INF 1010
Estruturas de Dados Avançadas

Árvores N-ária e Binária e Listas Generalizadas

árvores

Árvores

Uma árvore é uma estrutura de dados largamente usada que organiza os dados de forma hierárquica.

Os dados são organizados em nós e estes possuem recursos que podem ligar uns aos outros.

Árvores são organizadas em um grafo acíclico na qual cada nó possui zero ou mais filhos, e no máximo um nó pai.

Um nó é uma estrutura que pode conter um valor, uma condição ou um apontamento para uma outra estrutura de dados.

Árvore

estrutura hierárquica:

A

…B

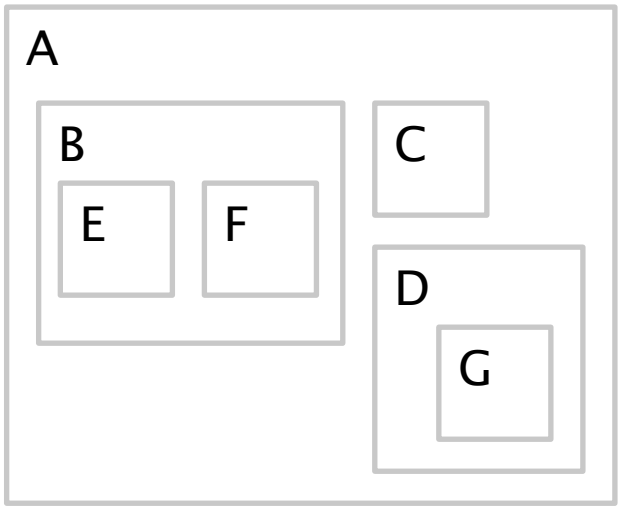
……E

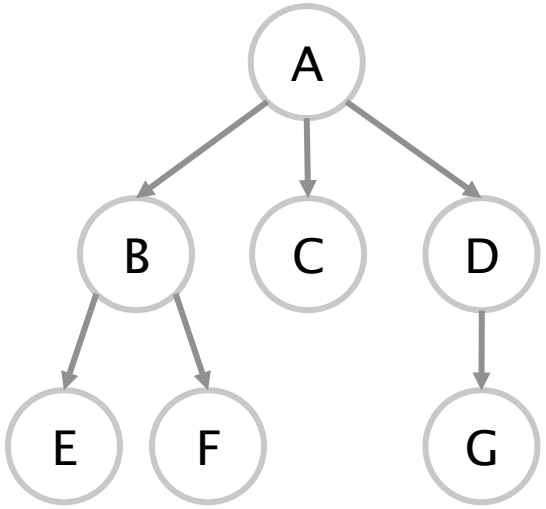
……F

…C

…D

……G





(A (B (E, F)), C, (D (G))) ???

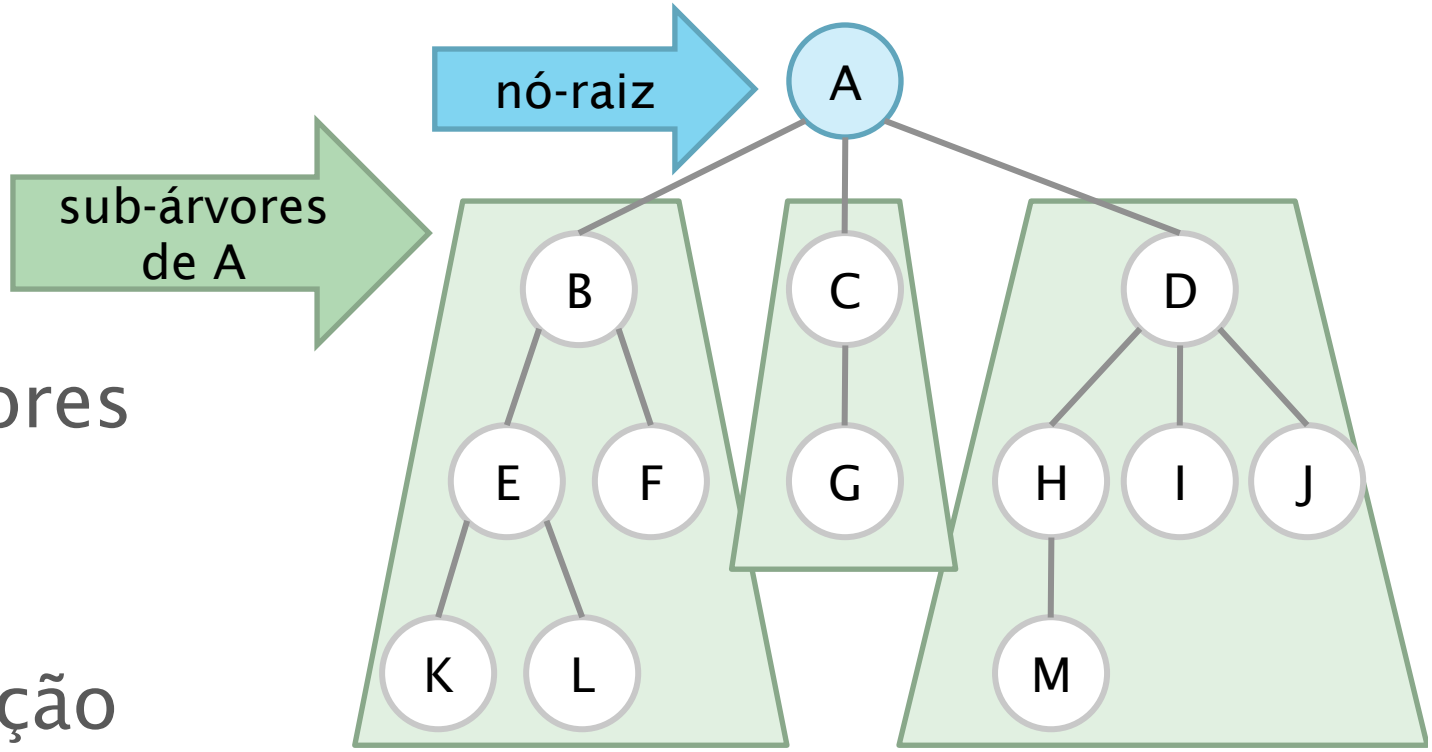
Árvores - definições

árvore:

- nó raiz
- sub-árvores

nó:

- informação
- ramos



Árvore - definições

grau de um nó:

