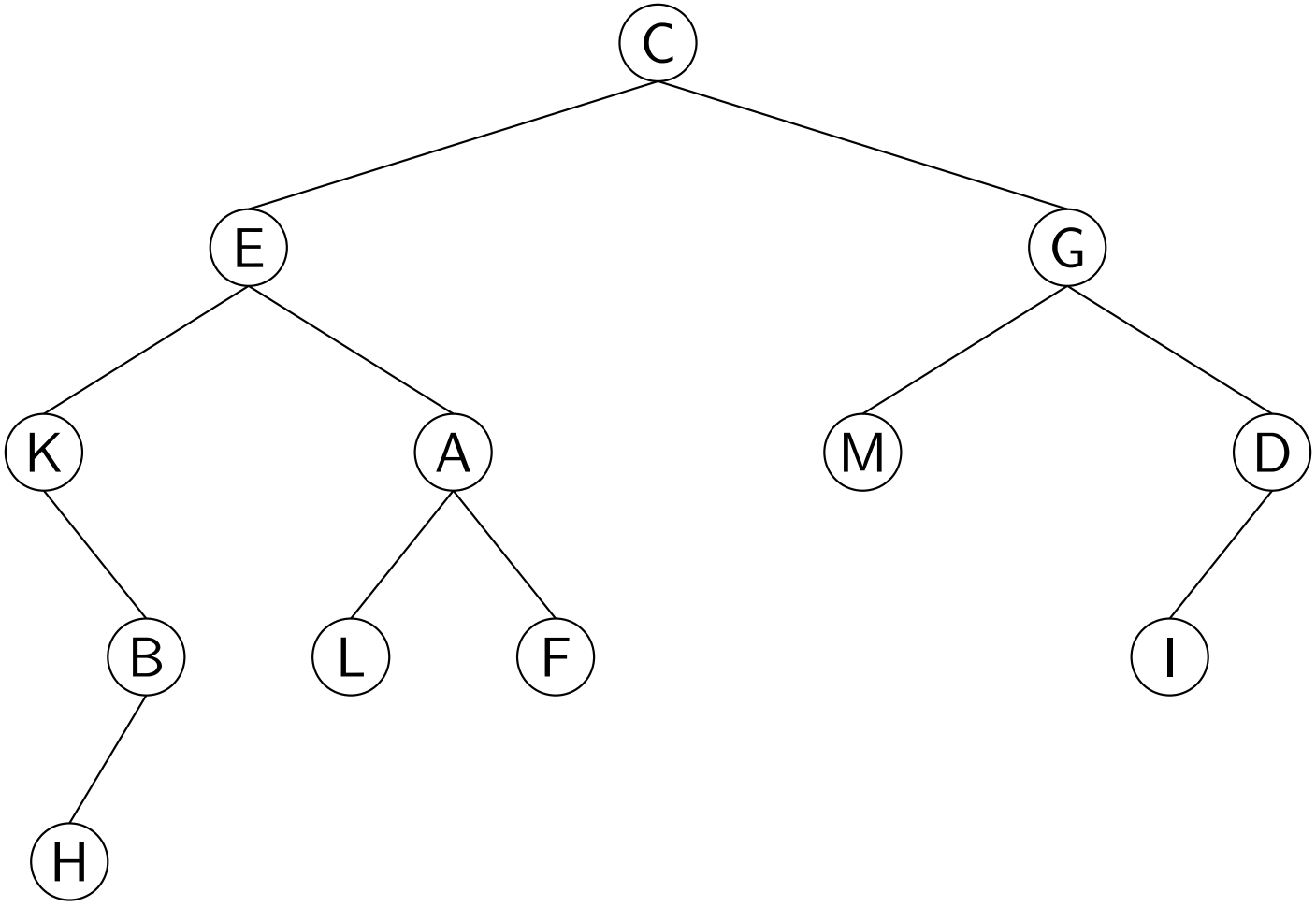
ArbBin – parcurgerea postordine

`parcurgerePostordine()`

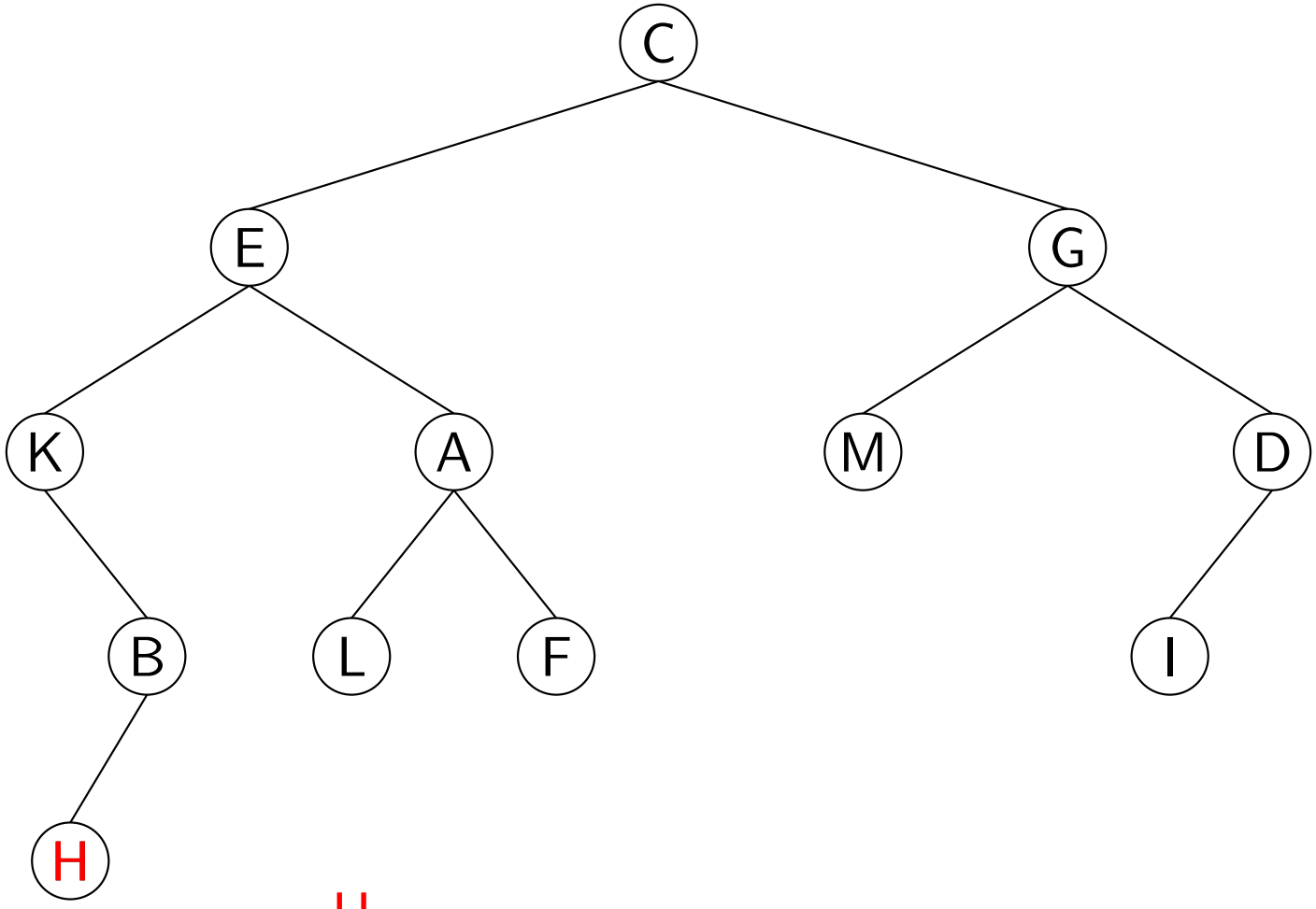
- intrare:
  - un arbore binar `t`;
  - o procedură `viziteaza()`.

- ieșire:
  - arborele `t`, dar cu nodurile procesate cu `viziteaza()` în ordinea
    * (S) – subarborele stânga
    * (D) – subarborele dreapta
    * (R) – rădăcina