número de sub-árvores do nó

grau de A = 3
grau de B = 2
grau de F = 0

se grau = 0

nó é chamado de **folha** ou **terminal**

{ F, G, I, J, K, L, M }

se grau > 0

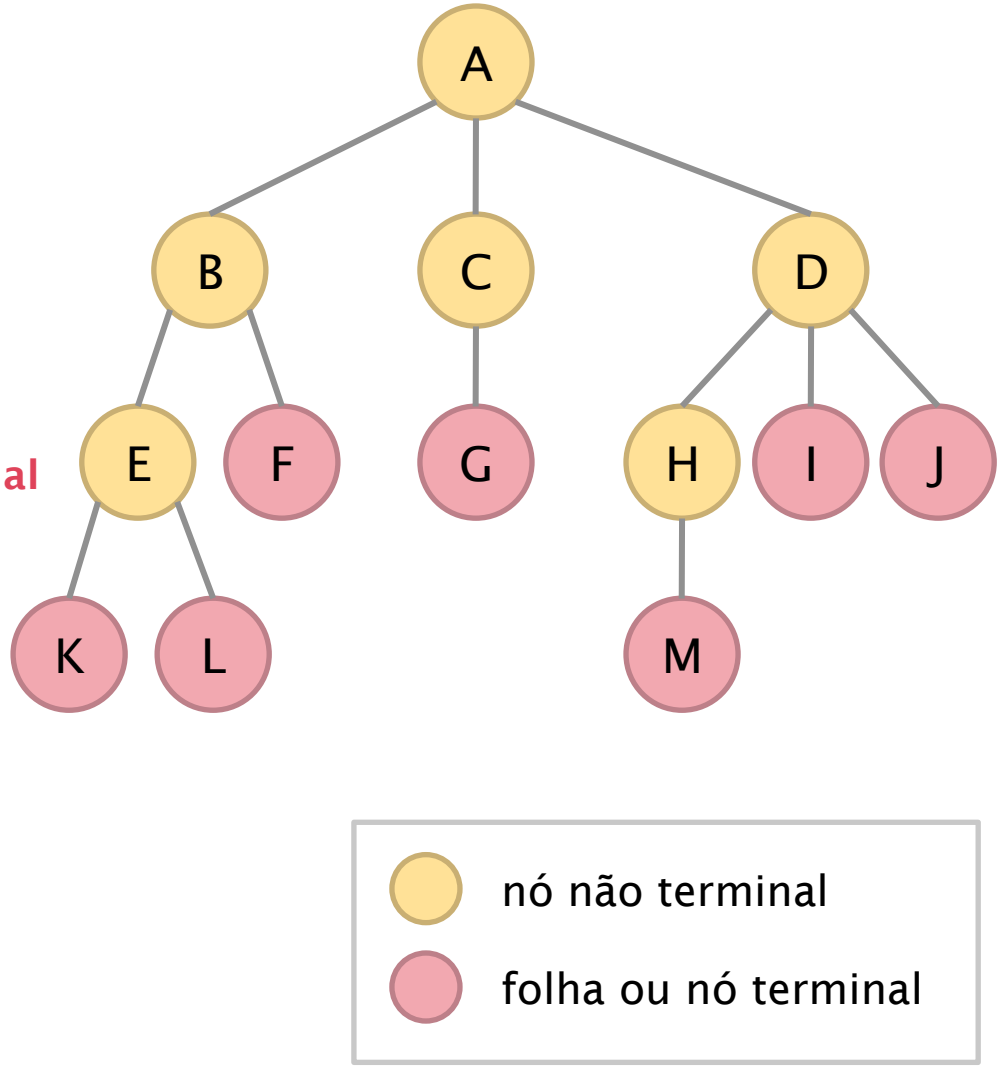
nó é chamado de **não-terminal**

{ A, B, C, D, E, H }

grau da árvore

maior dentre os graus dos nós

grau da árvore de exemplo = 3



Árvore - definições

filhos de A

raízes das sub-árvores de A

{B, C, D}

pai de B: A

X é pai dos seus filhos

irmãos

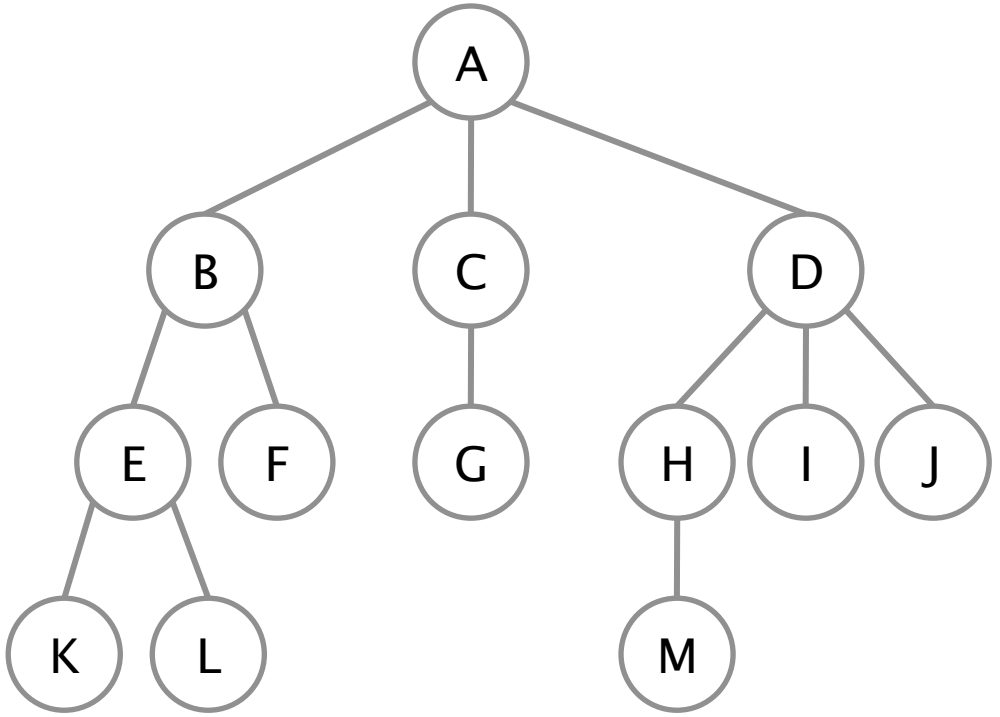
nós que têm um mesmo pai

{B, C, D}; {E, F}; {H, I, J}; {K, L}

ancestrais de K

nós no caminho da raiz até K

{A, B, E}



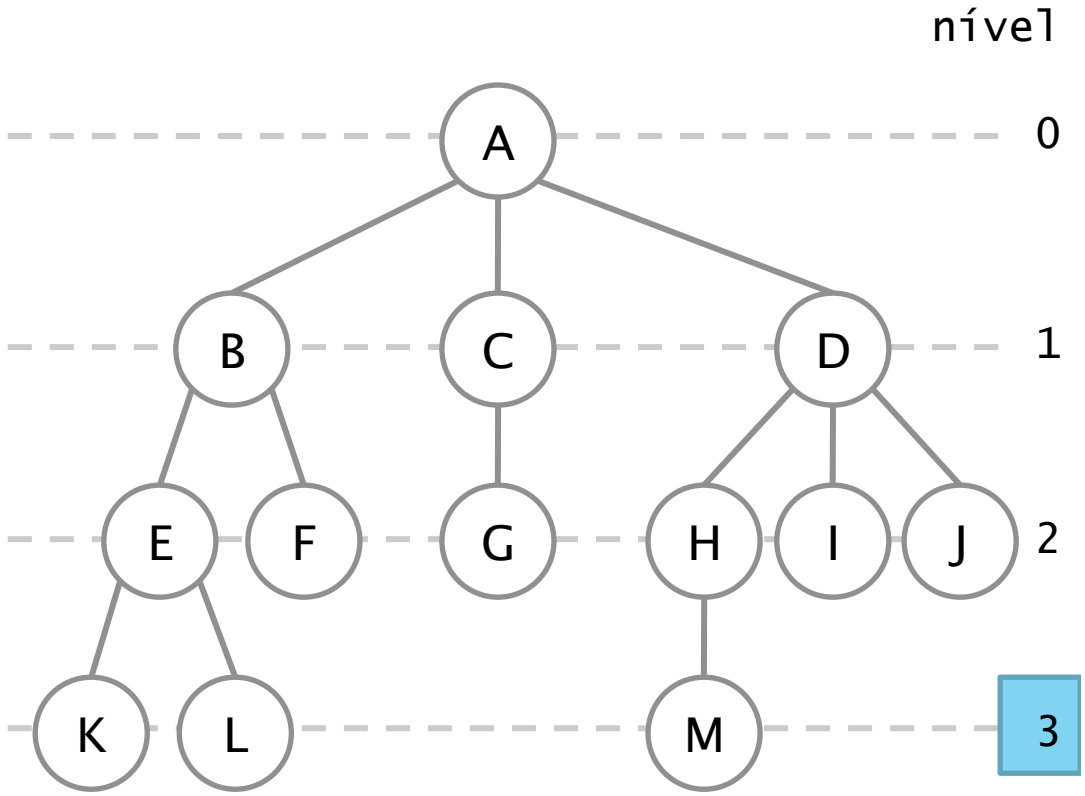
Árvore - definições

nível (de um nó)

raiz tem nível 0

se nó X tem nível n,

seus filhos têm nível n+1



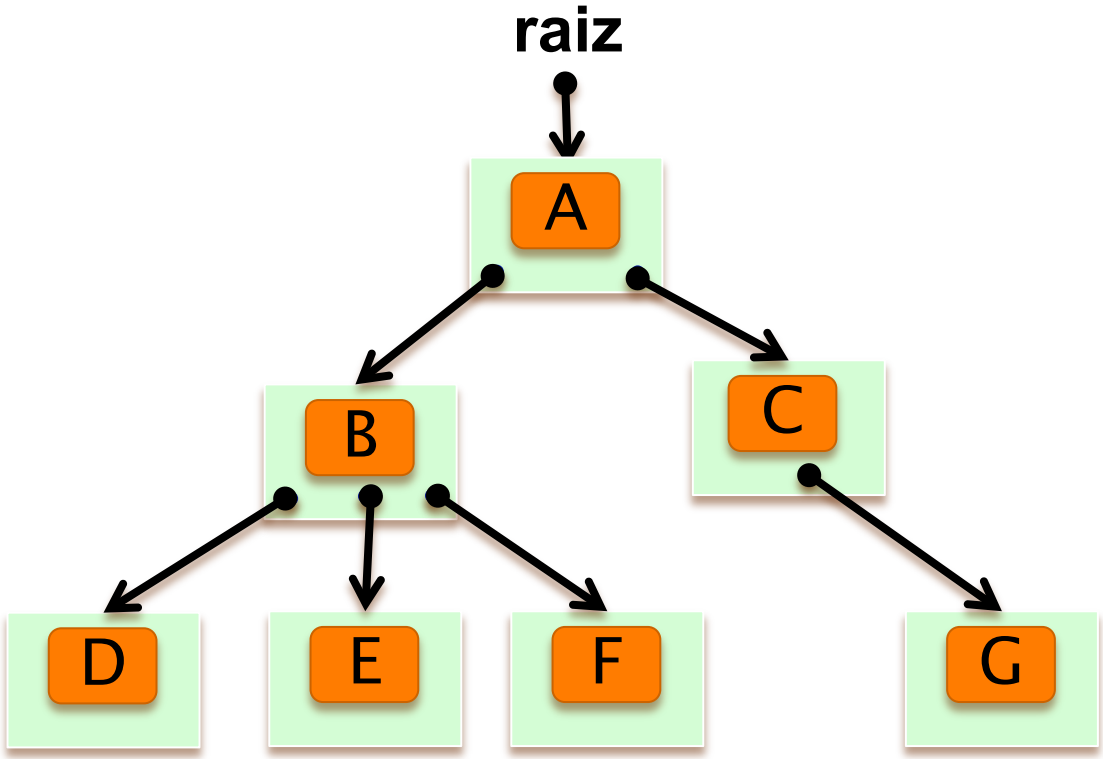
altura ou profundidade (da árvore)