Parcurgere postordine - exemplu

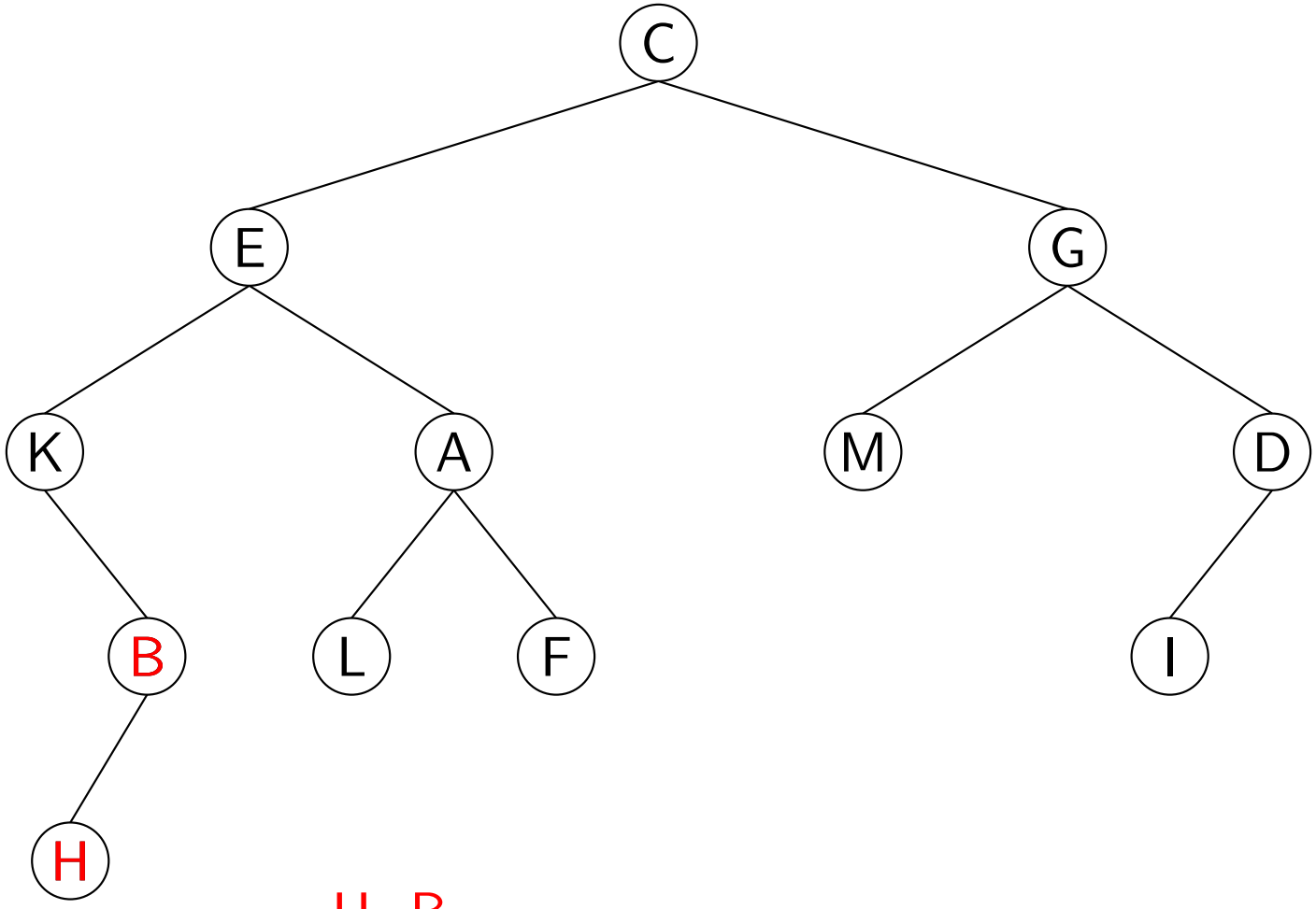


Parcurgere postordine - exemplu



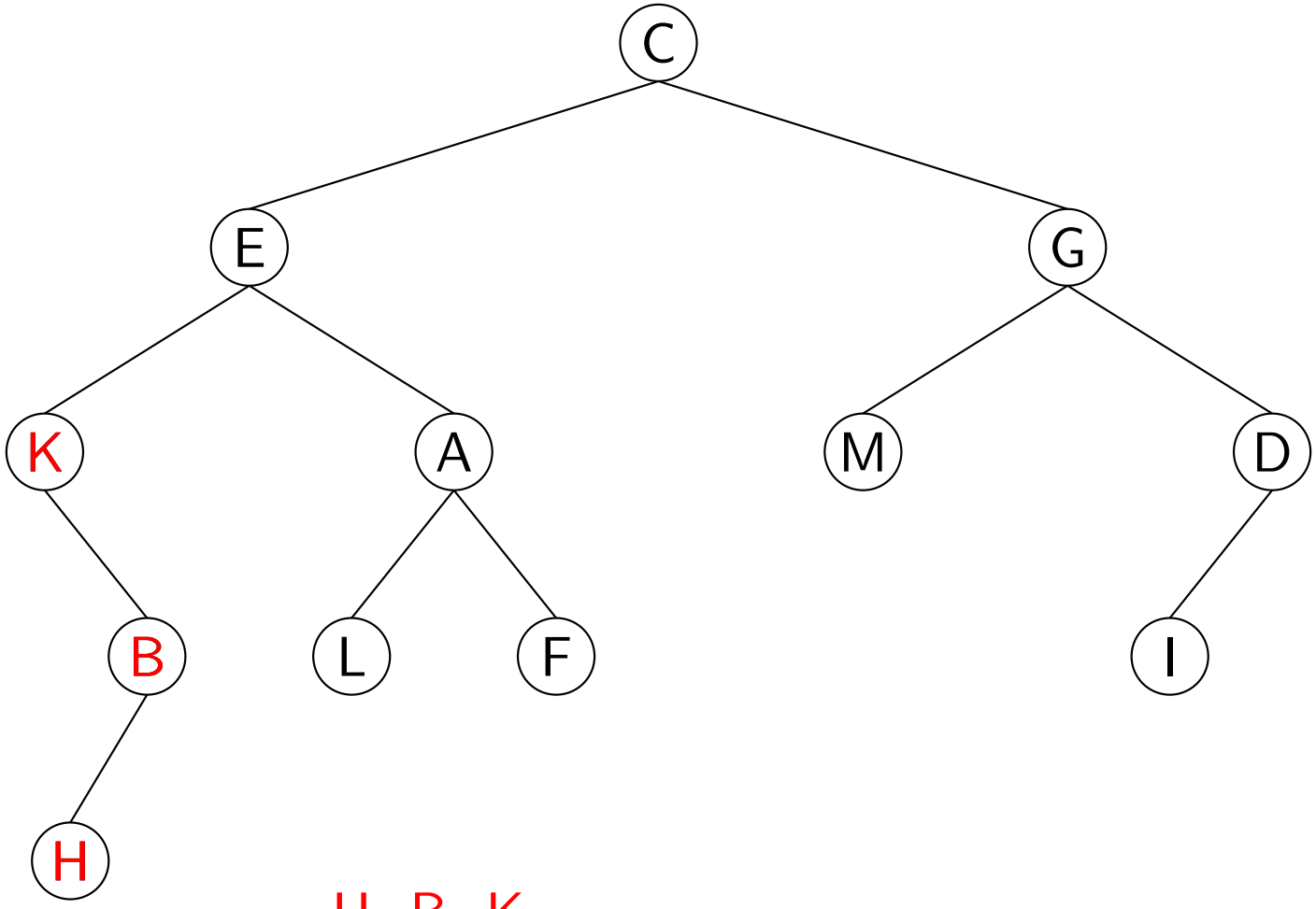
H

Parcurgere postordine - exemplu



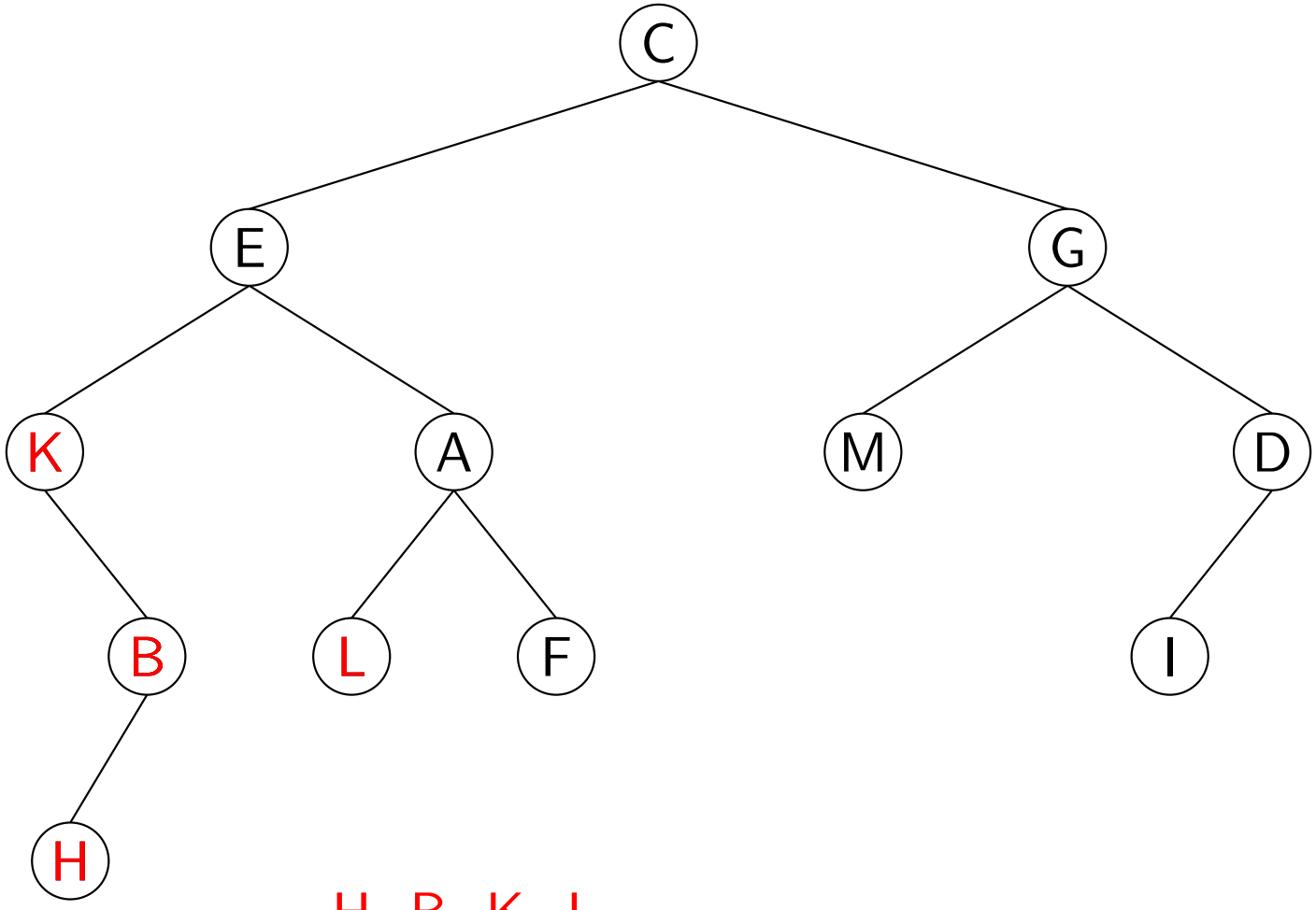
H B

Parcurgere postordine - exemplu



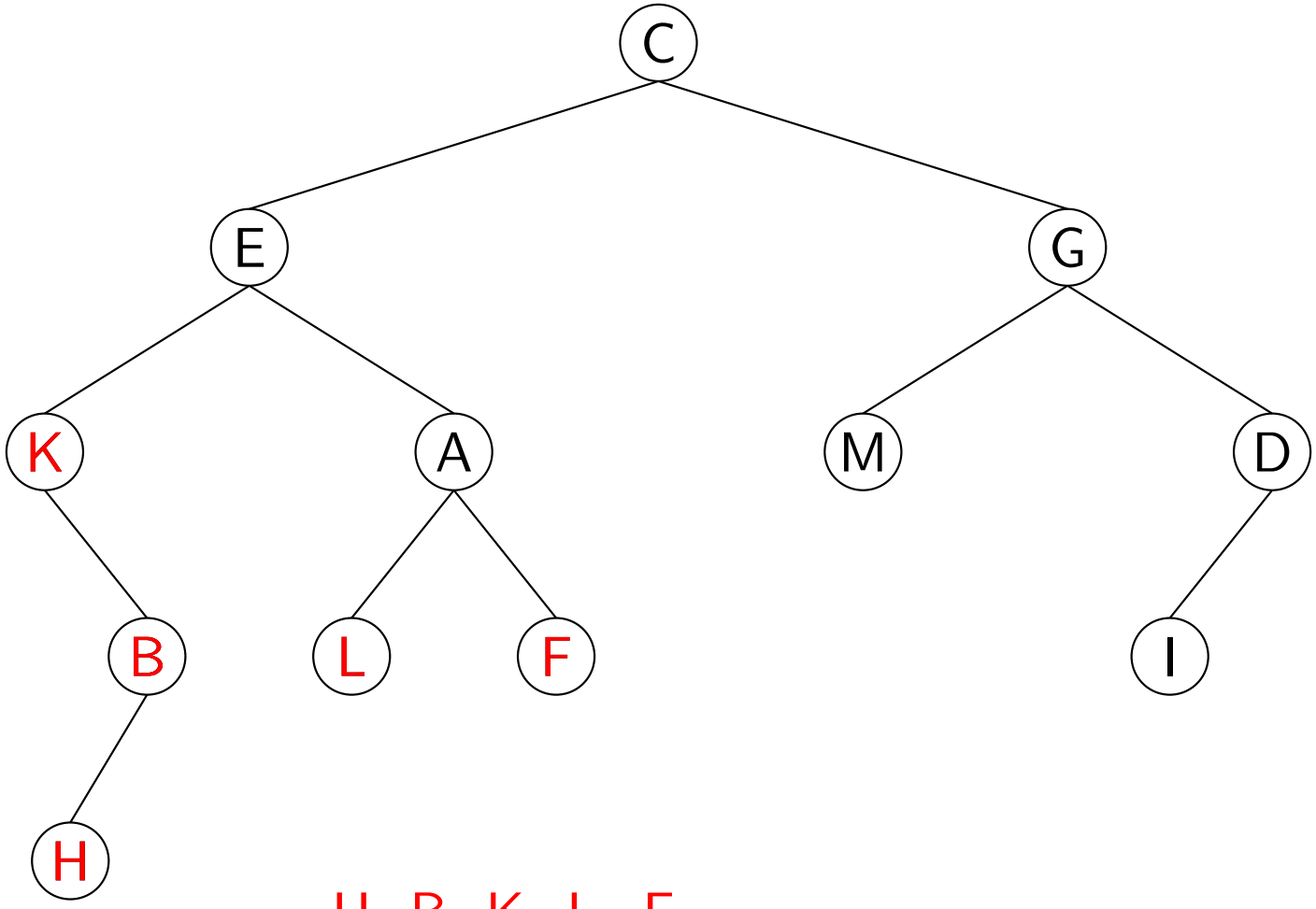
H B K

Parcurgere postordine - exemplu



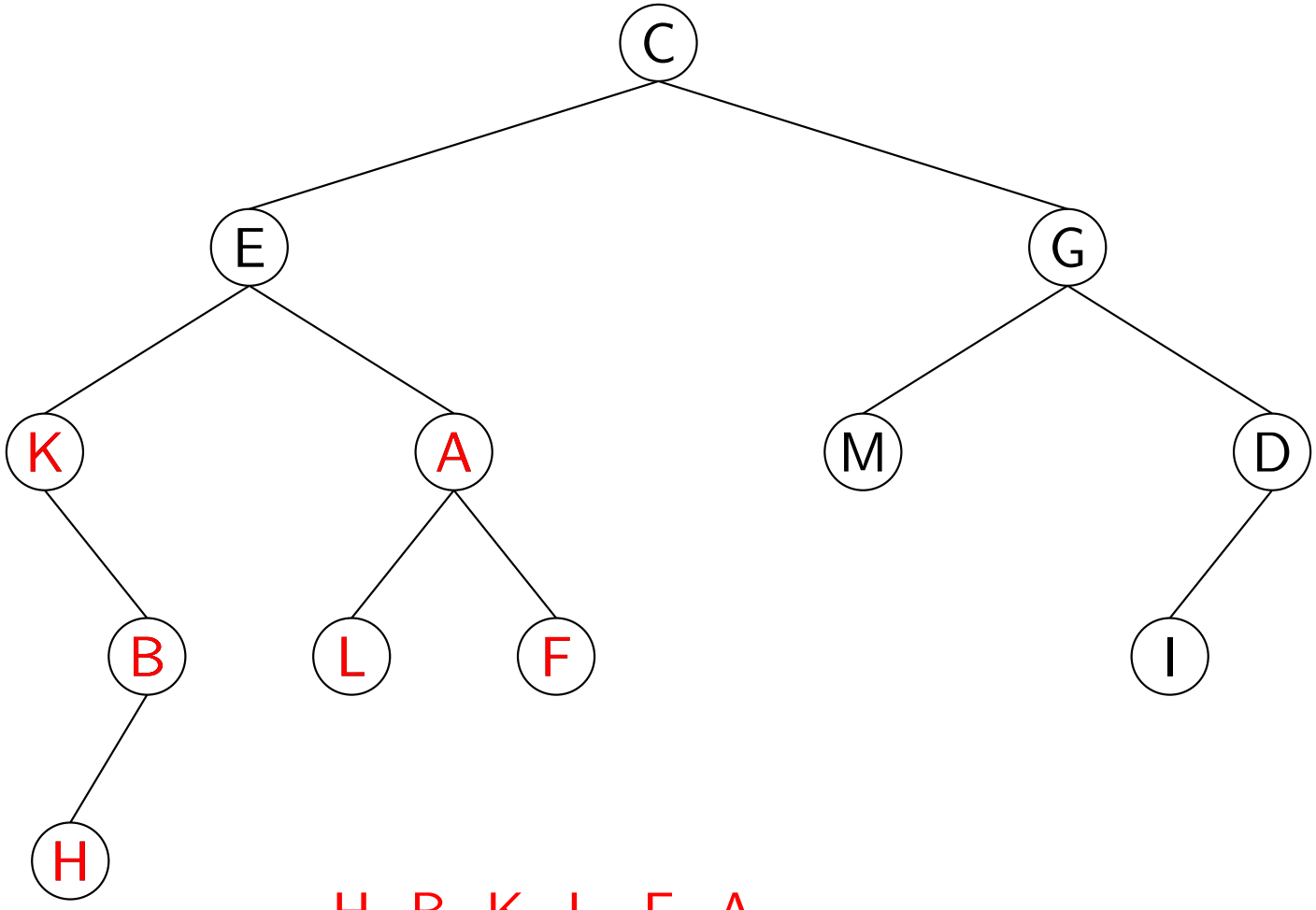
H B K L

Parcurgere postordine - exemplu



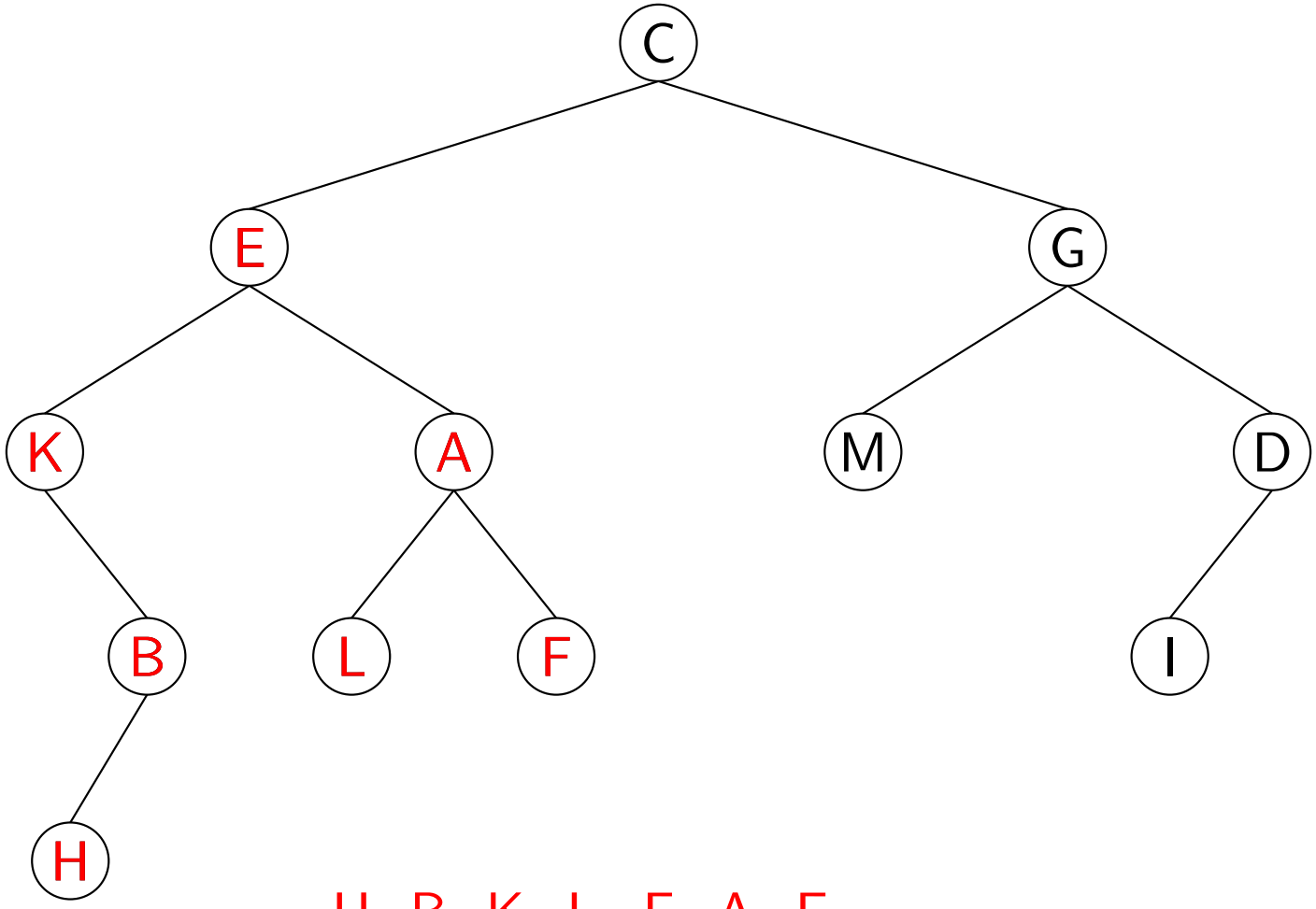
H B K L F

Parcurgere postordine - exemplu



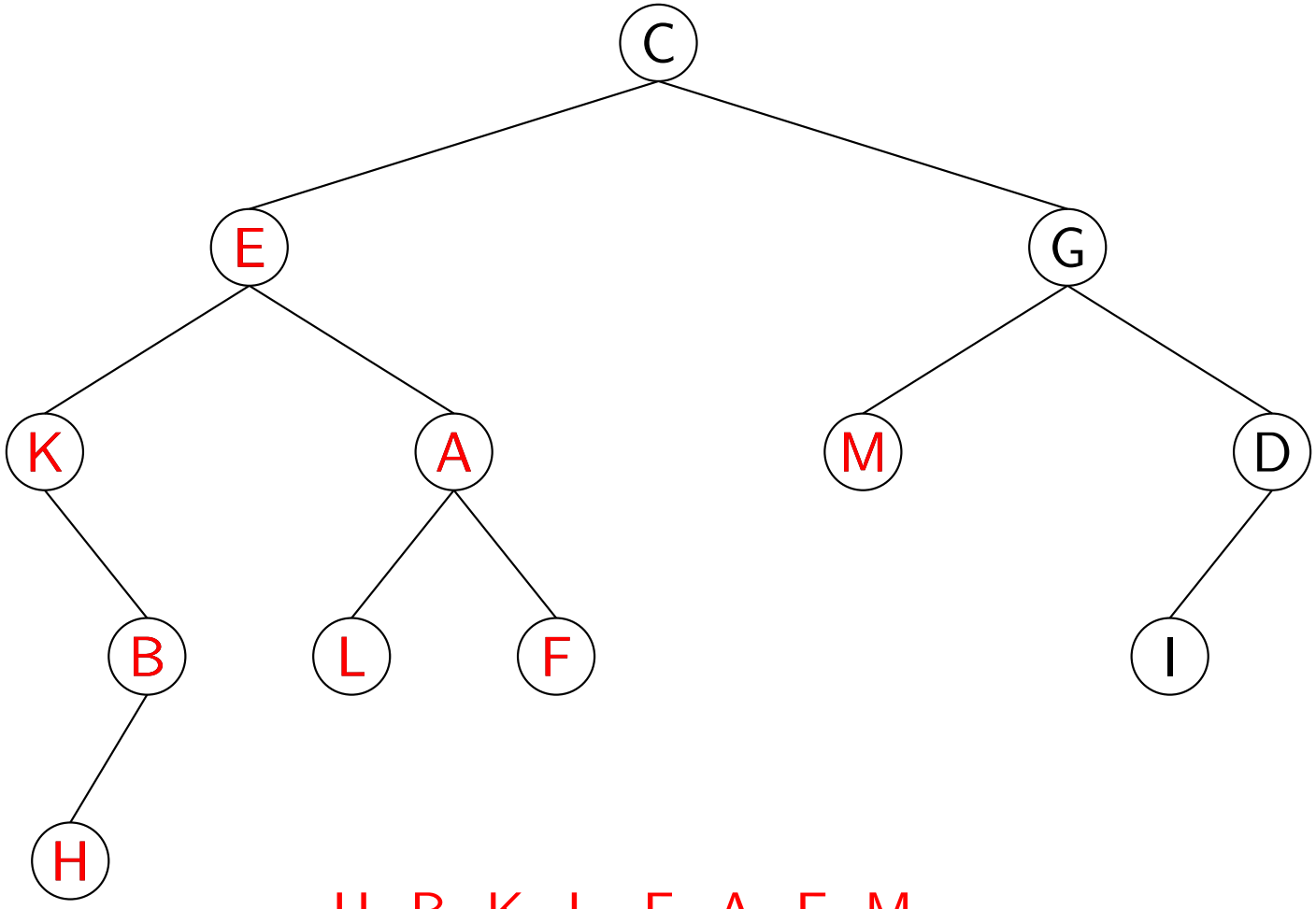
H B K L F A

Parcurgere postordine - exemplu



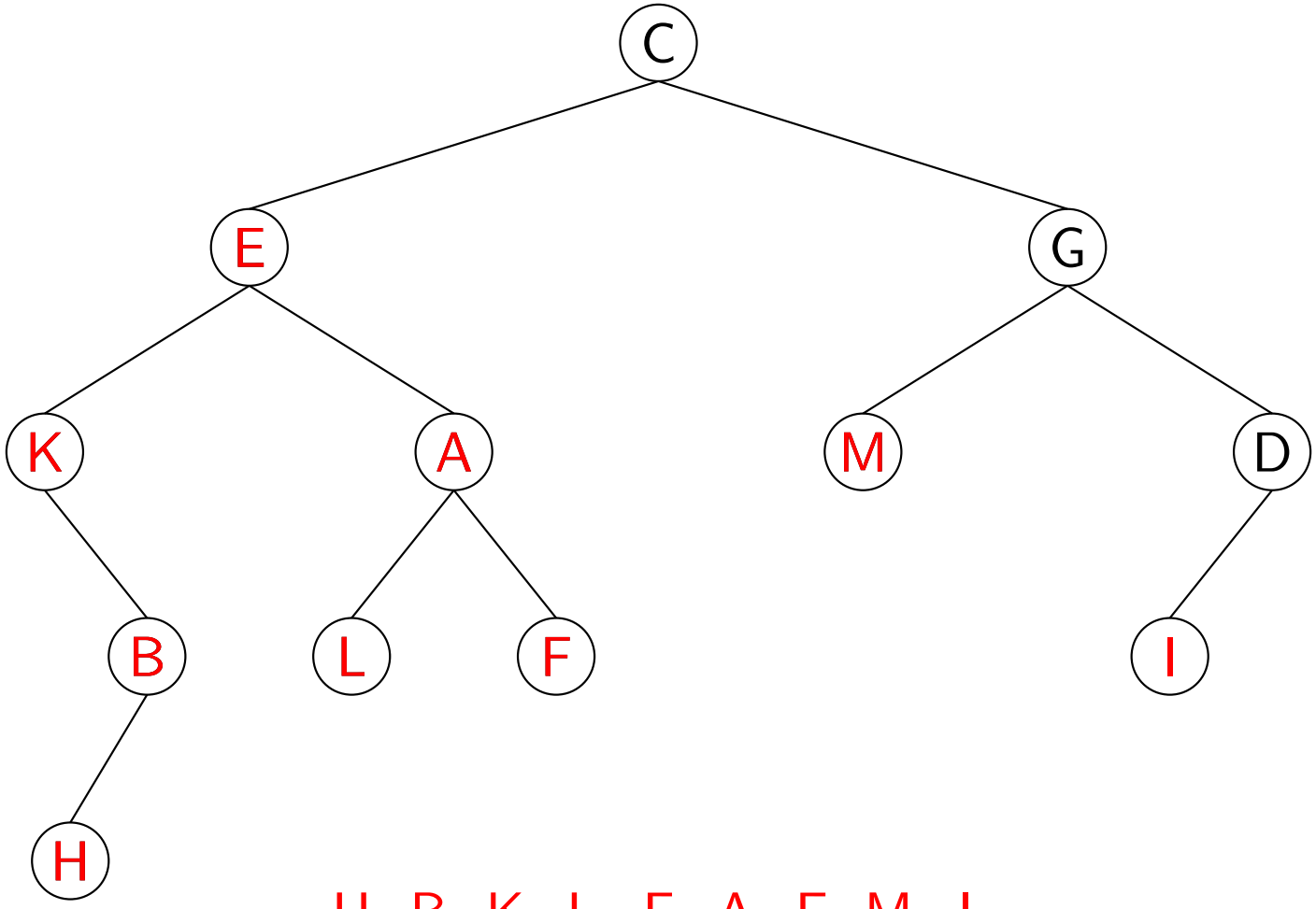
H B K L F A E

Parcurgere postordine - exemplu