maior nível dentre todos os nós

aqui: h = 3

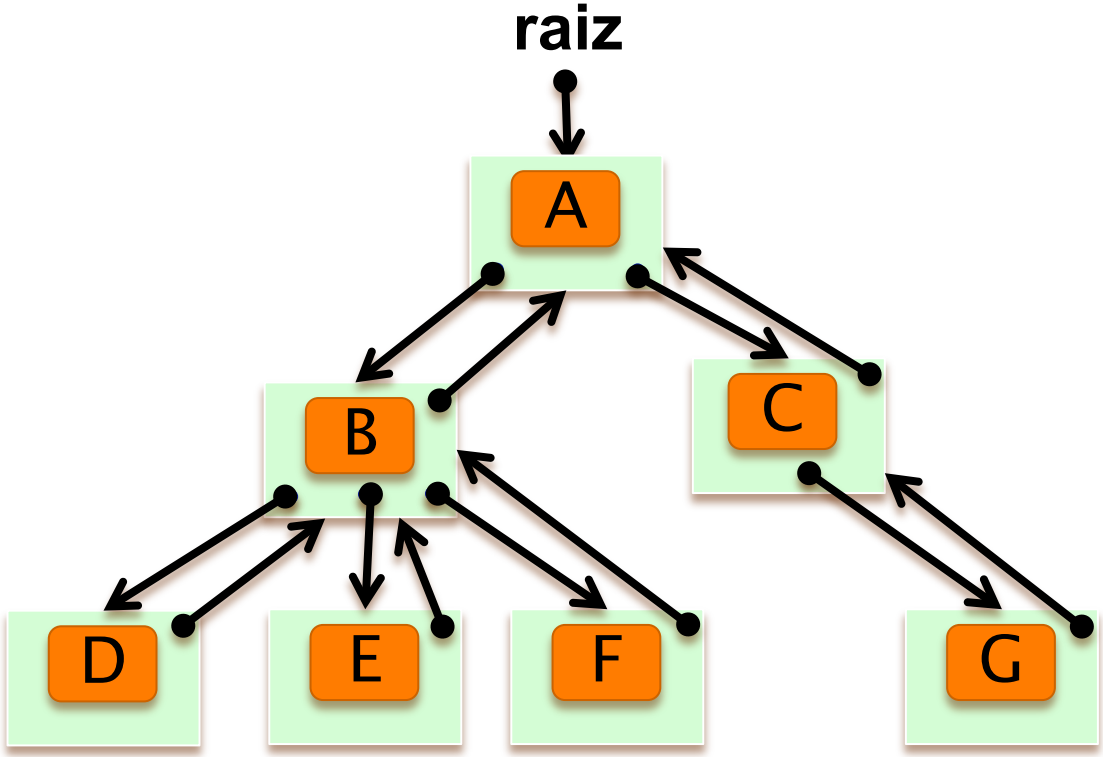
Representação Convencional

Cada nó aponta para seus filhos.



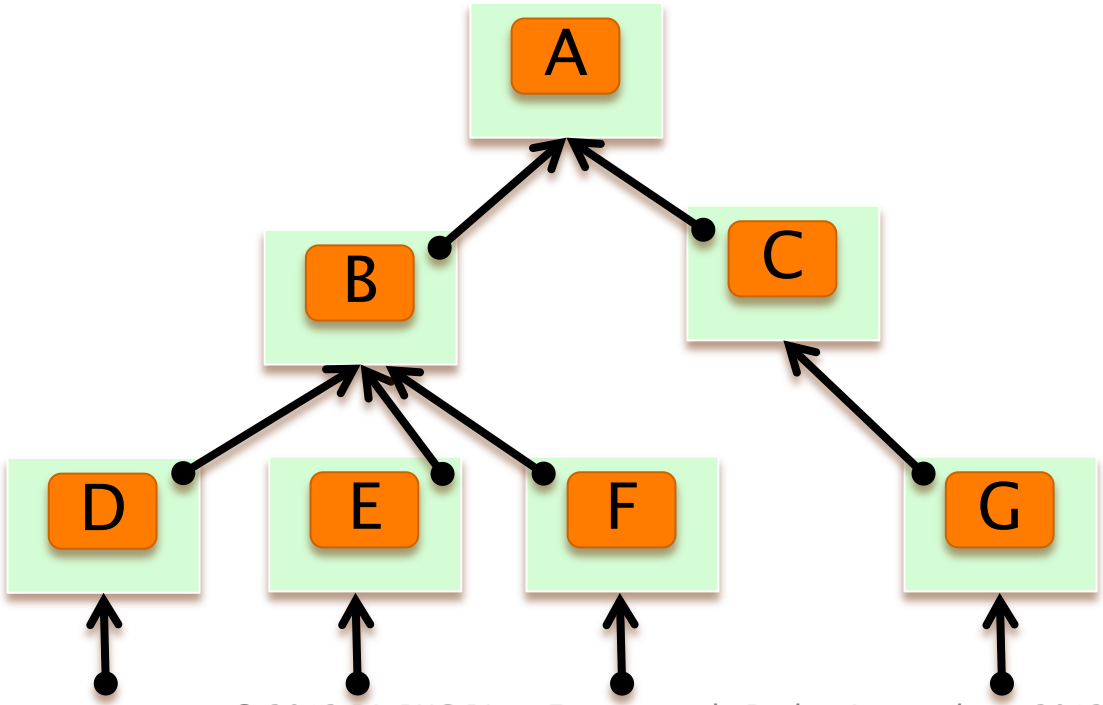
Representação por Lista Ligada Tripla

Um ponteiro adicional permite verificar quem é o pai do nó em questão. Permitindo assim se subir ou descer pela árvore

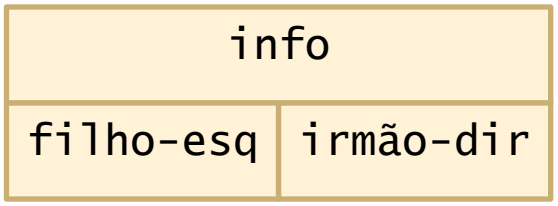


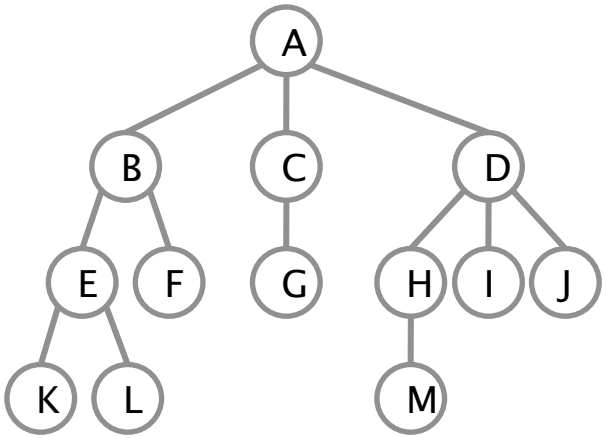
Representação Invertida

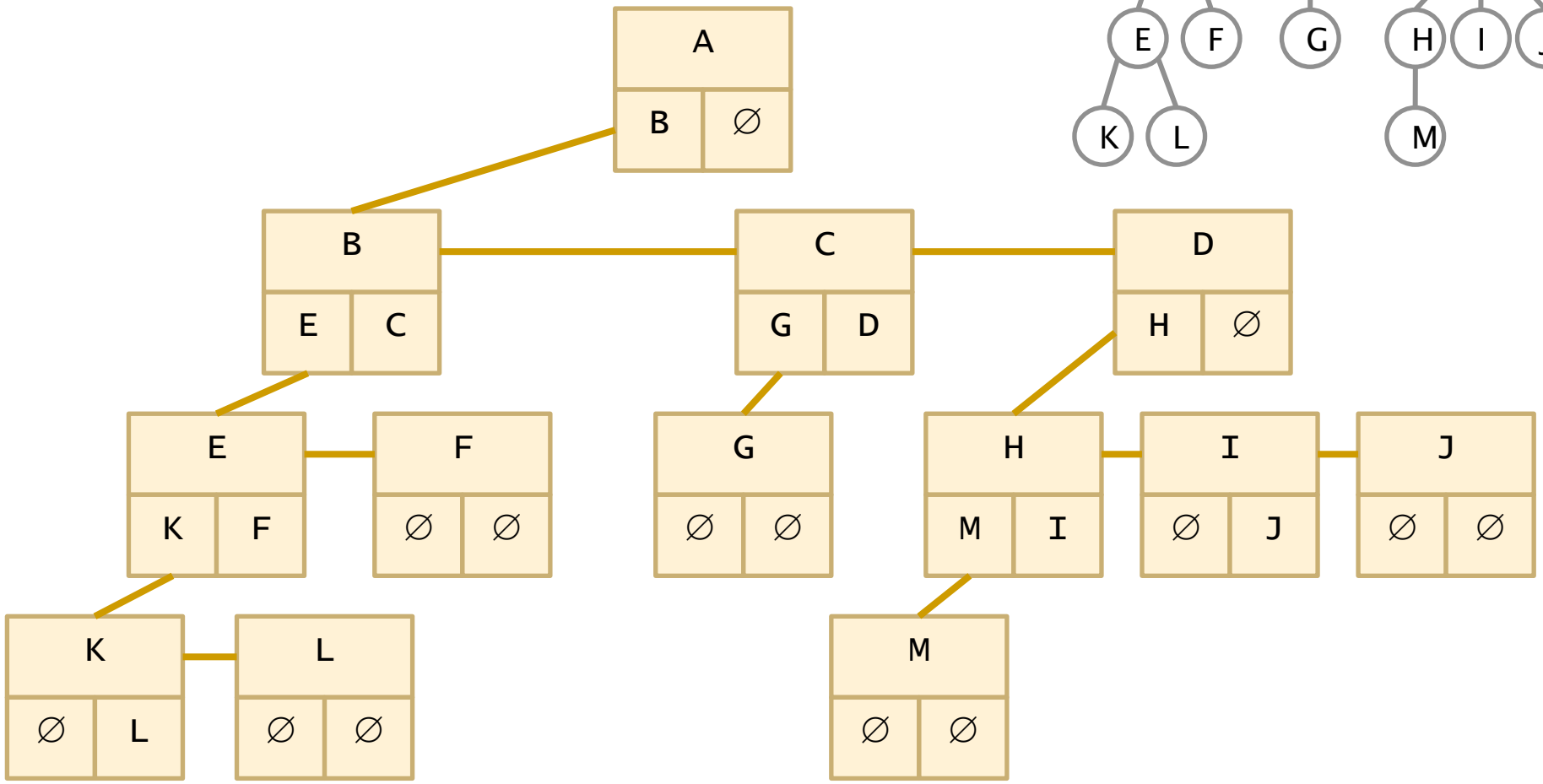
Os filhos apontam para o pai. Esta representação precisa de um conjunto de apontadores para todos os nós folhas.



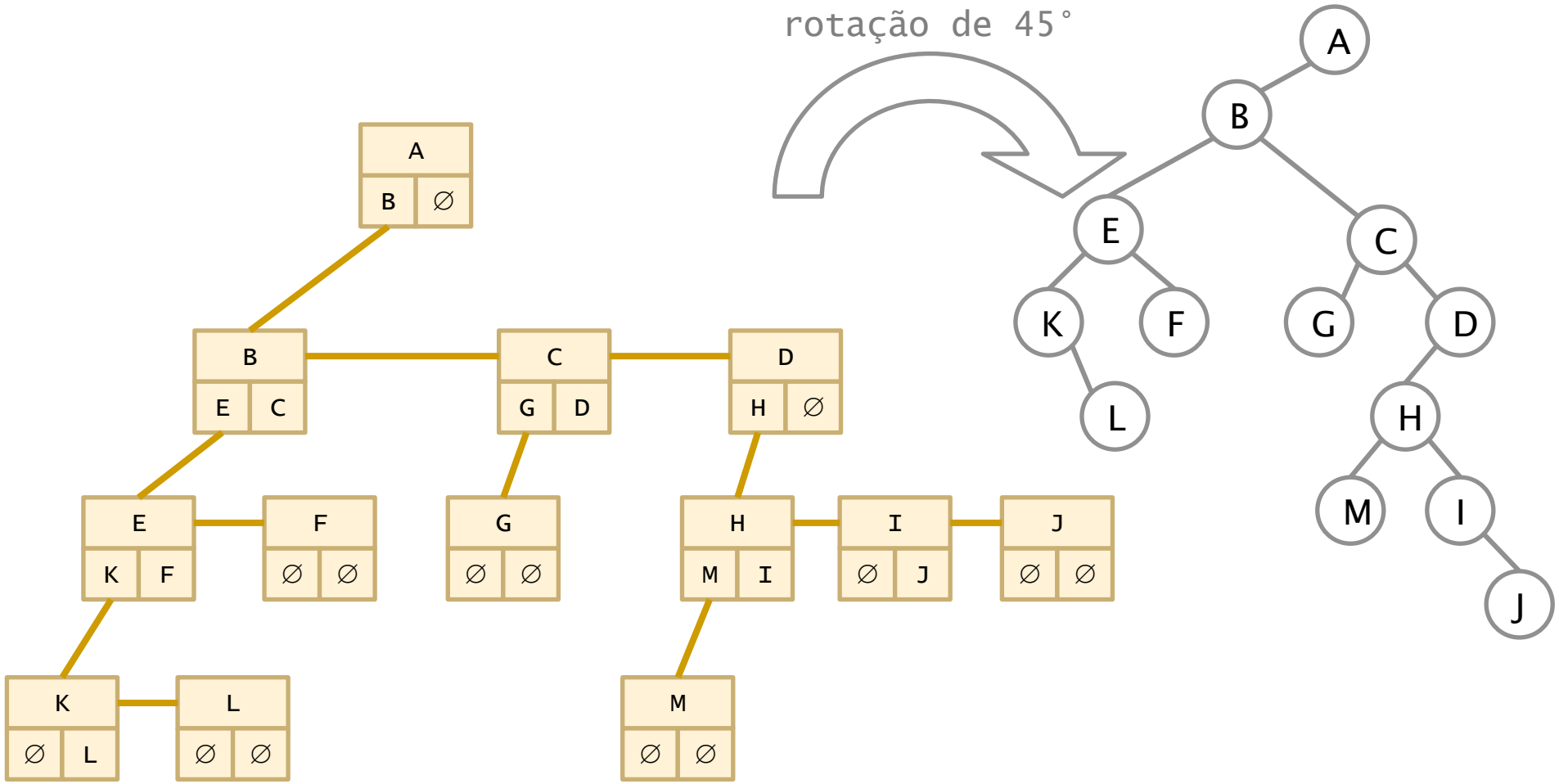
Representação binária de uma árvore







Representação binária como árvore de grau 2 (árvore binária)



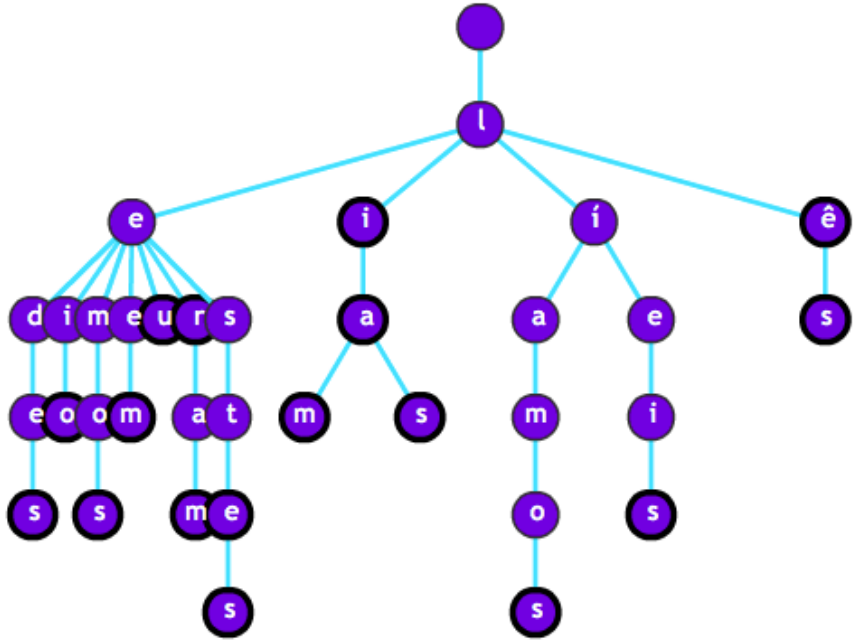
Árvores Digitais (Tries)

Uma árvores digital (trie, prefix tree) é uma estrutura de árvore ordenada usada para armazenar vetores associativos.