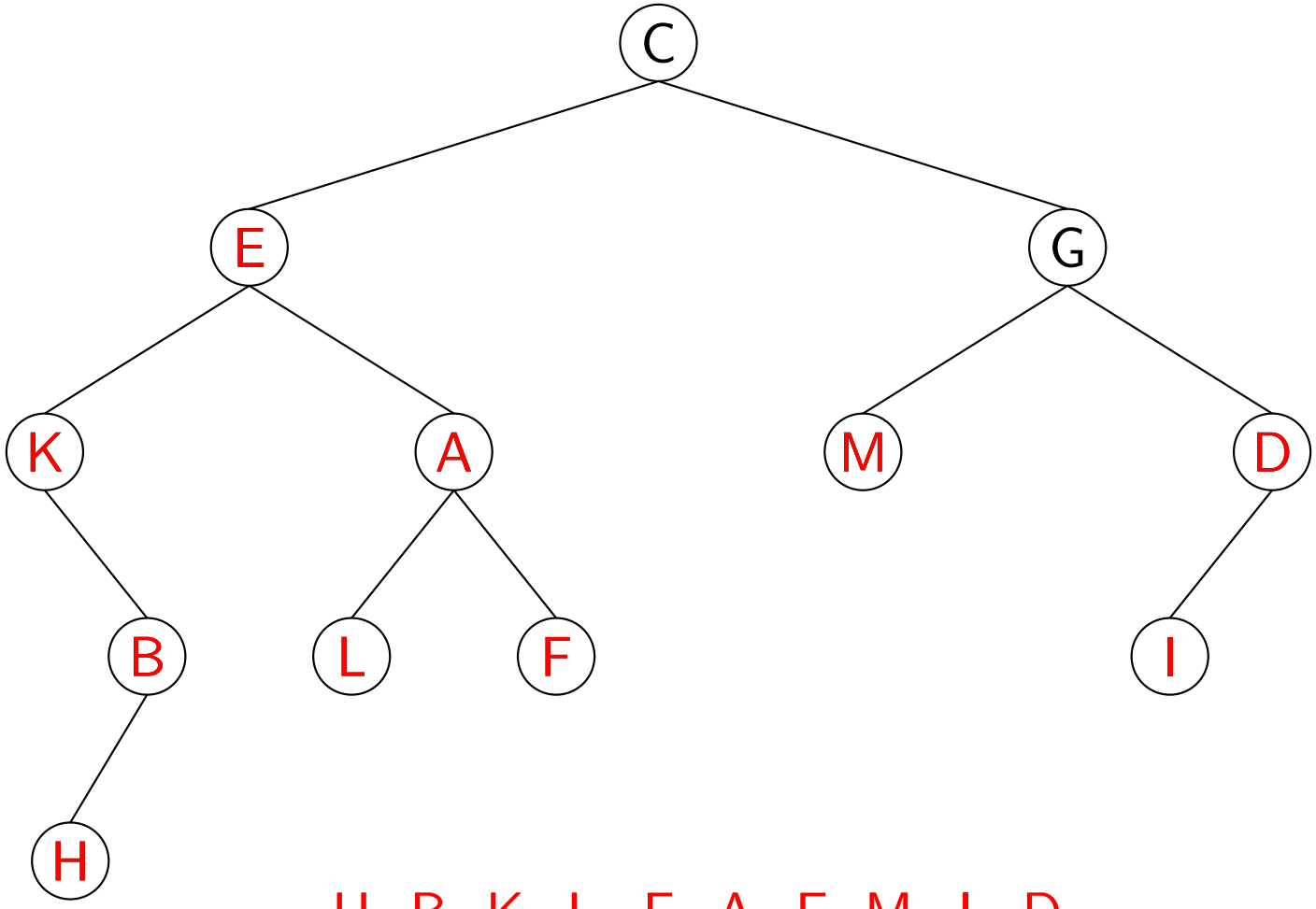
H B K L F A E M

Parcurgere postordine - exemplu



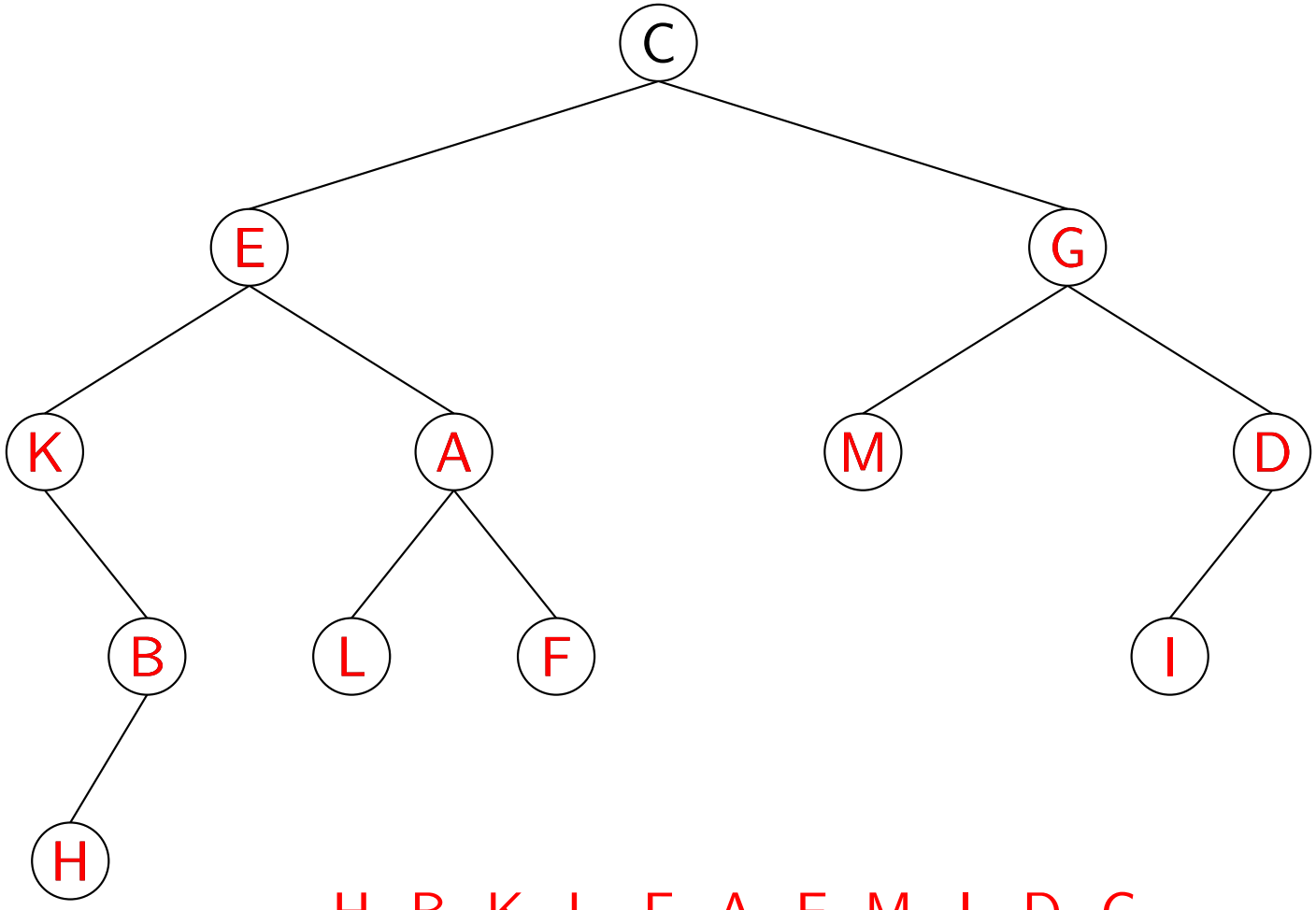
H B K L F A E M I

Parcurgere postordine - exemplu



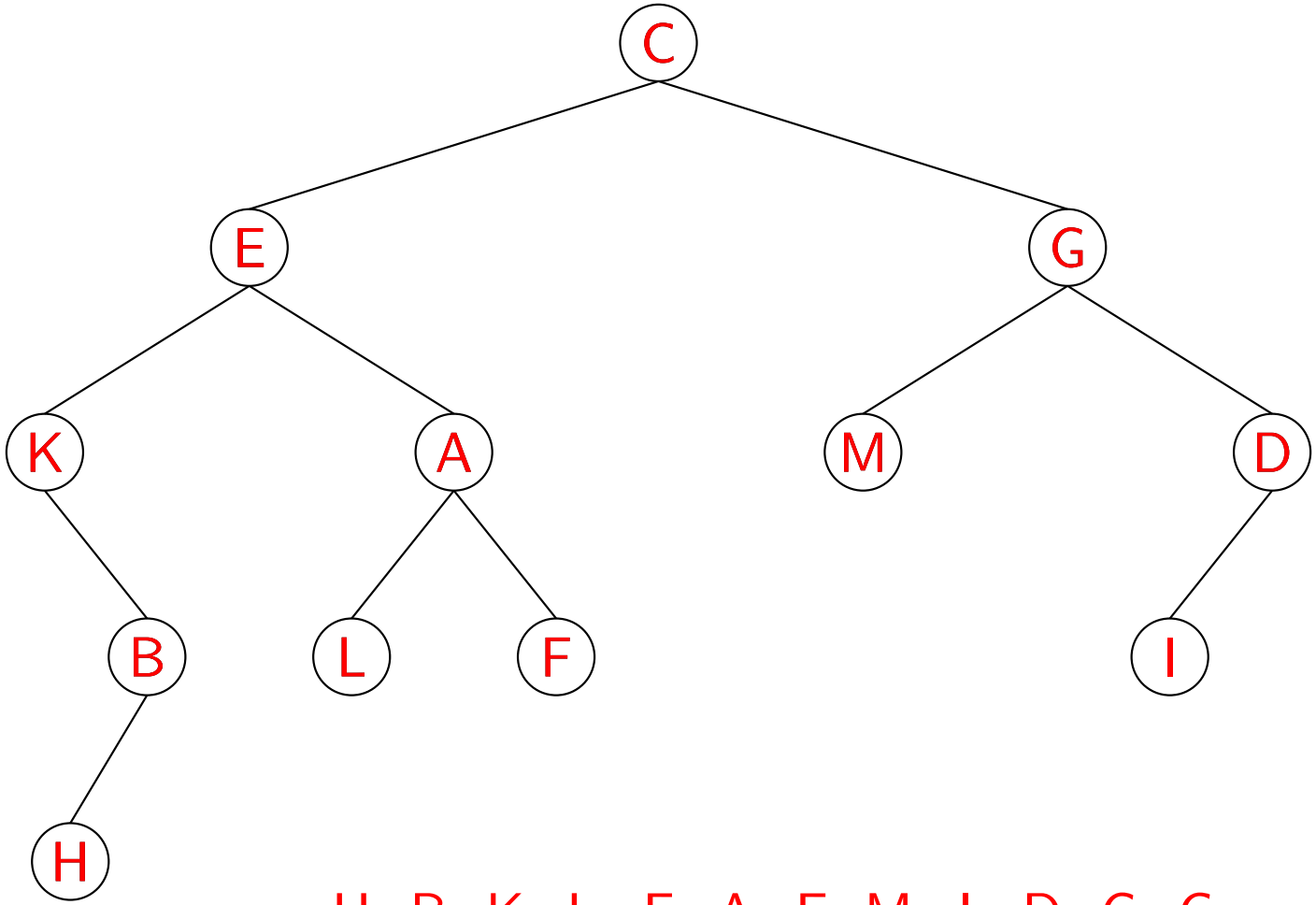
H B K L F A E M I D

Parcurgere postordine - exemplu



H B K L F A E M I D G

Parcurgere postordine - exemplu



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

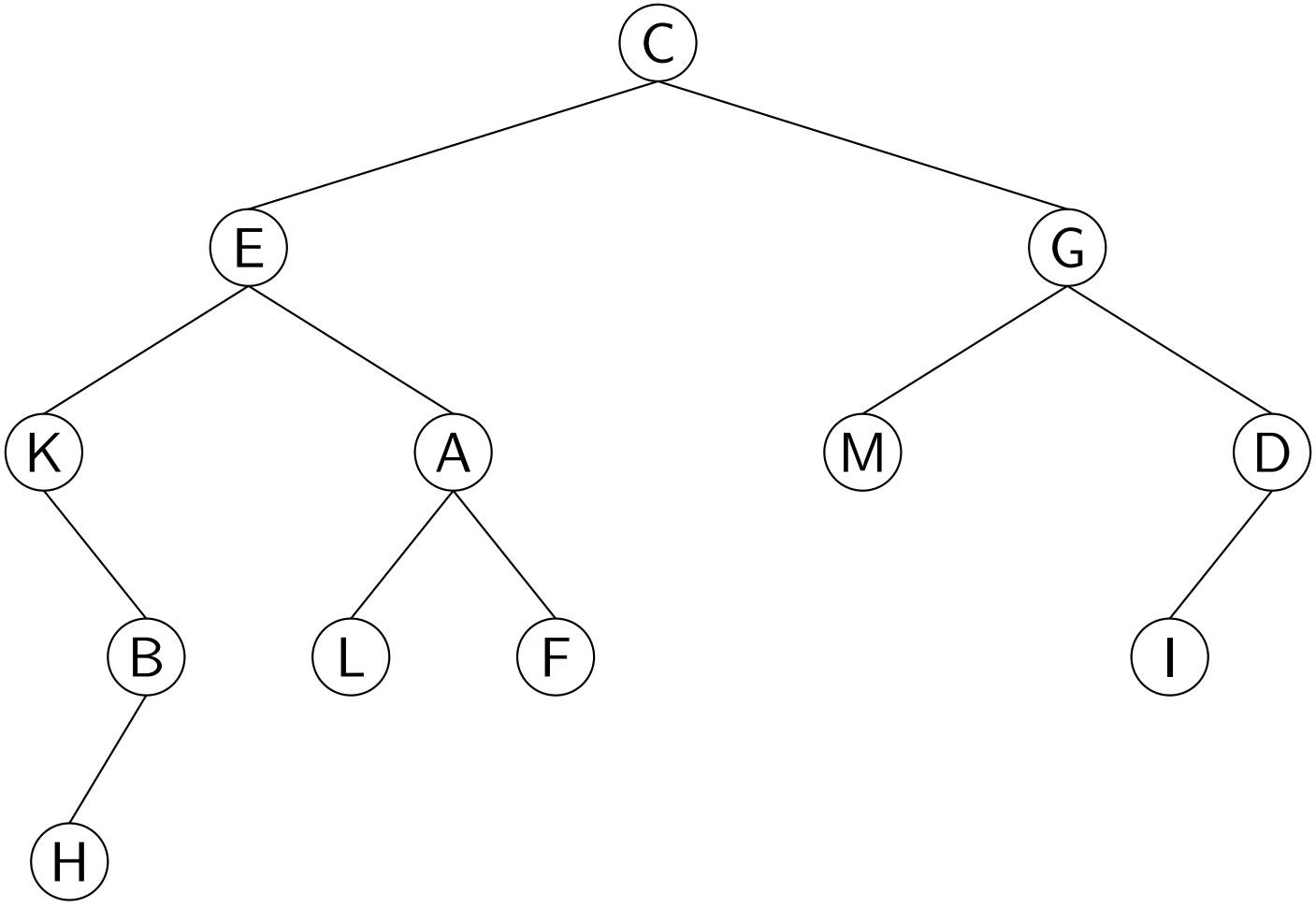
ArbBin – parcurgerea BFS (pe lățime)

`parcurgereBFS()`

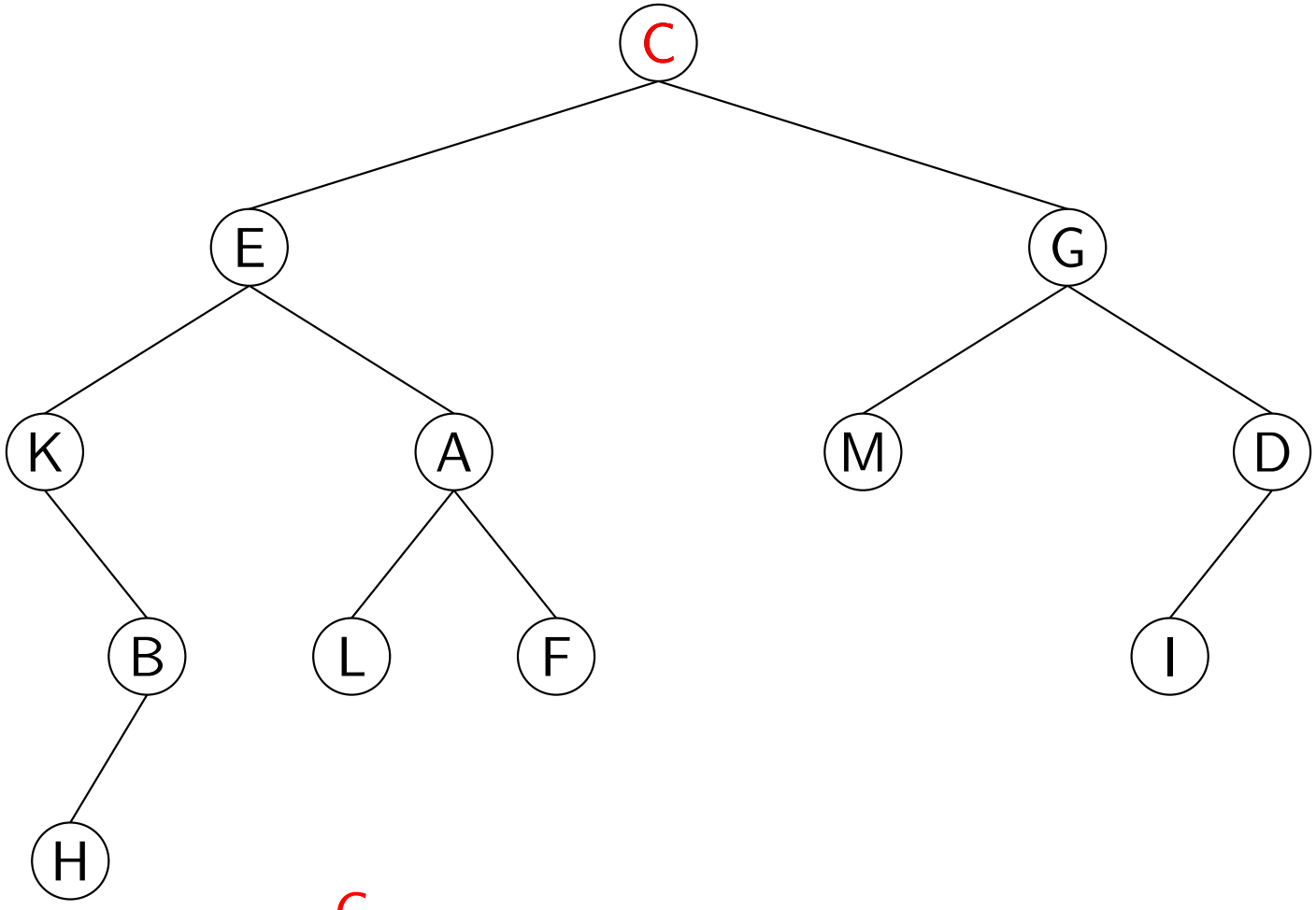
- intrare:
  - un arbore binar `t`;
  - o procedură `viziteaza()`.

- ieșire:
  - arborele `t`, dar cu nodurile procesate cu `viziteaza()` în ordinea BFS (pe lățime / pe niveluri).

Parcurgere BFS - exemplu

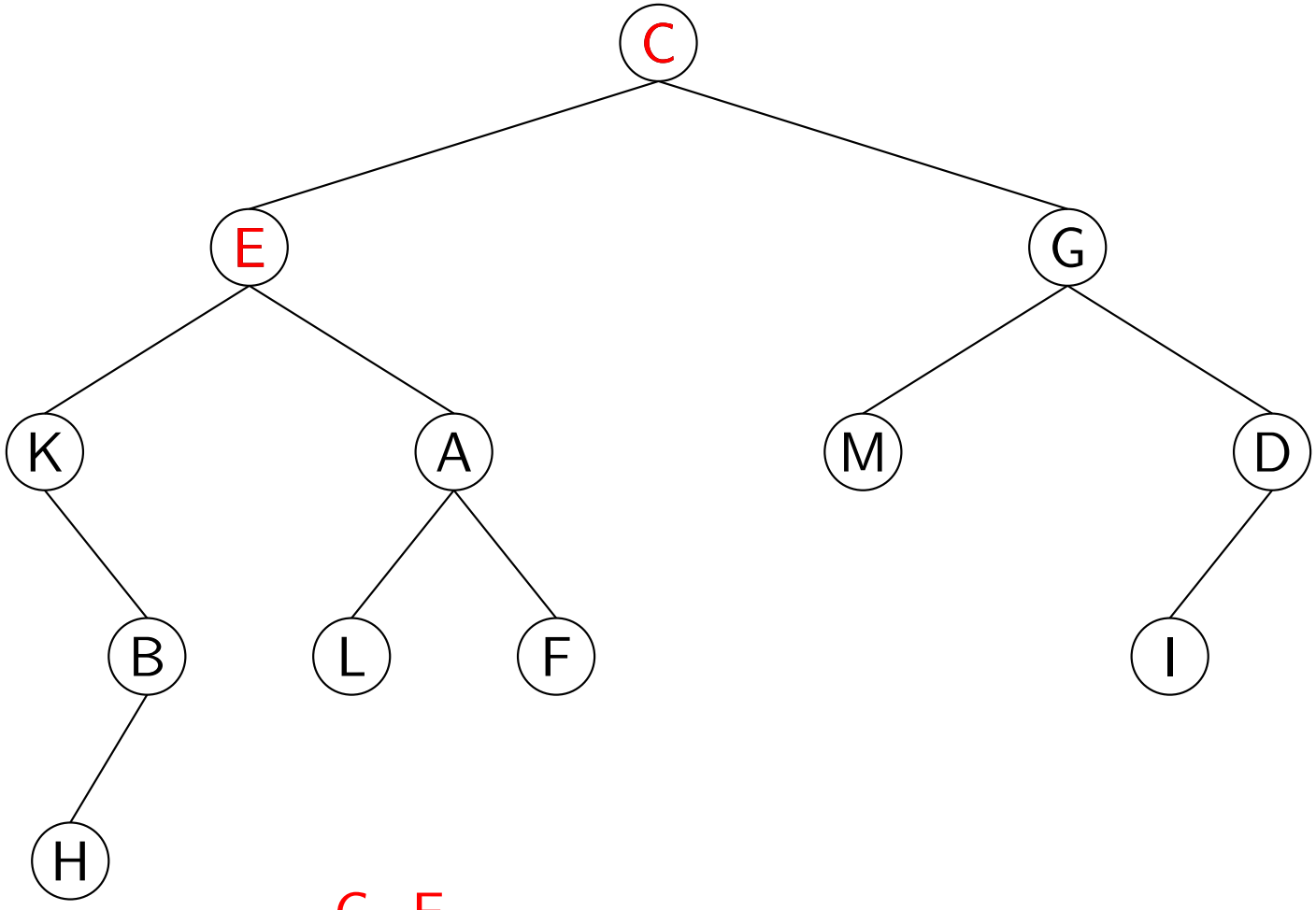


Parcurgere BFS - exemplu



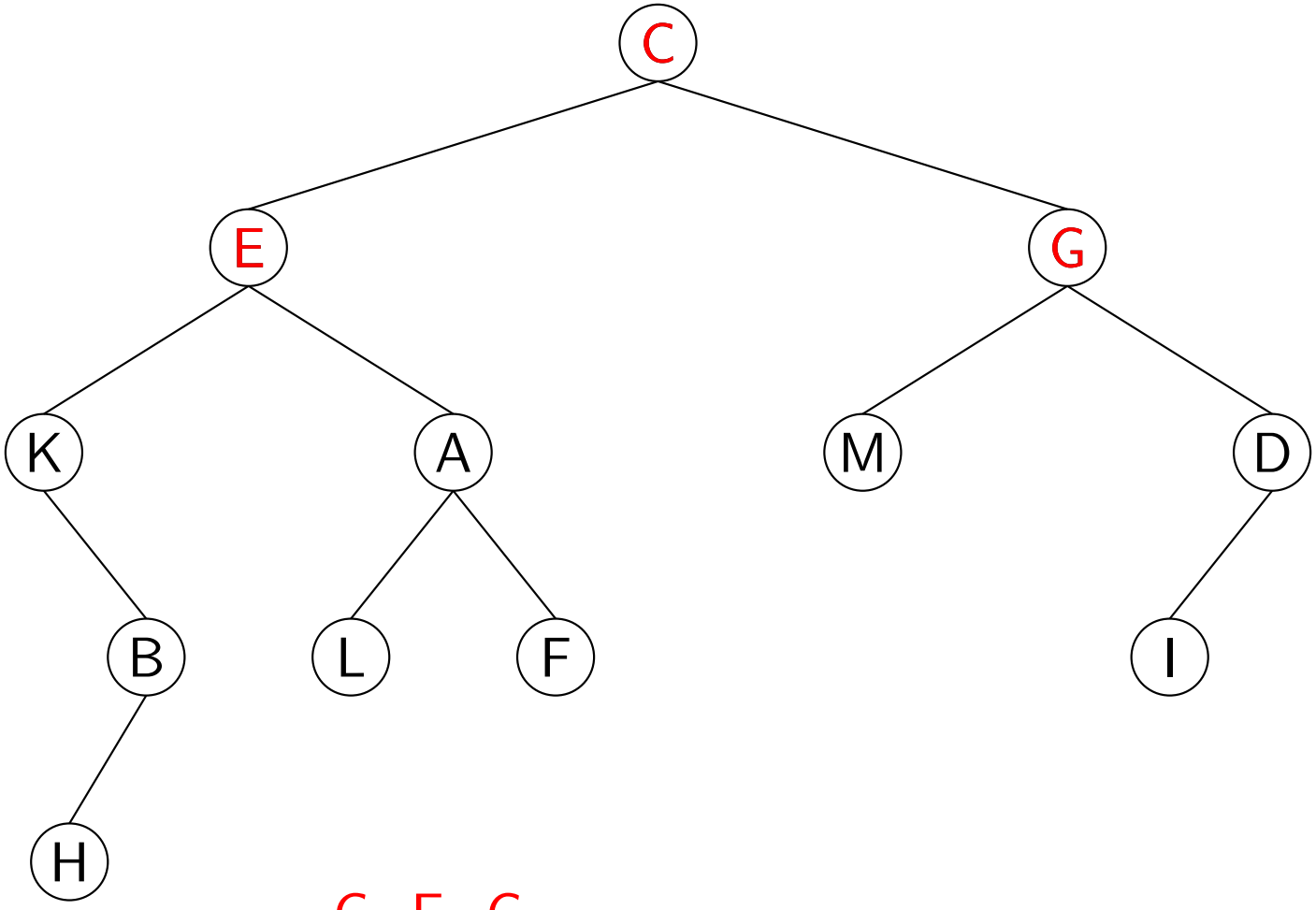
C

Parcurgere BFS - exemplu



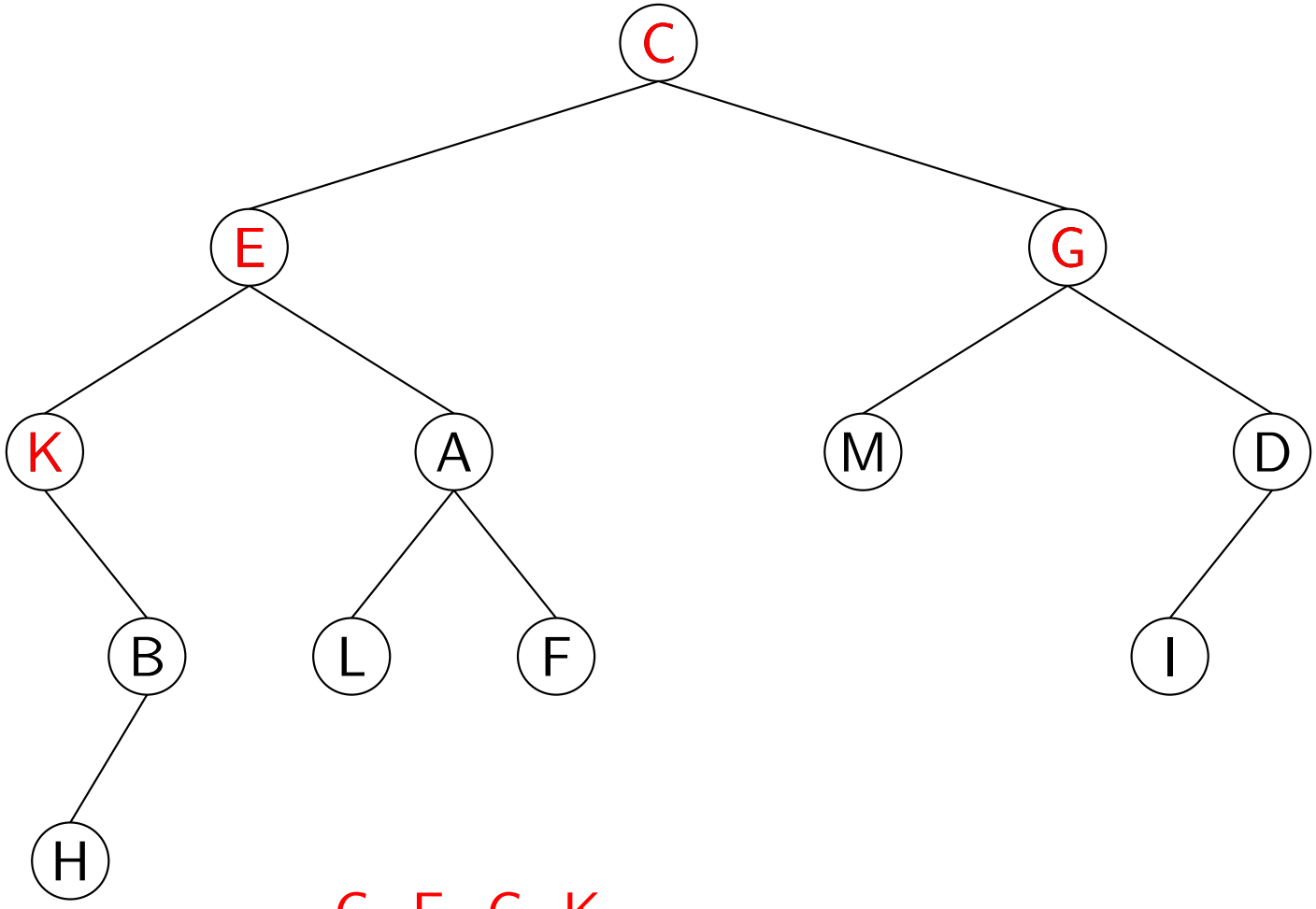
C E

Parcurgere BFS - exemplu



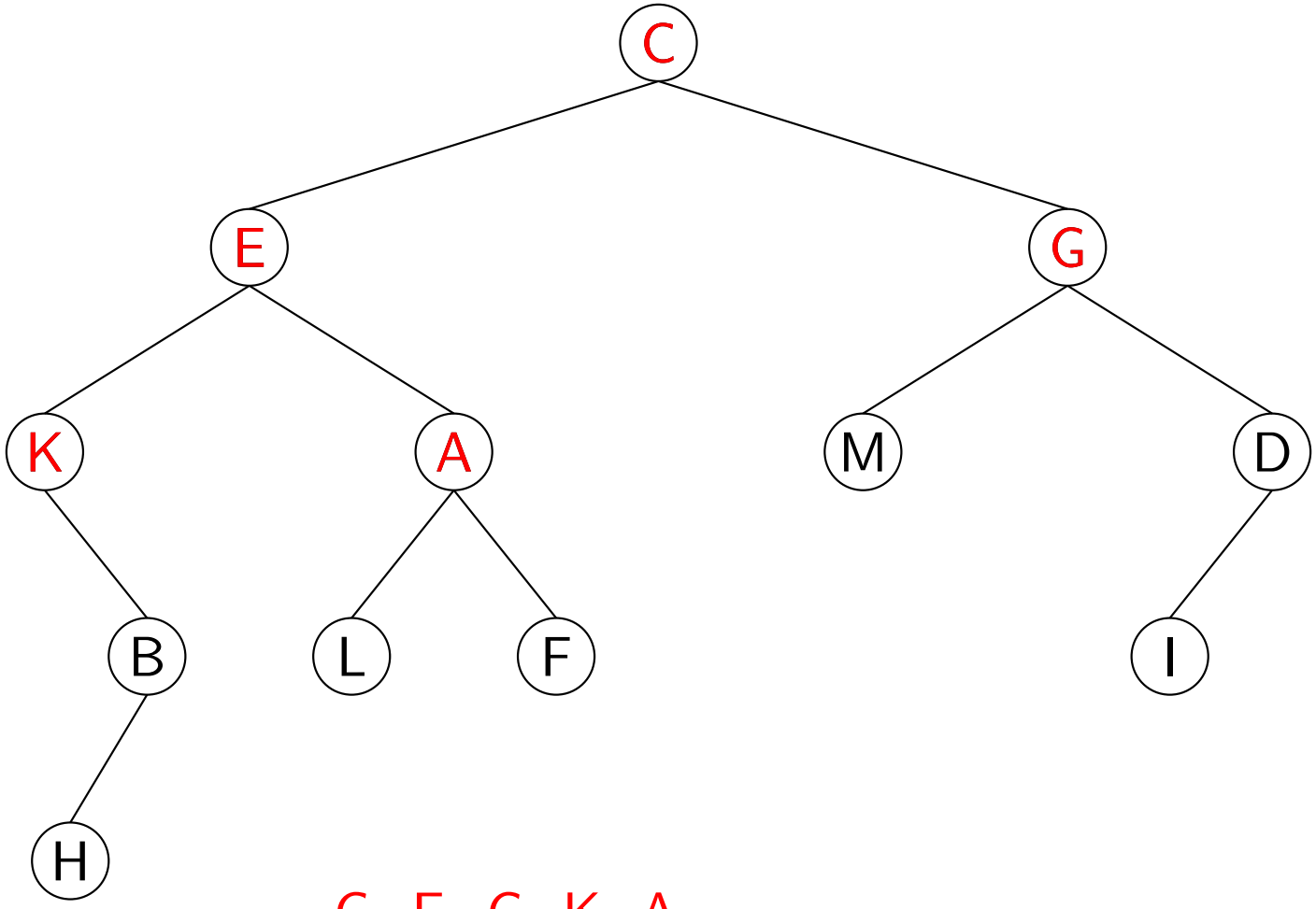
C E G

Parcurgere BFS - exemplu



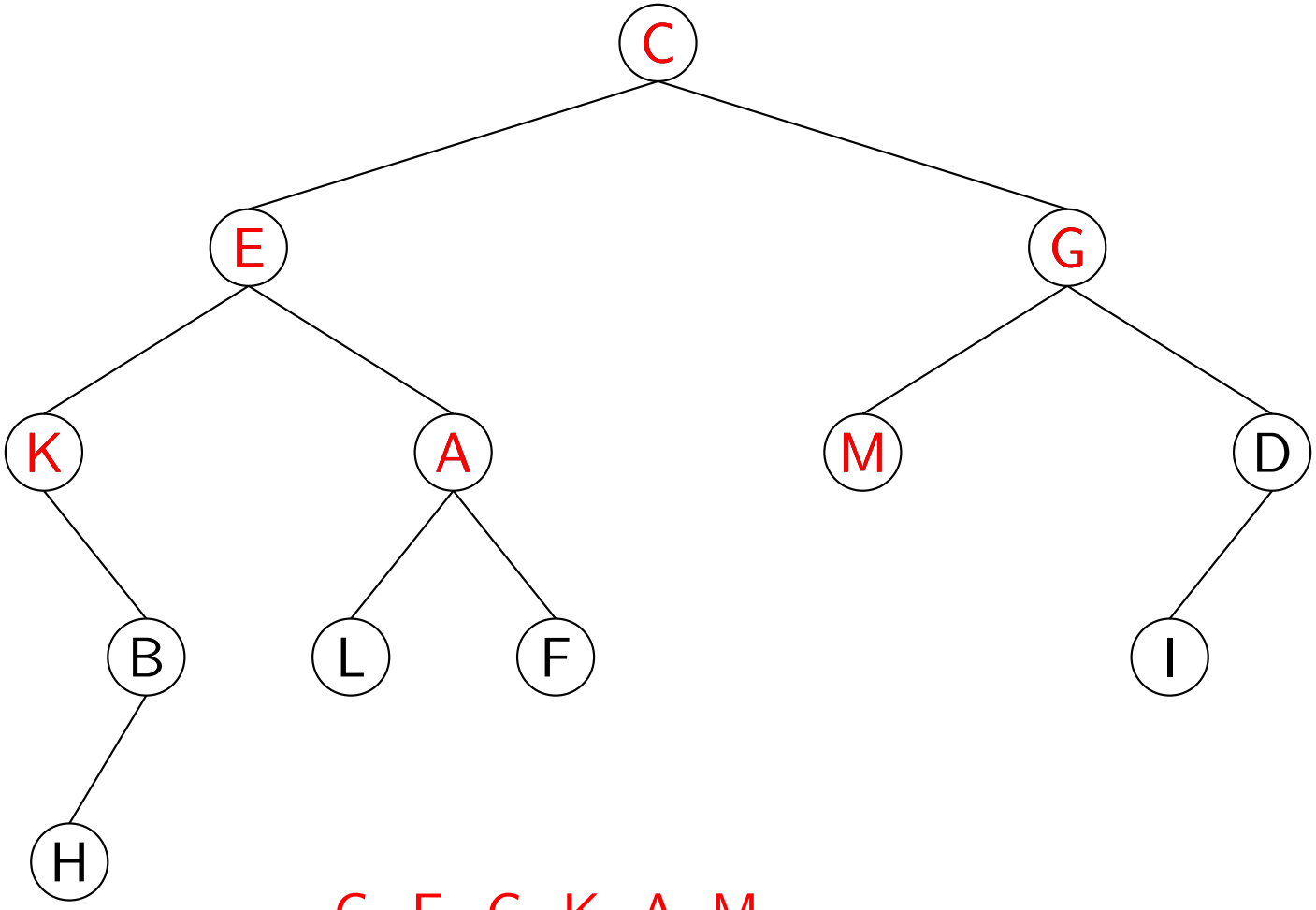
C E G K

Parcurgere BFS - exemplu



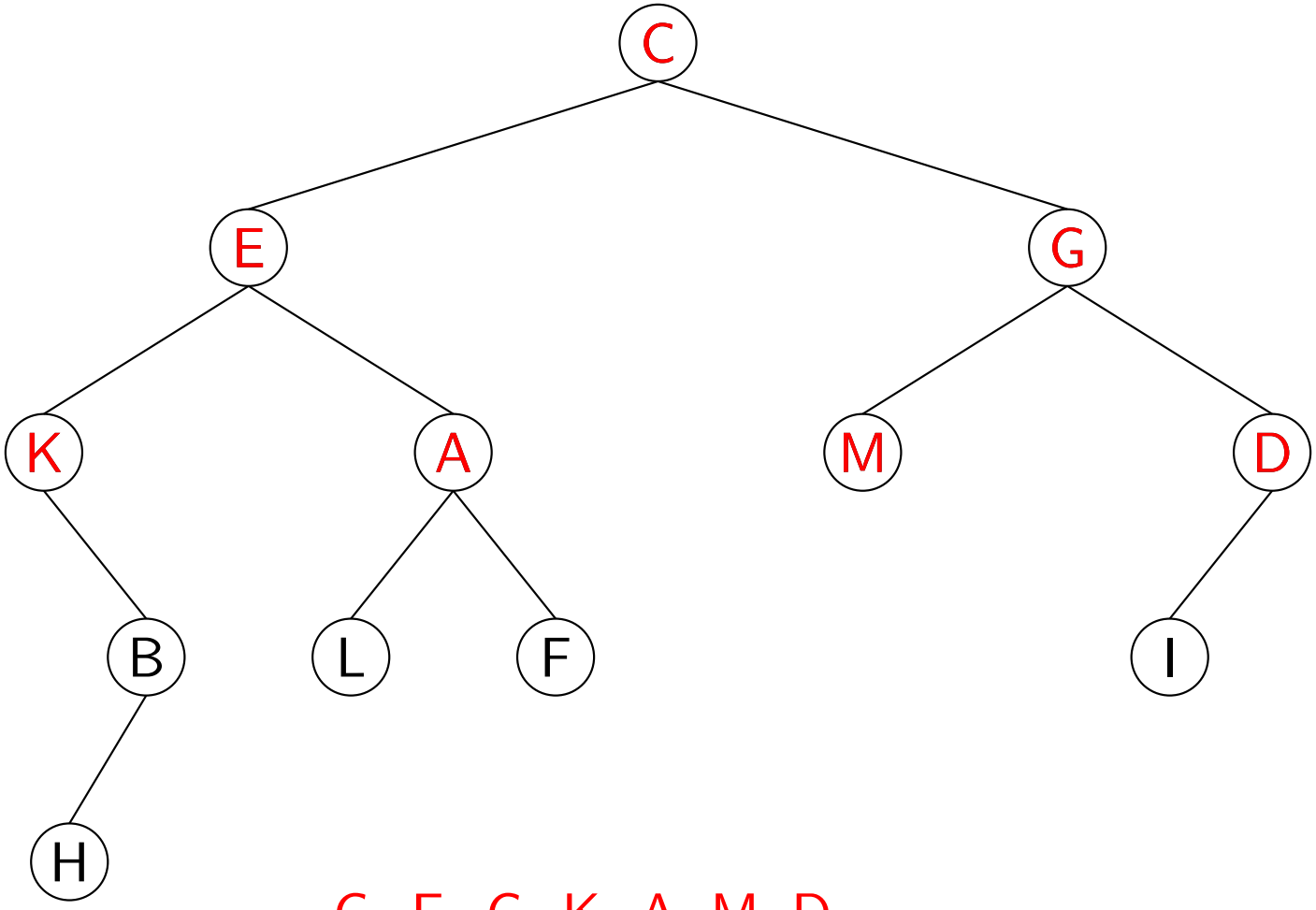
C E G K A

Parcurgere BFS - exemplu



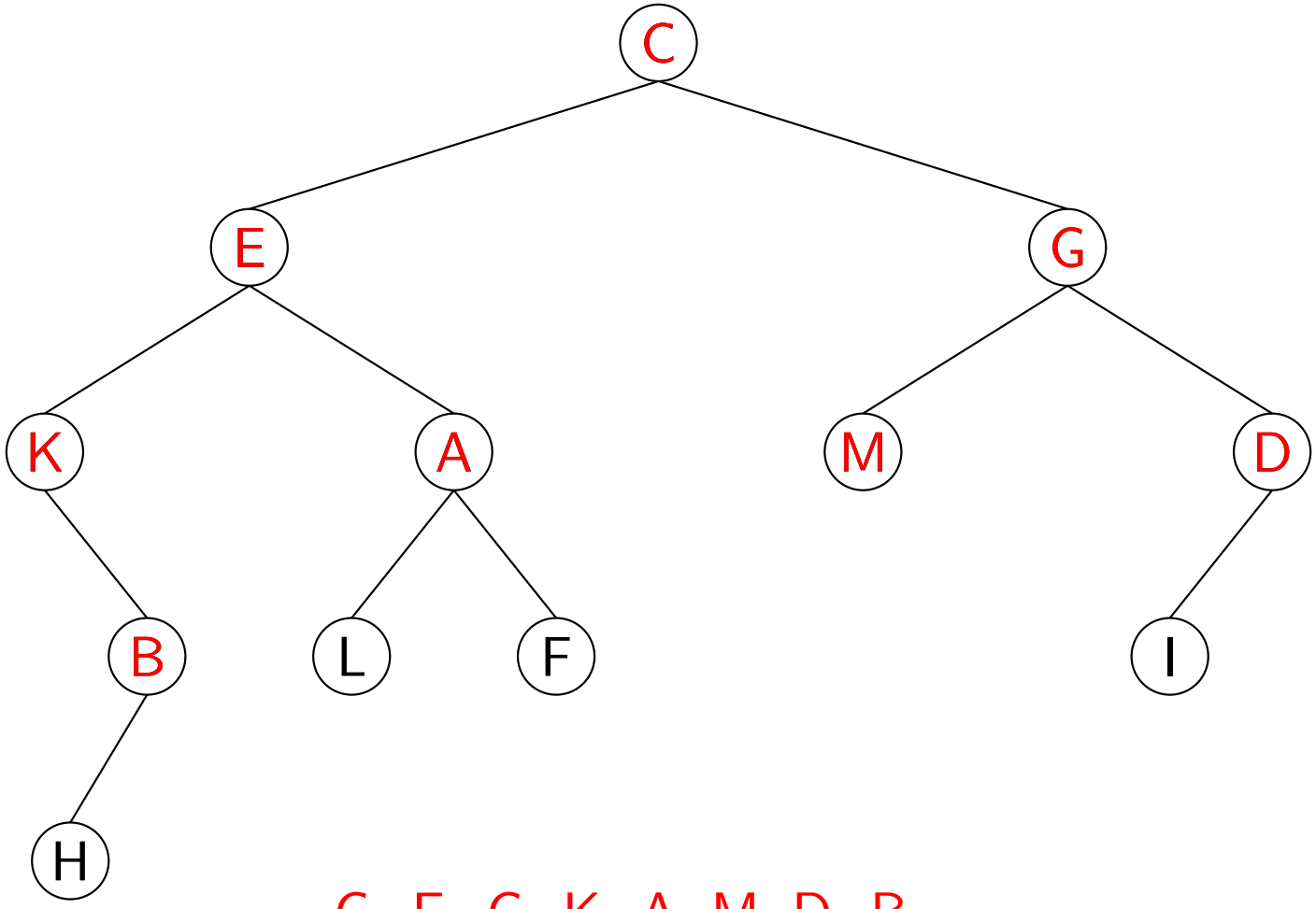
C E G K A M

Parcurgere BFS - exemplu



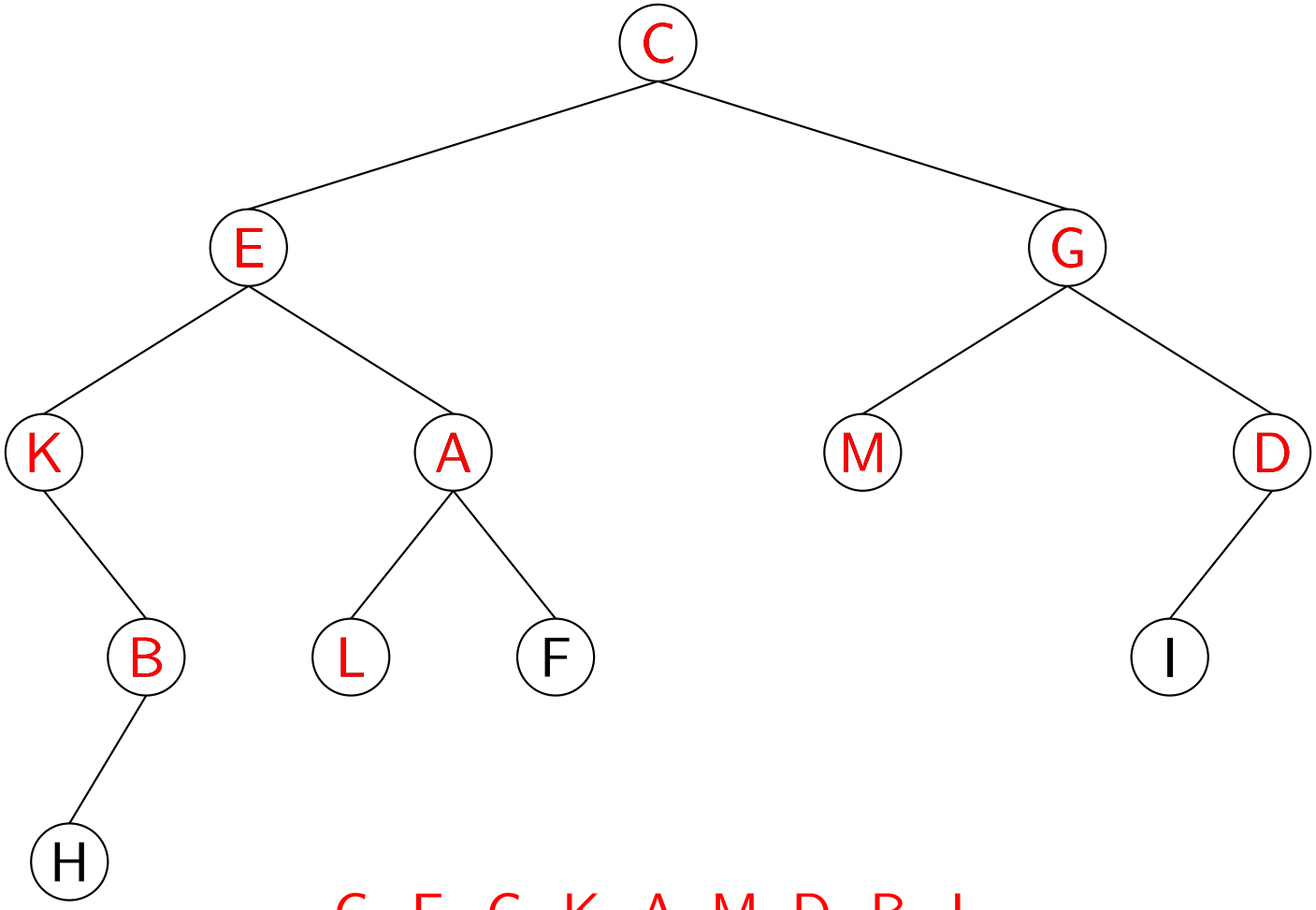
C E G K A M D

Parcurgere BFS - exemplu



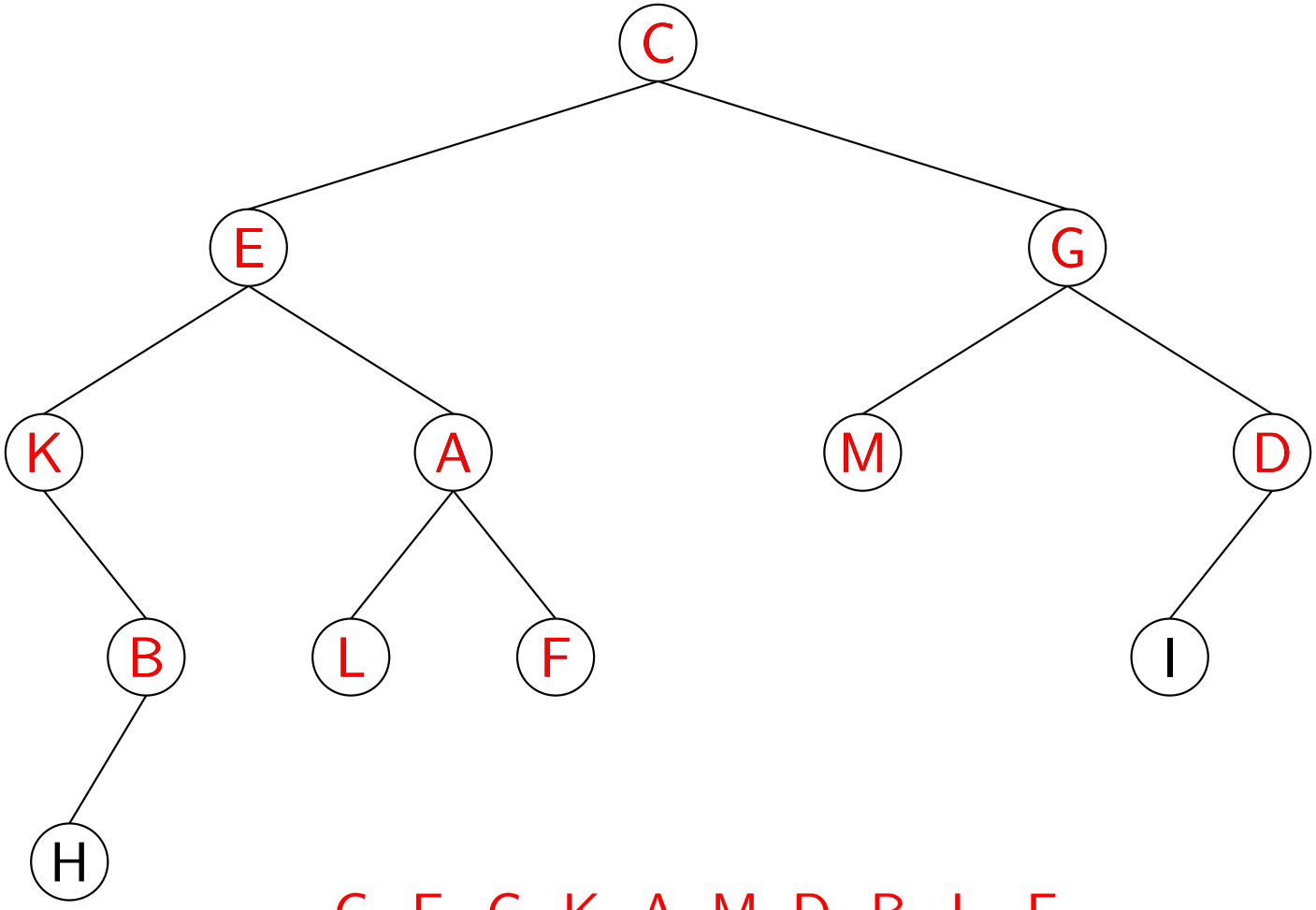
C E G K A M D B