O principio é usar os elementos do 'conteúdo' para indicar o 'caminho de busca'

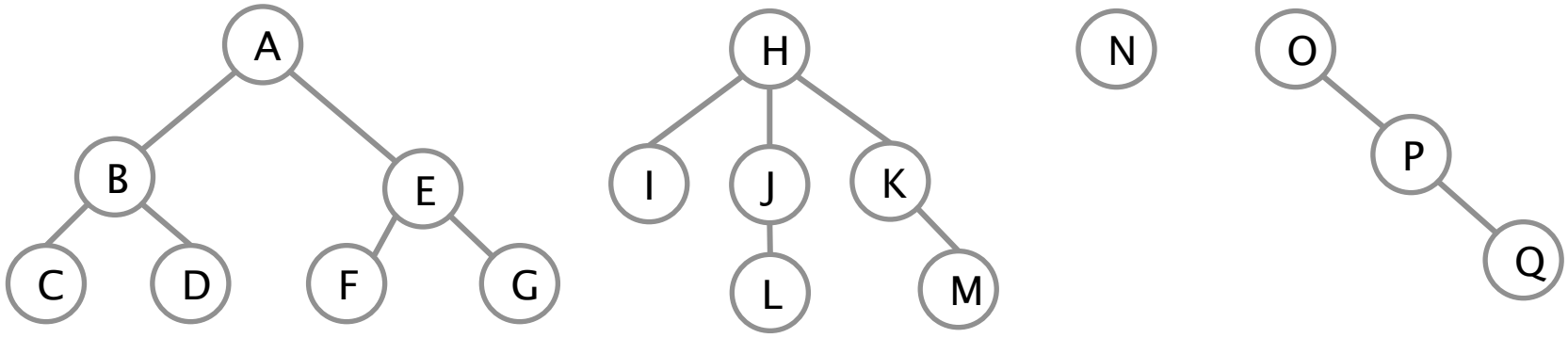
Árvores Digitais (Tries)

Sequencias de caracteres podem ser representadas por árvores digitais. As arestas recebem os caracteres. Nós finais delimitam palavras.



Floresta

Uma sequencia de árvores é chamada de floresta.
As sub árvores de um nó podem ser consideradas uma floresta.



dúvidas?

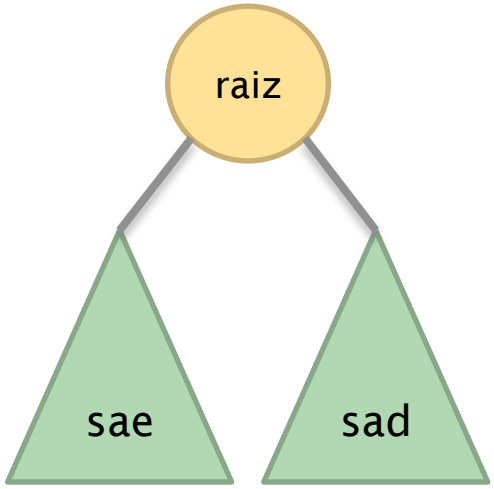
árvores binárias

Árvore binária - definição

árvore binária: conjunto finito de nós

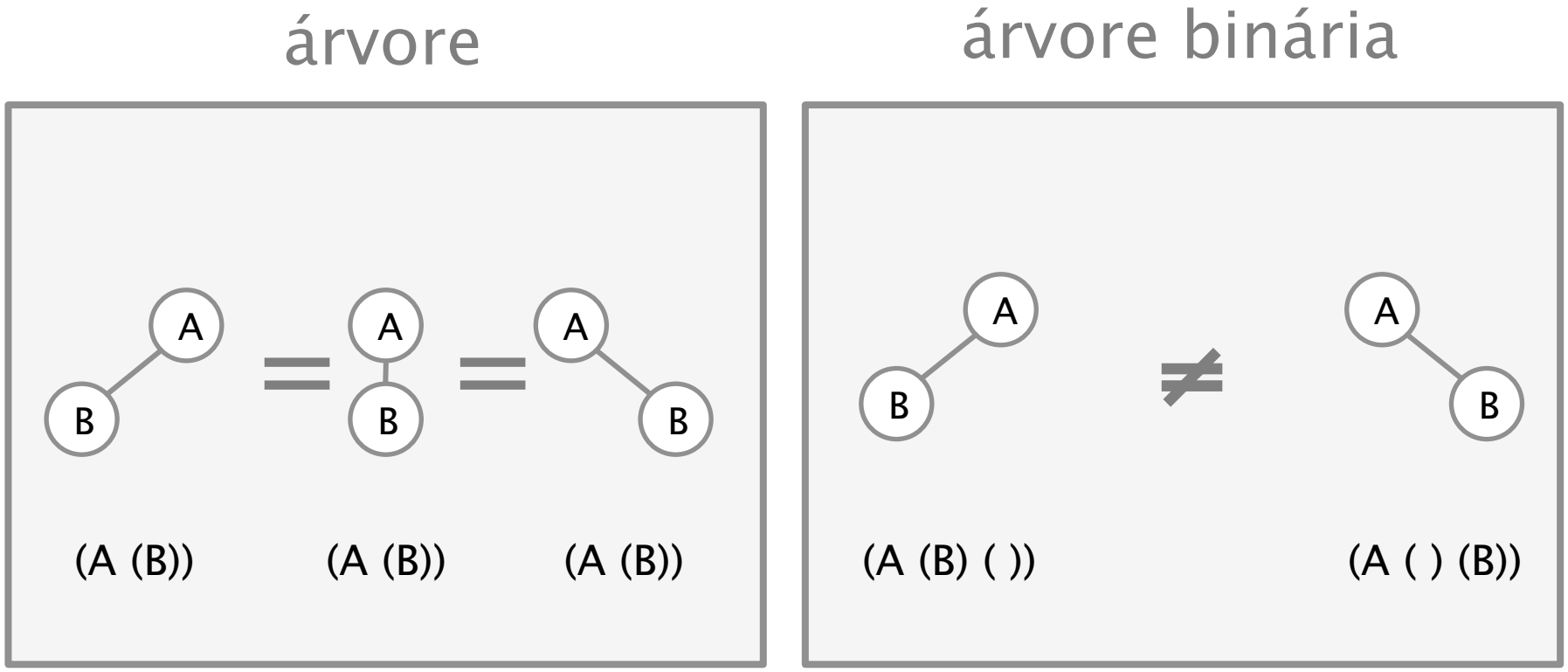
- ∅ (árvore vazia)
- {raiz, sub-árvore esquerda, sub-árvore direita}, onde sae e sad são conjuntos disjuntos

∅ ou




```
/* nó da árvore binária */
struct arvbin {
    char info;
    struct arvbin *esq;
    struct arvbin *dir;
};
```

Árvore binária não é árvore comum



Árvore binária - conceitos

número máximo de nós no nível *i*:

2^i

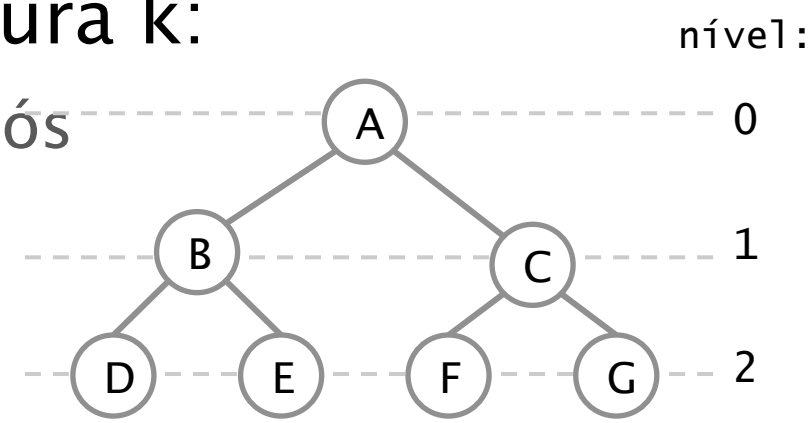
número máximo de nós na árvore de altura *k*:

$2^{k+1} - 1, k \geq 0 \quad (= 2^k \ldots + 2^2 + 2 + 1)$

árvore binária **cheia** de altura k:

árvore que possui $2^{k+1} - 1$ nós

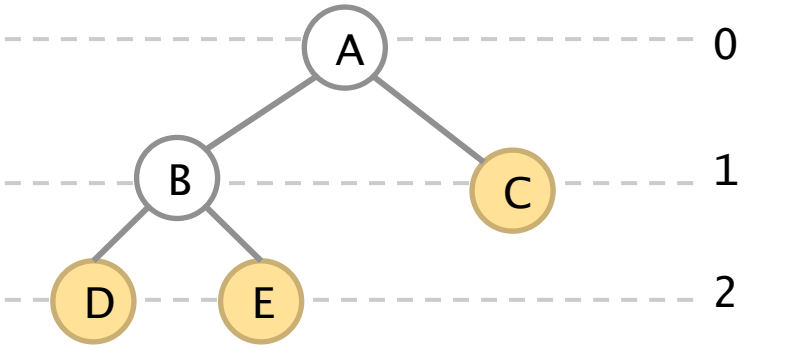
se k = 2, árvore binária cheia possui $2^3-1 = 7$ nós



Árvore binária – conceitos (cont.)

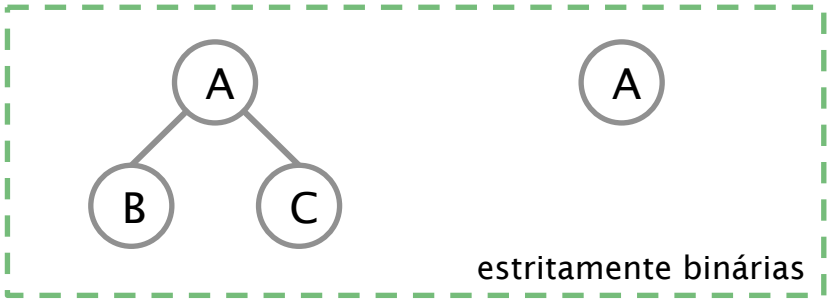
árvore binária **completa**

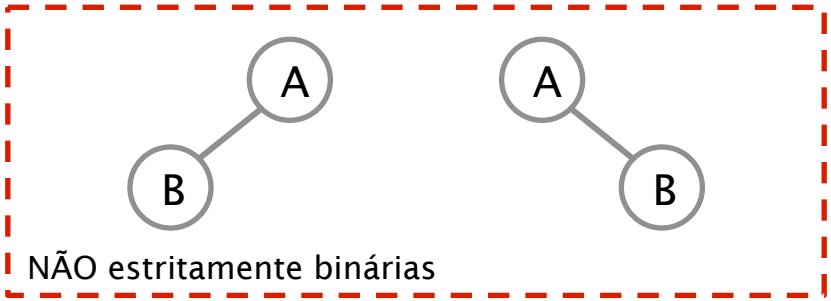
toda folha está no último ou penúltimo nível



árvore **estritamente binária**

cada nó tem 0 ou dois filhos





TAD Árvore Binária

```
typedef struct arvbin ArvBin;

ArvBin* arvbin_cria_vazia (void);
ArvBin* arvbin_cria (char c, ArvBin *sae, ArvBin *sad);
ArvBin* arvbin_libera (ArvBin *a);
int arvbin_vazia (ArvBin *a);
ArvBin* arvbin_esq (ArvBin *a);
ArvBin* arvbin_dir (ArvBin *a);
ArvBin* arvbin_encontra (ArvBin *a, char c);
char arvbin_elemento (ArvBin *a);
void arvbin_exibe_pre (ArvBin *a);
void arvbin_exibe_pos (ArvBin *a);
void arvbin_exibe_in (ArvBin *a);
void arvbin_exibe_nivel(ArvBin *a);
```

Percursos em Árvores

Pré-ordem:

- Visitar a raiz da sub-árvore
- Percorrer a sub-árvore esquerda em pré-ordem
- Percorrer a sub-árvore direita em pré-ordem

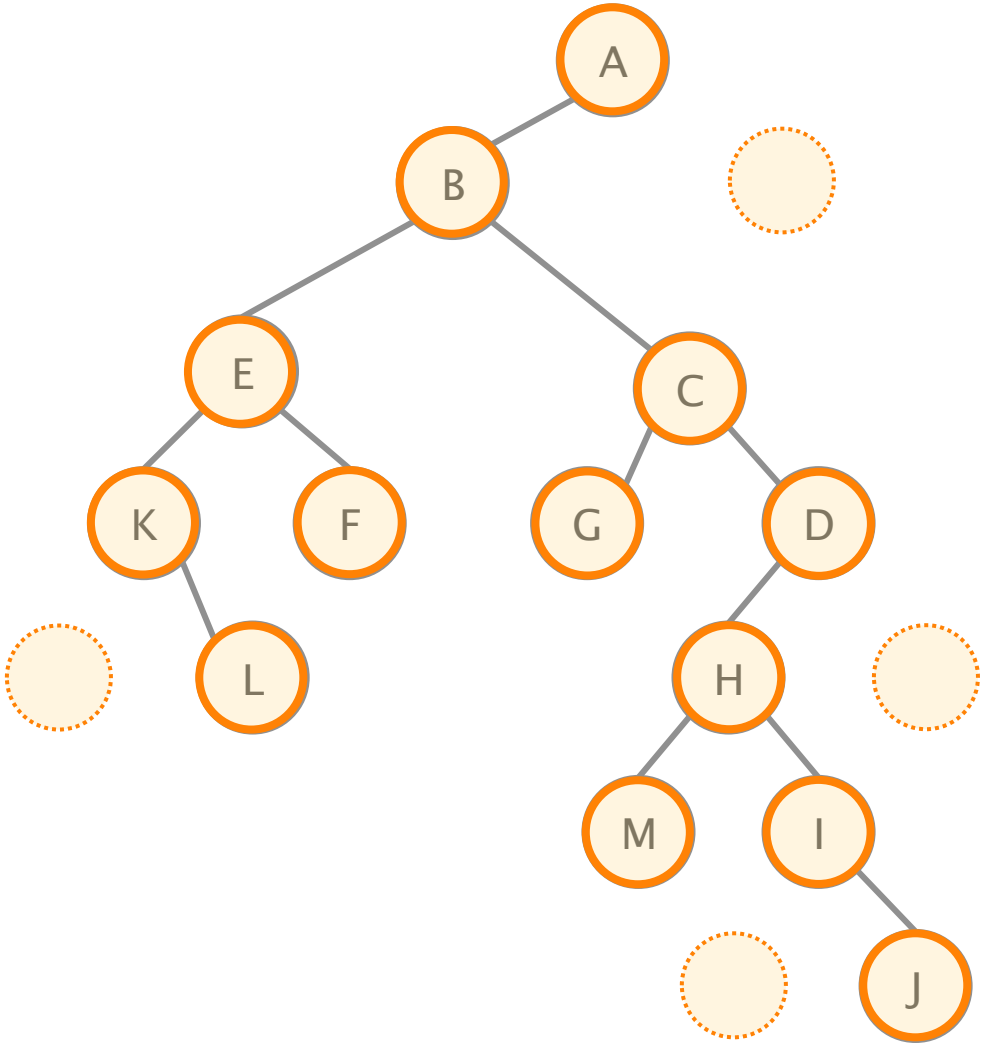
Pós-ordem

- Percorrer a sub-árvore esquerda em pós-ordem
- Percorrer a sub-árvore direita em pós-ordem
- Visitar a raiz da sub-árvore

Ordem Simétrica (Inordem):

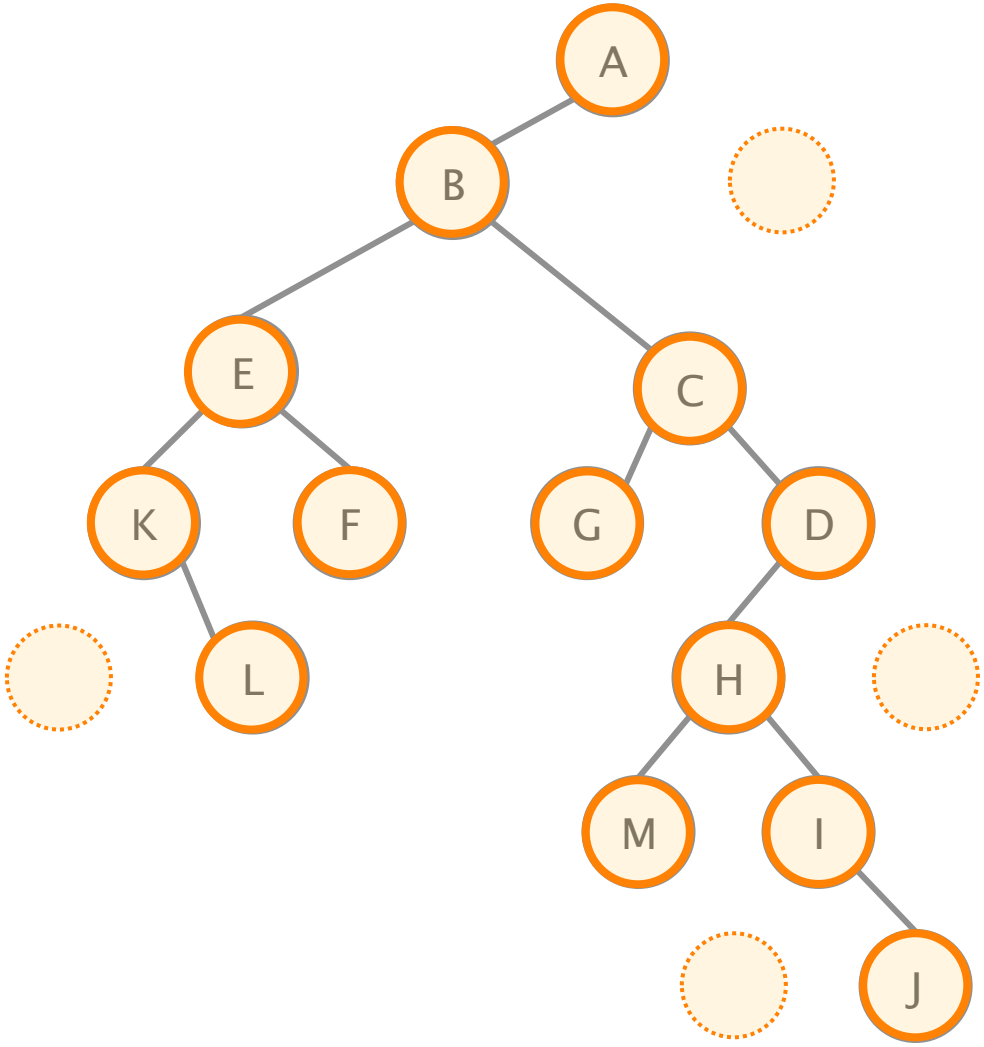
- Percorrer a sub-árvore esquerda em inordem
- Visitar a raiz da sub-árvore
- Percorrer a sub-árvore direita em inordem