Parcurgere BFS - exemplu



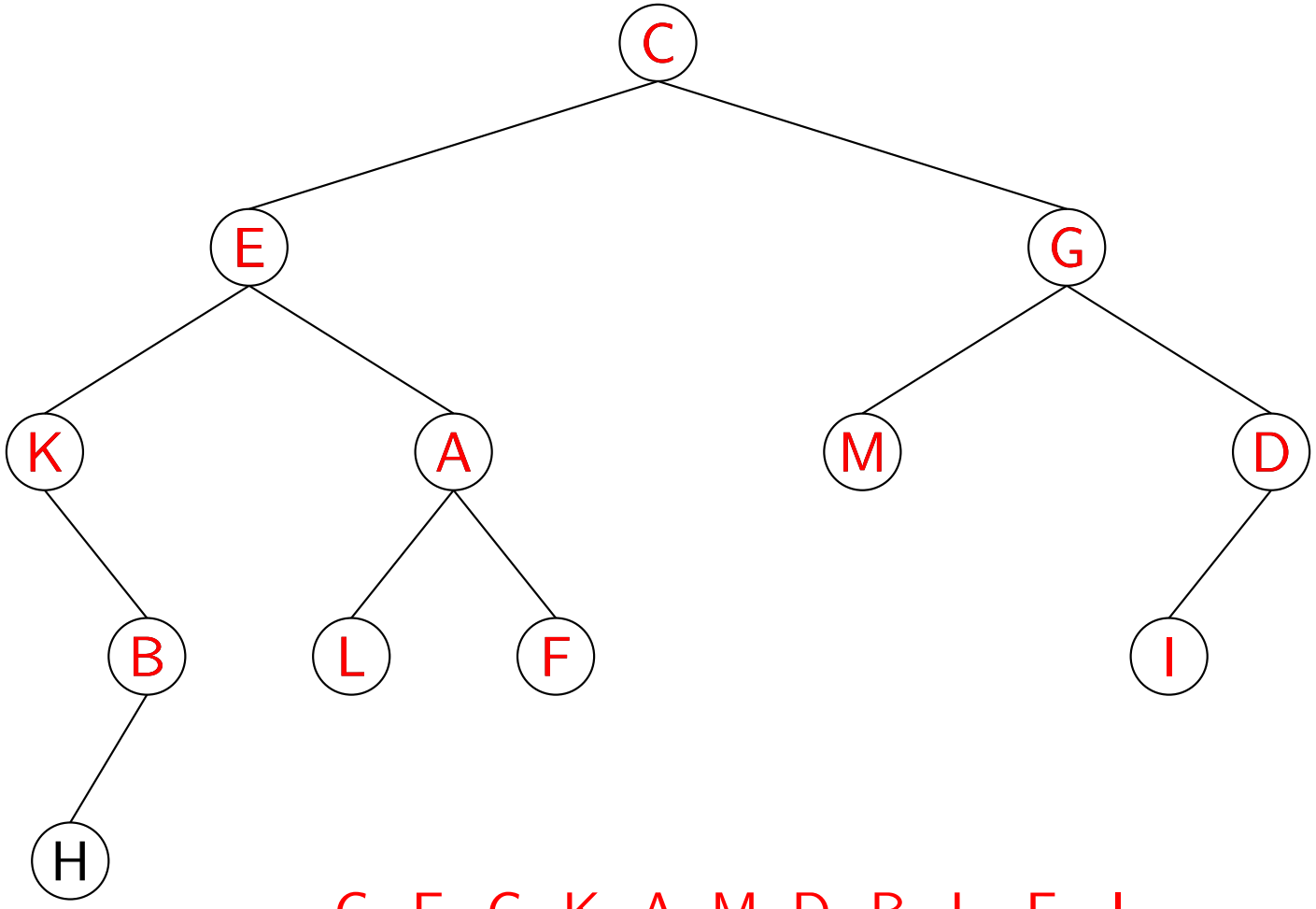
C E G K A M D B L

Parcurgere BFS - exemplu



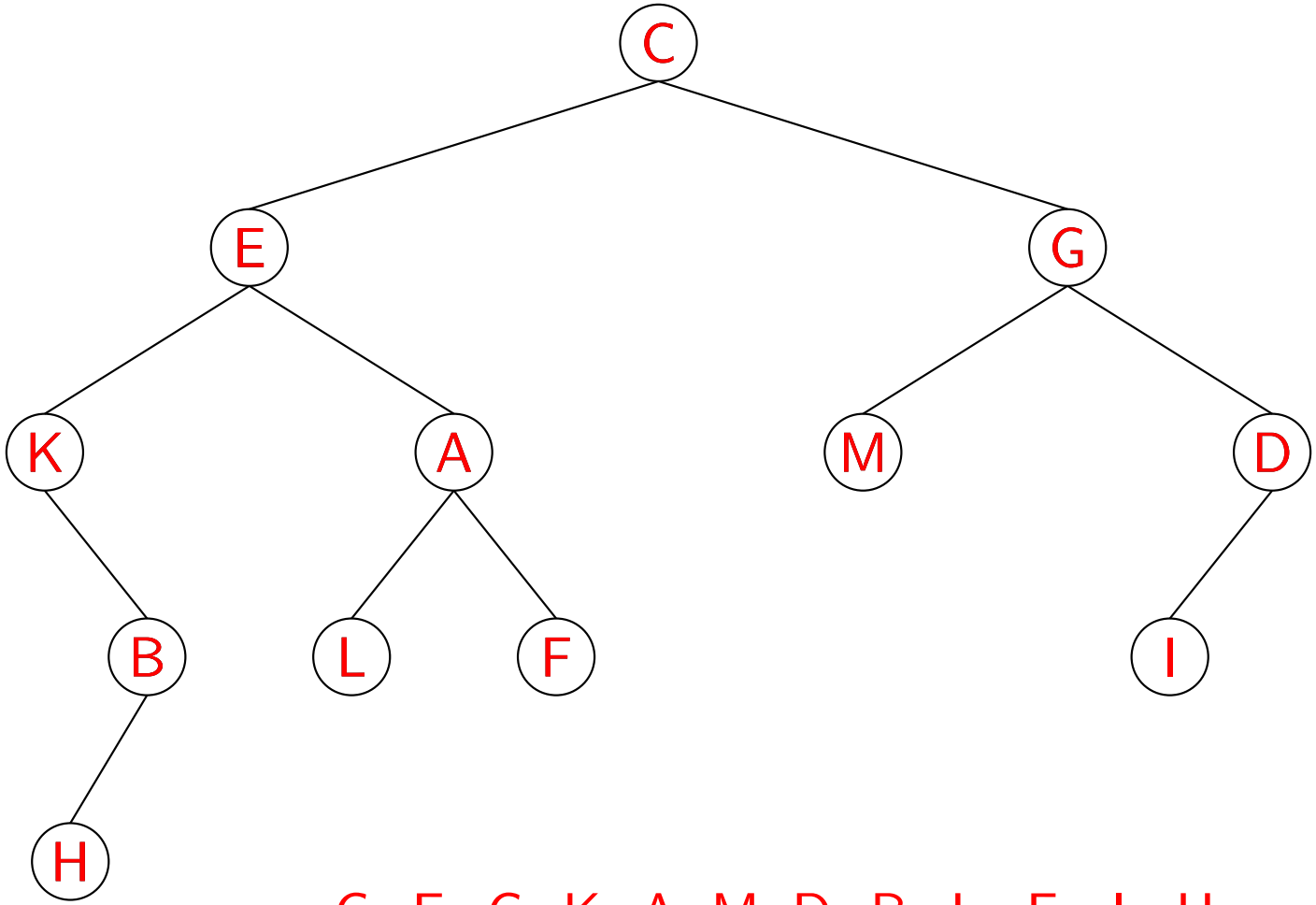
C E G K A M D B L F

Parcurgere BFS - exemplu



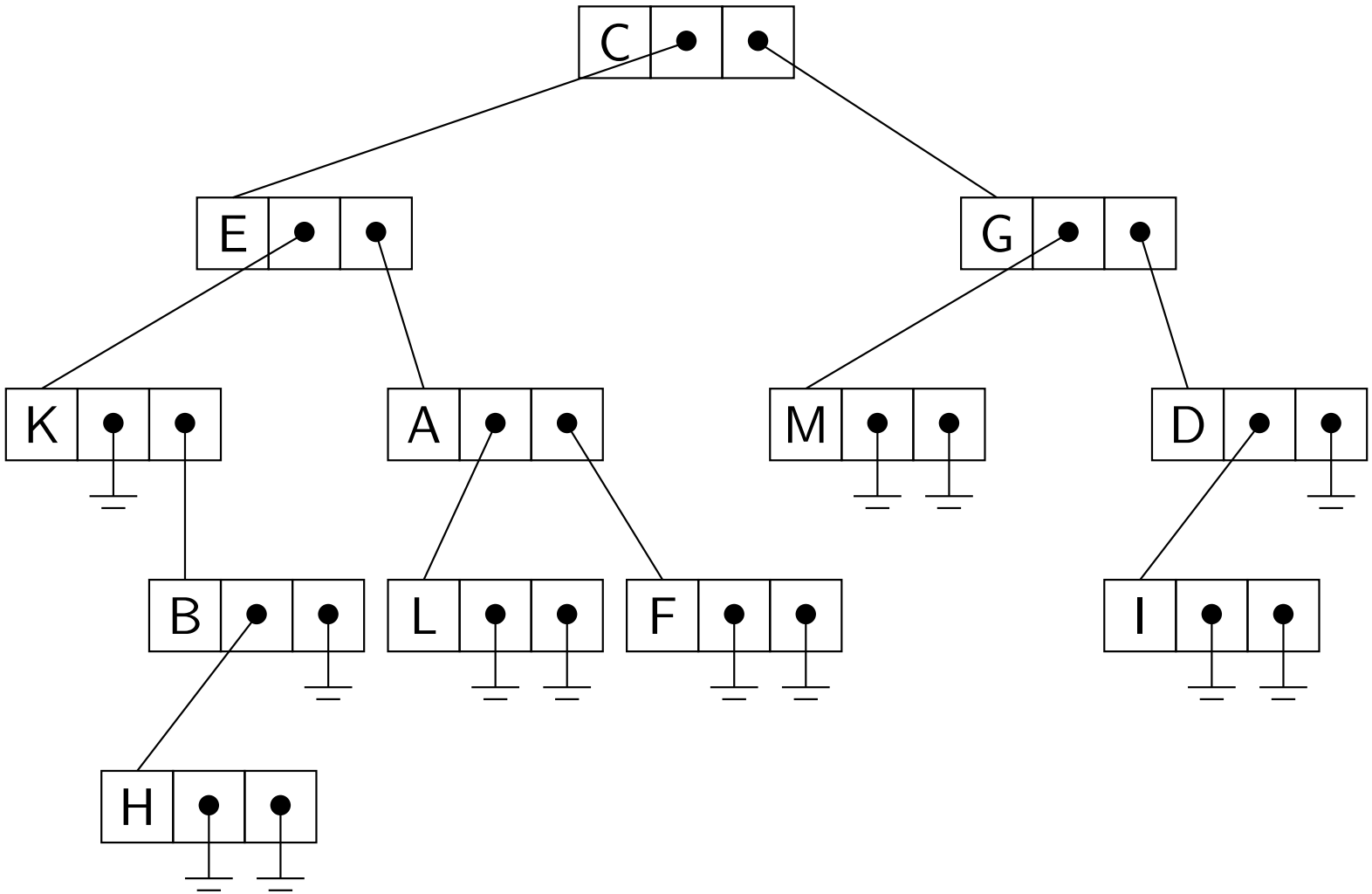
C E G K A M D B L F I

Parcurgere BFS - exemplu



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

ArbBin: implementarea cu structuri înlănțuite



ArbBin: structura unui nod

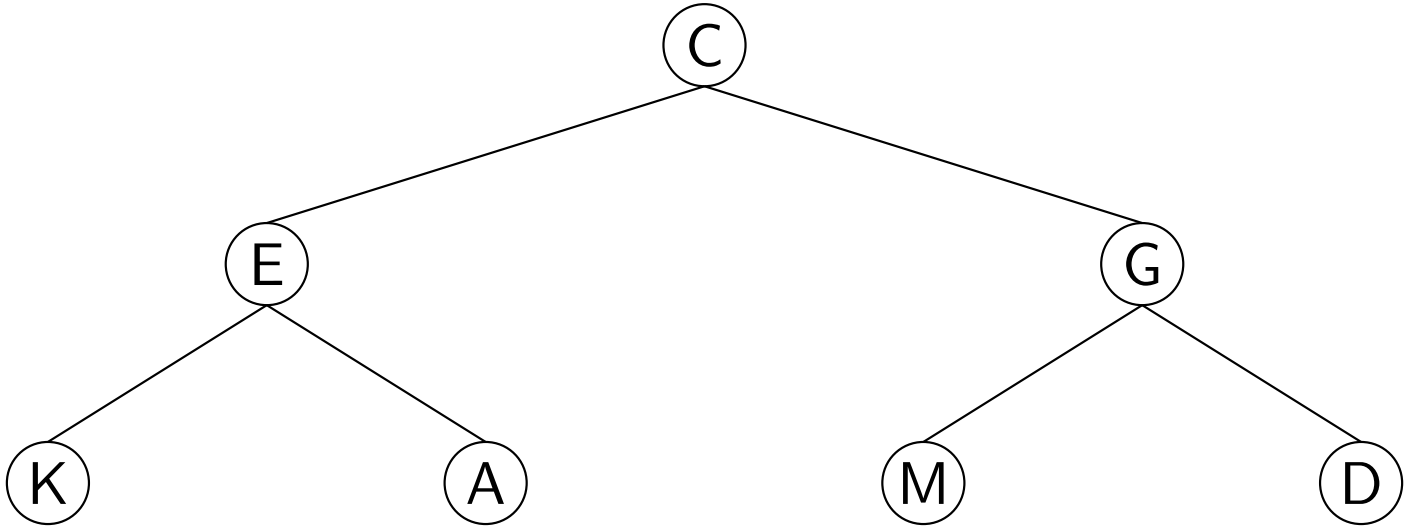
Un nod `v` (aflat la adresa de memorie `v`) este o structură cu trei câmpuri:

- `v->inf` – informația memorată în nod;
- `v->stg` – adresa fiului stânga;
- `v->drp` – adresa fiului dreapta.

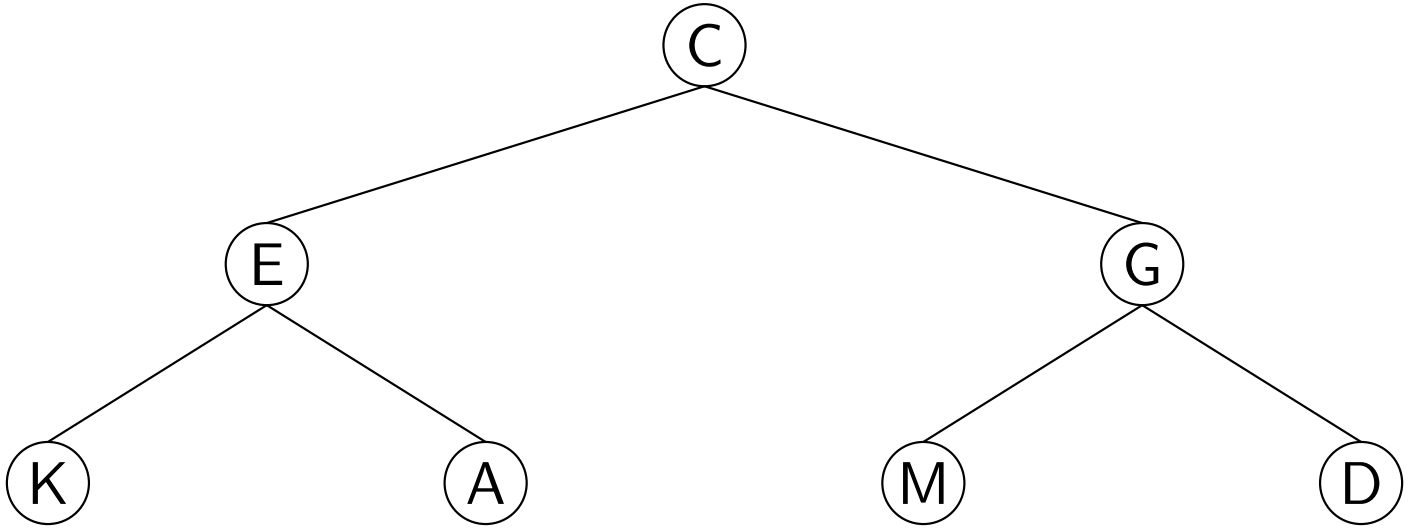
ArbBin: parcurgePreordine()

```
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



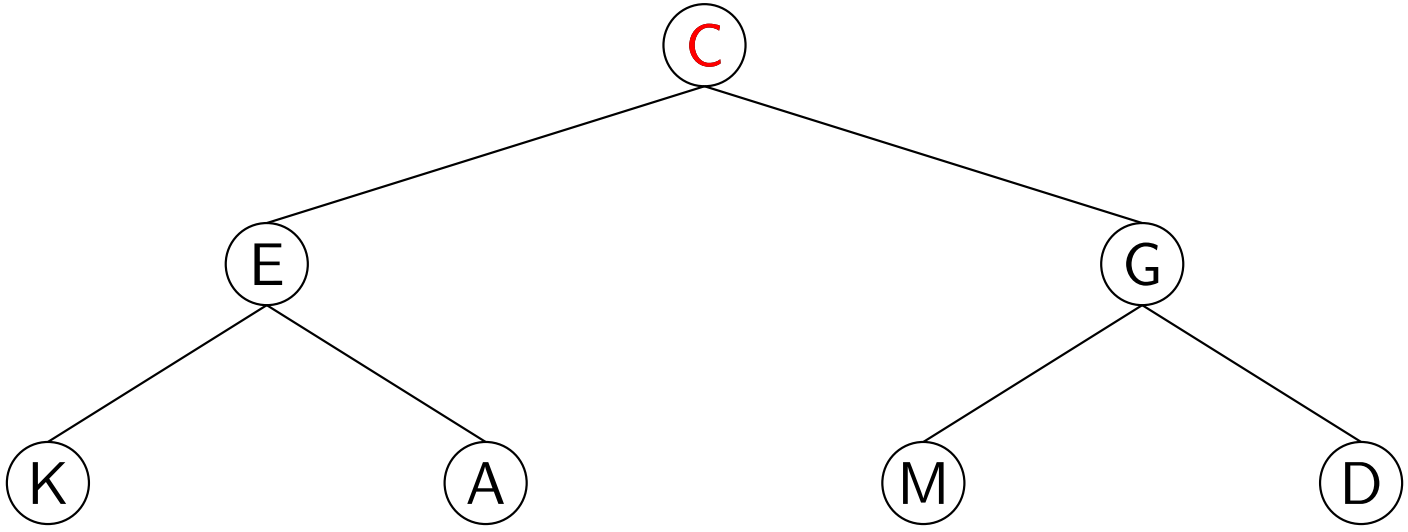
Implementarea parcurgerii BFS



BFS =

Coada = ( )

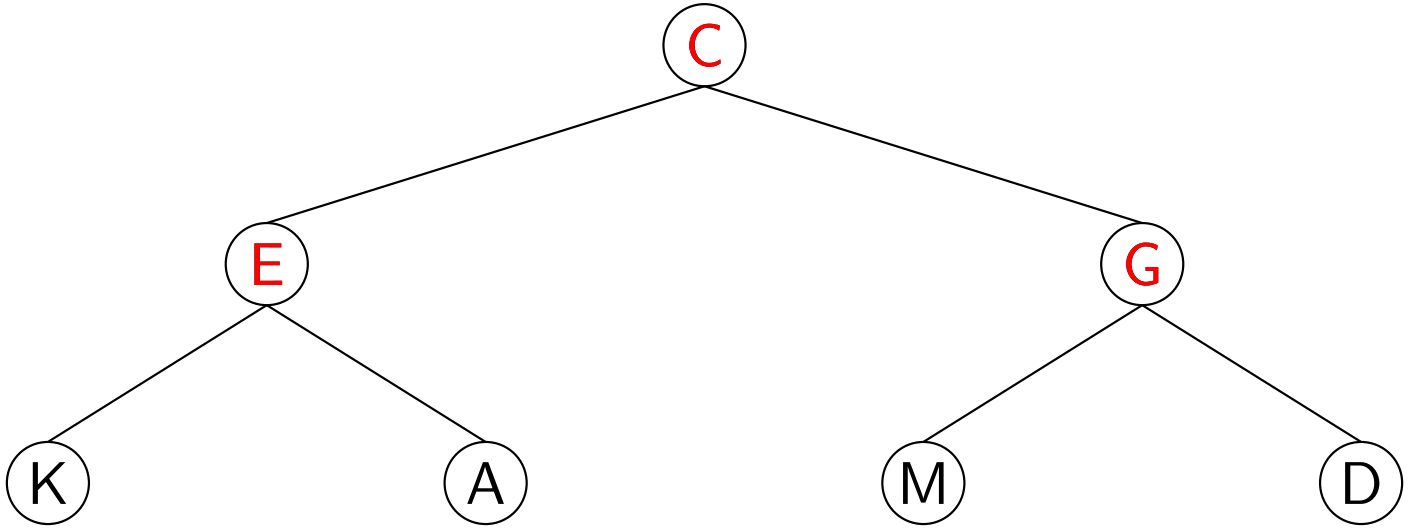
Implementarea parcurgerii BFS



BFS =

Coada = (C)

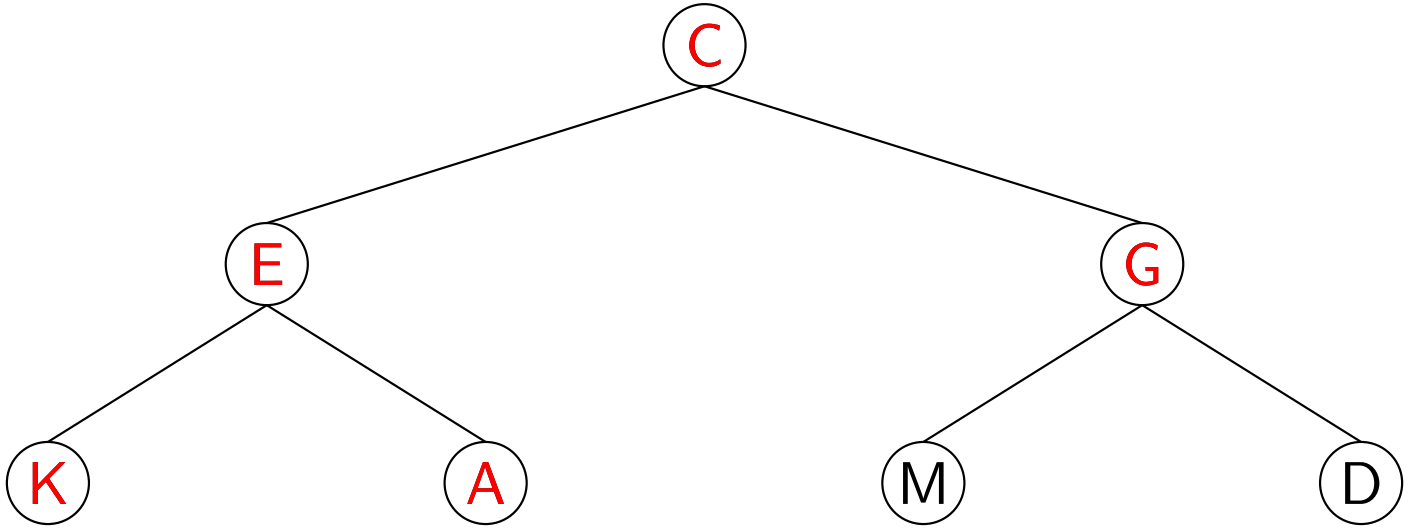
Implementarea parcurgerii BFS



BFS = C

Coada = (E G)

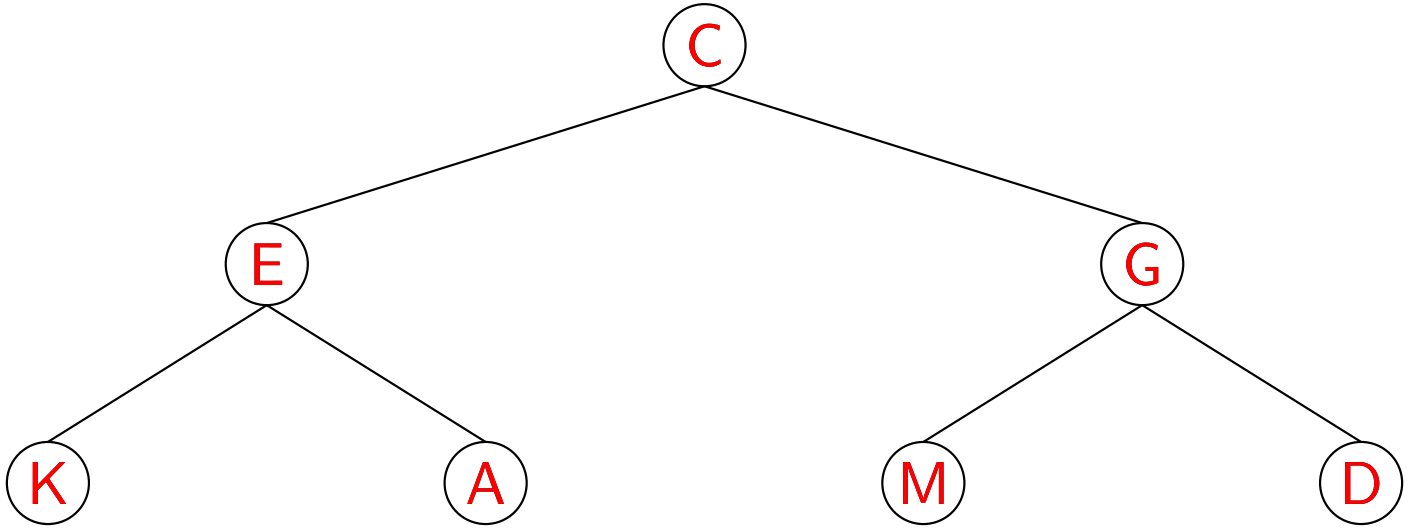
Implementarea parcurgerii BFS



BFS = C E

Coada = (G K A)

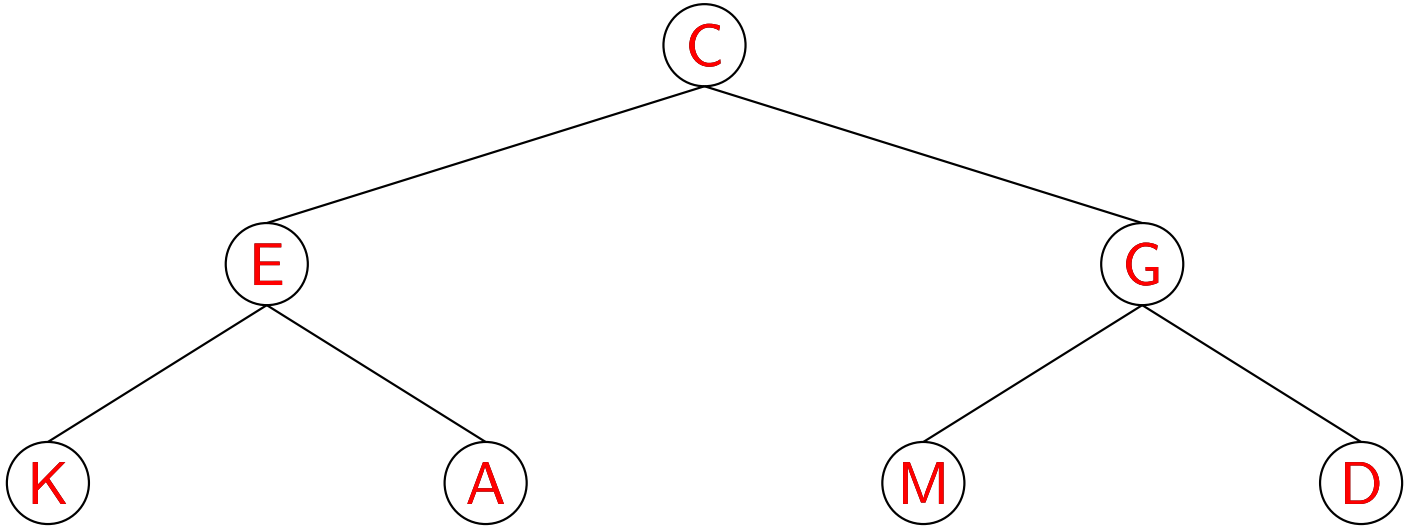
Implementarea parcurgerii BFS



BFS = C E G

Coada = (K A M D)

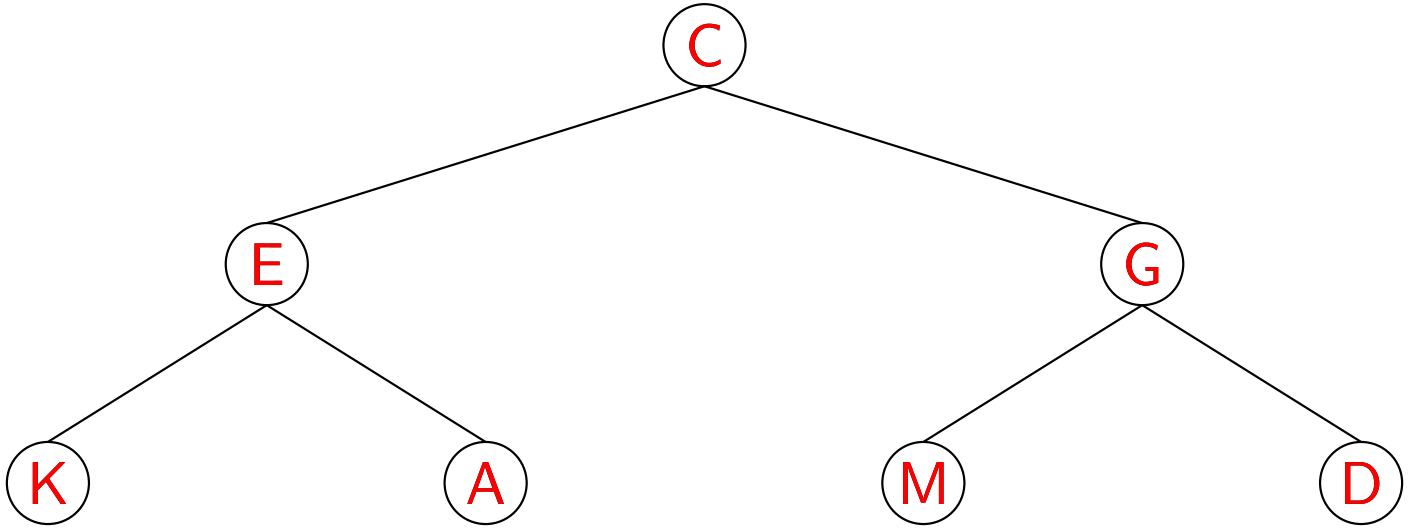
Implementarea parcurgerii BFS



BFS = C E G K

Coada = (A M D)