Percurso – pré-ordem



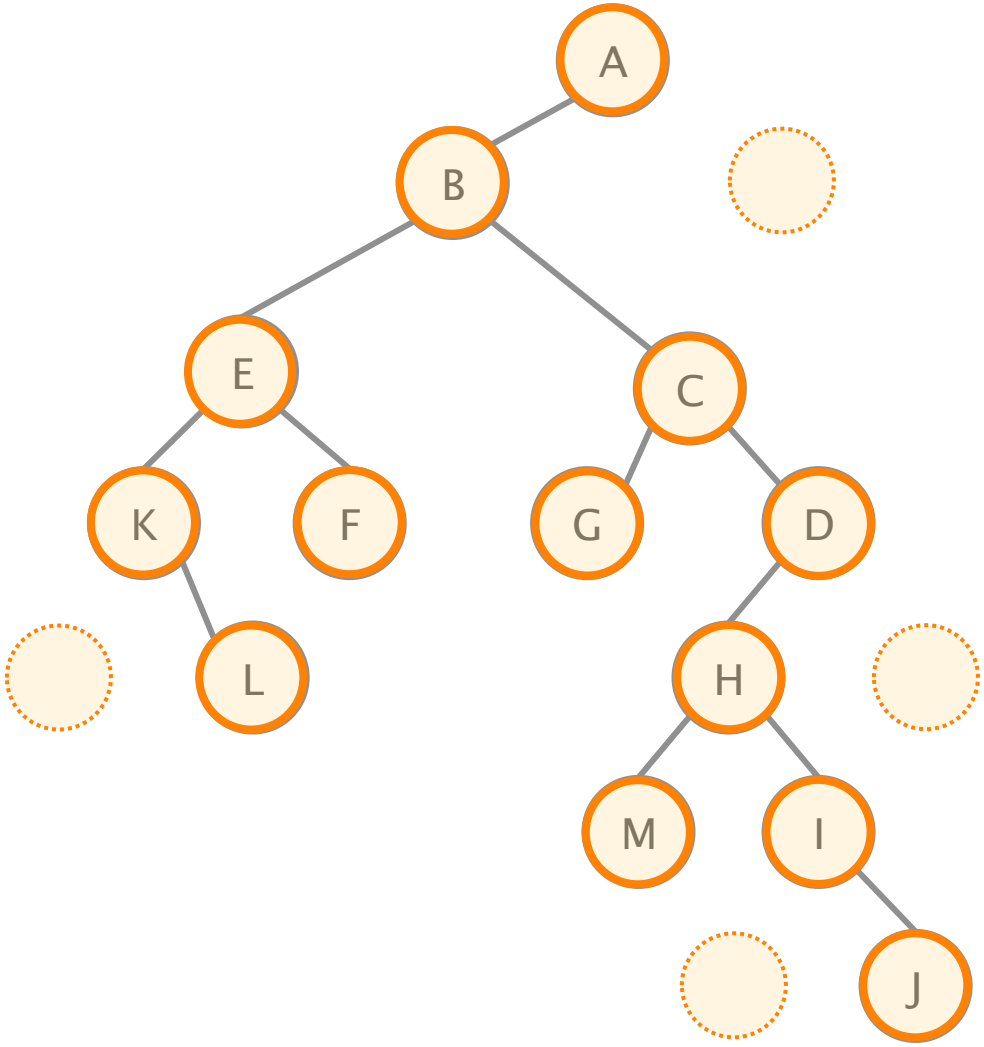
A	B	E	K	*	L	F	C	G	D	H	M	I	*	J	*	*

Percurso – pós-ordem



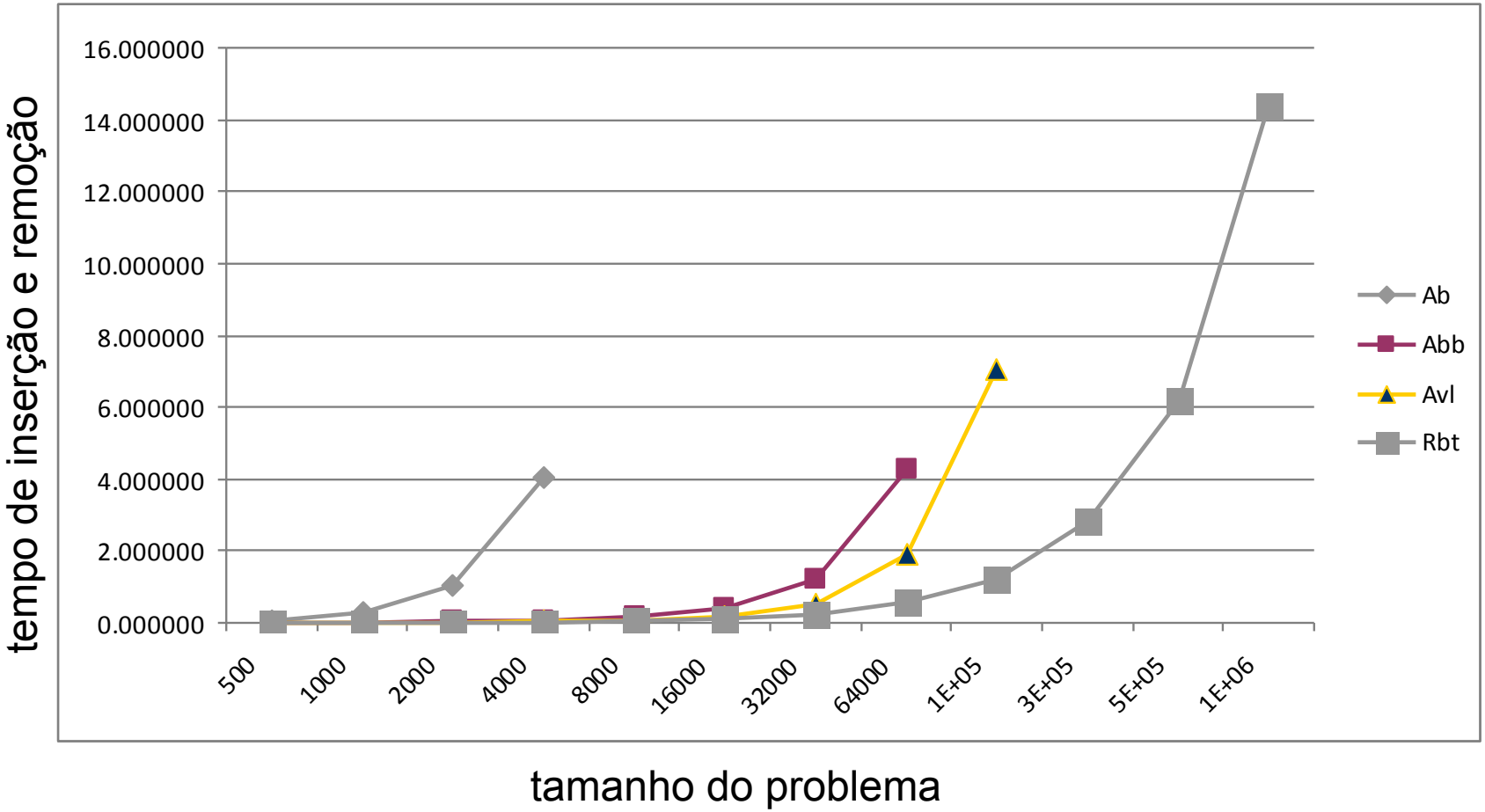
*	L	K	F	E	G	M	*	J	I	H	*	D	C	B	*	A

Percurso – ordem simétrica



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

O que vamos estudar de árvores binárias?



dúvidas?

listas generalizadas

Listas Generalizadas

Cada elemento de uma lista generalizada ou é um á*tomo*, ou é uma lista generalizada.

Um átomo pode ser um valor, uma letra, um texto, etc...

Listas Generalizadas
Notação Parentética

Representação:

$(\alpha_1, \alpha_2, \ldots, \alpha_n)$

α_i é um átomo ou uma lista generalizada

Exemplo:

A = ( )
B = (1,2,3)
C = ((3),(4,8),(5,(6)))
D = (B,4,5)
E = (1,E,2)

Expansão

Lista D = (B,4,5) pode ser expandida:

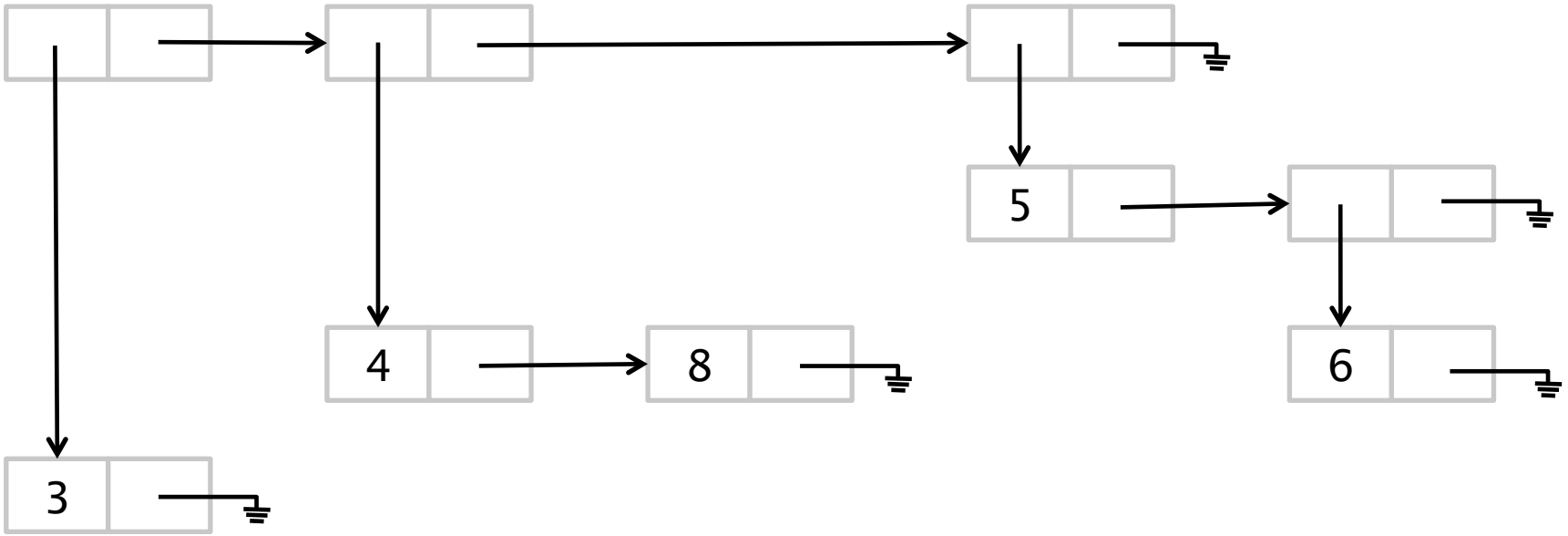
D = ((1,2,3),4,5)

Lista E = (1,E,2) é uma recursão infinita:

E = (1,(1,(1,(1,(1,...,2),2),2),2),2),2)

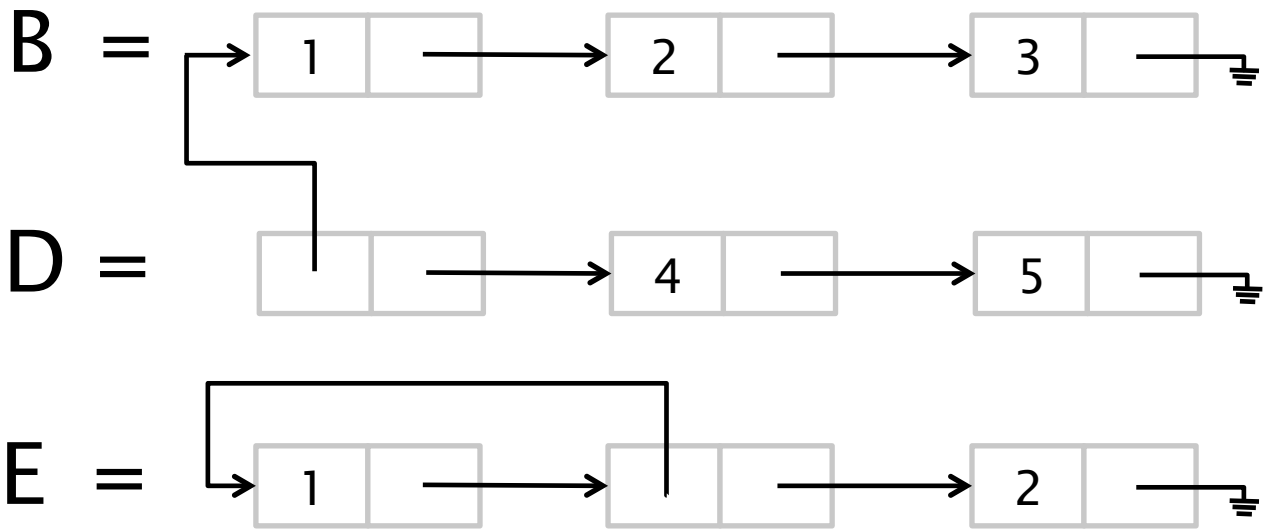
Representação Gráfica

C = ((3),(4,8),(5,(6)))



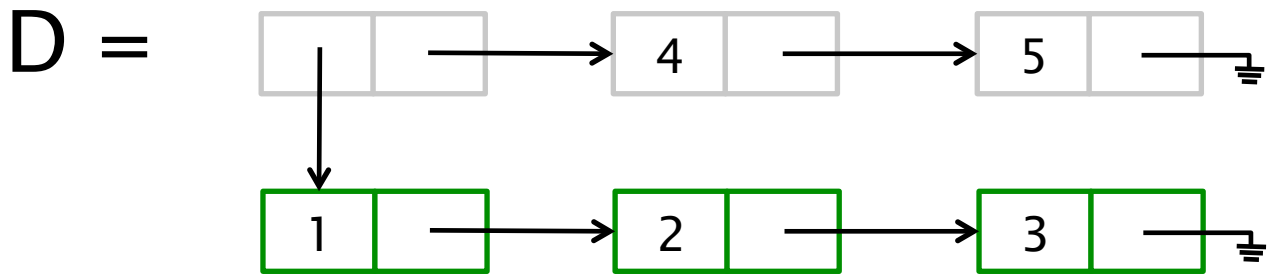
Implementação Compartilhada

As listas podem compartilhar diretamente as outras listas:



Implementação com Cópia

Neste modo os dados são copiados para a lista. Porém, não é possível completar a expansão de listas recursivas.



dúvidas?