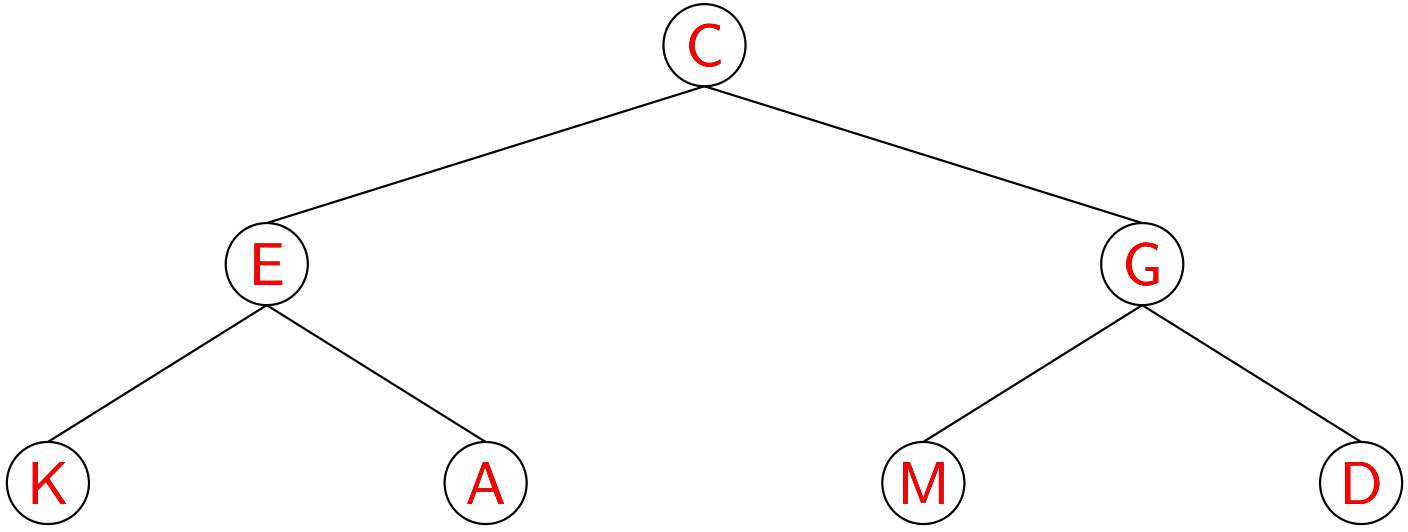
Implementarea parcurgerii BFS



BFS = C E G K A

Coada = (M D)

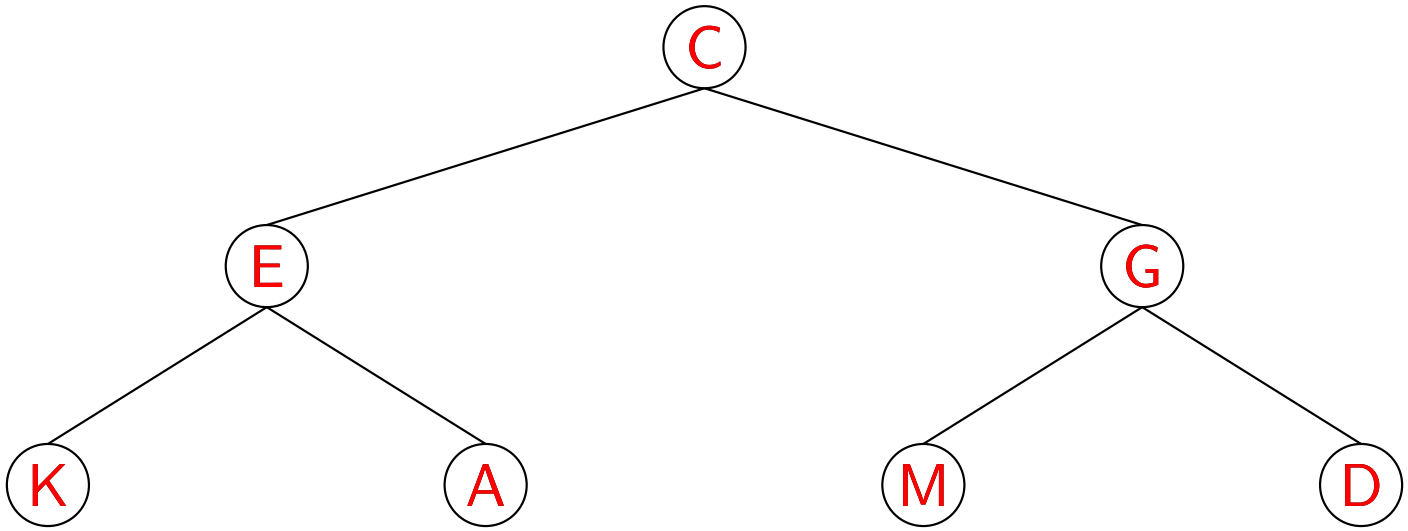
Implementarea parcurgerii BFS



BFS = C E G K A M

Coada = (D)

Implementarea parcurgerii BFS



BFS = C E G K A M D

Coada = ( )

Implementarea parcurgerii BFS

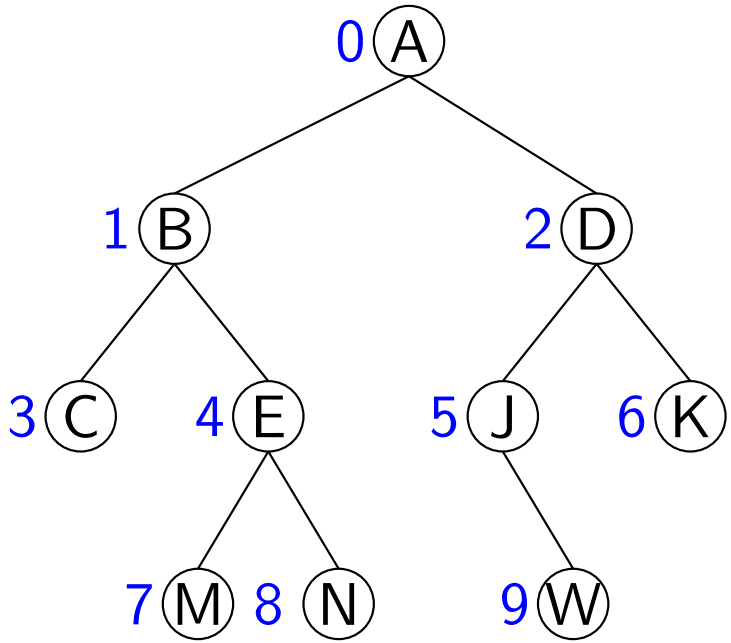
```
procedure parcurgeBFS(t, viziteaza)
begin
    if (t == NULL) then
        return
    else
        Coada ← coadaVida()
        insereaza(Coada, t)
        while not esteVida(Coada) do
            citeste(Coada, v)
            viziteaza(v)
            if (v->stg != NULL) then
                insereaza(Coada,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

ArbBin: implementarea cu liste

- tablou de părinți: reprezentarea relației "părinte".

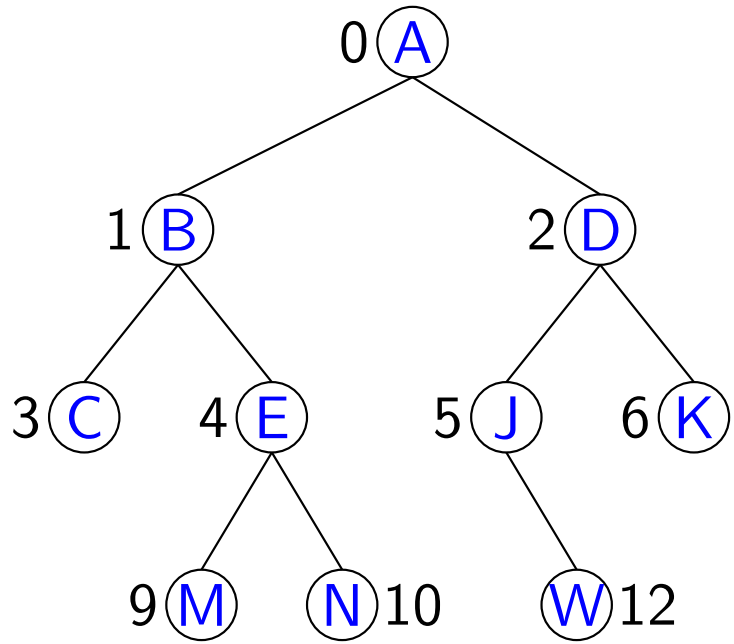
-1	0	0	1	1	2	2	4	4	5
0	1	2	3	4	5	6	7	8	9

- Avantaje:
  - simplitate;
  - acces ușor de la un nod spre rădăcină;
  - economie de memorie.
- Inconveniente:
  - acces dificil de la rădăcină spre noduri.



ArbBin: implementarea cu tablouri

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

Conținut

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

```
<int> ::= ... -2 | -1 | 0 | 1 | 2 ...
<op_bin> ::= + | - | * | / | %
<exr_int> ::= <int>
            | (<exp_int>)
            | <exp_int> <op_bin> <exp_int>
```

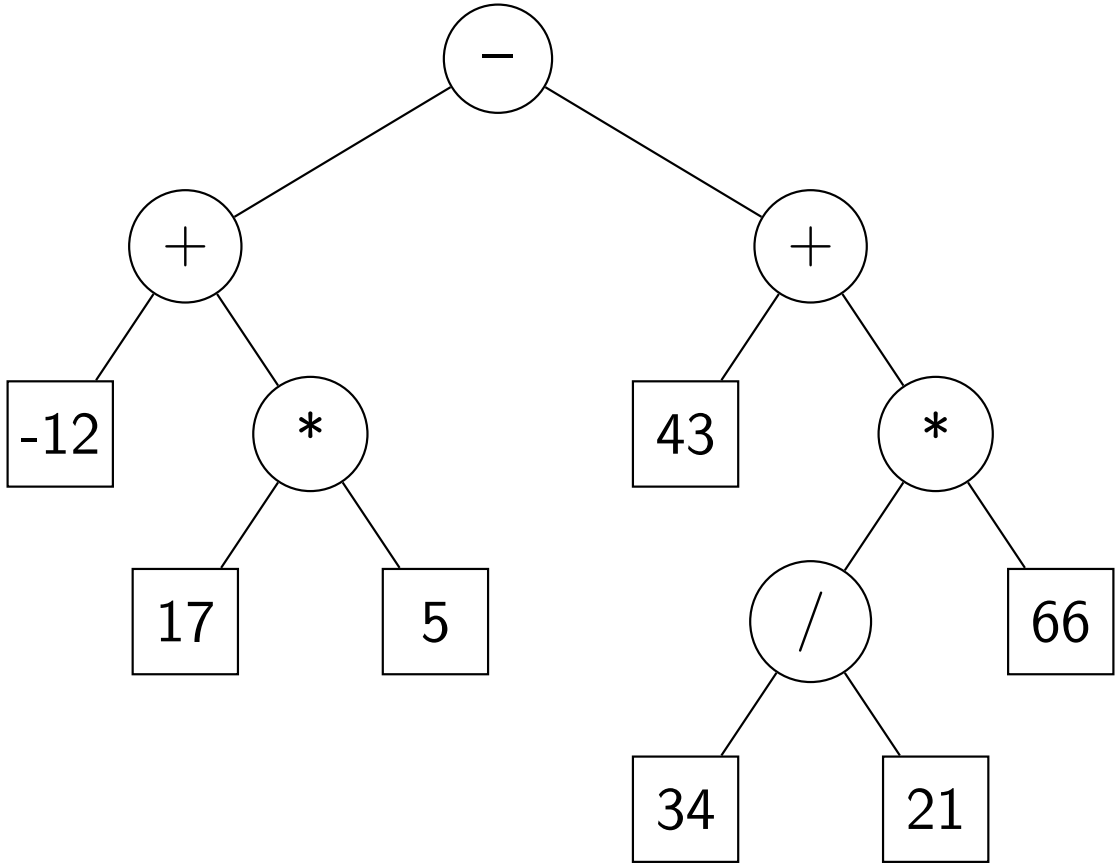
- reguli de precedență
  $12 - 5 * 2$ este $(12 - 5) * 2$ sau $12 - (5 * 2)$?
- reguli de asociere
  $15/4/2$ este $(15/4)/2$ sau $15/(4/2)$?
  $15/4 * 2$ este $(15/4) * 2$ sau $15/(4 * 2)$?

Expresiile reprezentate ca arbori

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

Expresiile reprezentate ca arbori

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



Notațiile postfixate și prefixate

- Notația postfixată se obține prin parcurgerea postordine
  $-12, 17, 5, *, +, 43, 34, 21, /, 66, *, +, -$
- Notația prefixată se obține prin parcurgerea preordine
  $-, +, -12, *, 17, 5, +, 43, *, /, 34, 21, 66$