

Algoritmos e Estruturas de Dados I

Aula 09: Árvores

Prof. Márcio Porto Basgalupp

créditos: Prof. Jurandy G. Almeida Jr.

Universidade Federal de São Paulo
Departamento de Ciência e Tecnologia

- Listas podem ser convenientemente definidas da seguinte forma: Uma lista do tipo **T** é
 - Uma lista (estrutura) vazia ou
 - Uma concatenação (cadeia) de um elemento do tipo **T** com uma lista cujo tipo básico também seja **T**
- Nota-se que a recursão é utilizada como ferramenta de definição

- Uma árvore é uma estrutura sofisticada cuja definição por meio de recursão é elegante e eficaz
- Uma árvore, com tipo **T**, pode ser definida recursivamente da seguinte forma:
 - Uma árvore (estrutura) vazia ou
 - Um nó do tipo **T** associado a um número finito de estruturas disjuntas de árvore do mesmo tipo **T**, denominadas **subárvores**

- Observando a similaridade das definições é evidente que uma lista possa ser considerada como uma árvore na qual cada nó tem, no máximo, uma única subárvore
- Por esse motivo, uma **lista** é também denominada **árvore degenerada**

Definição

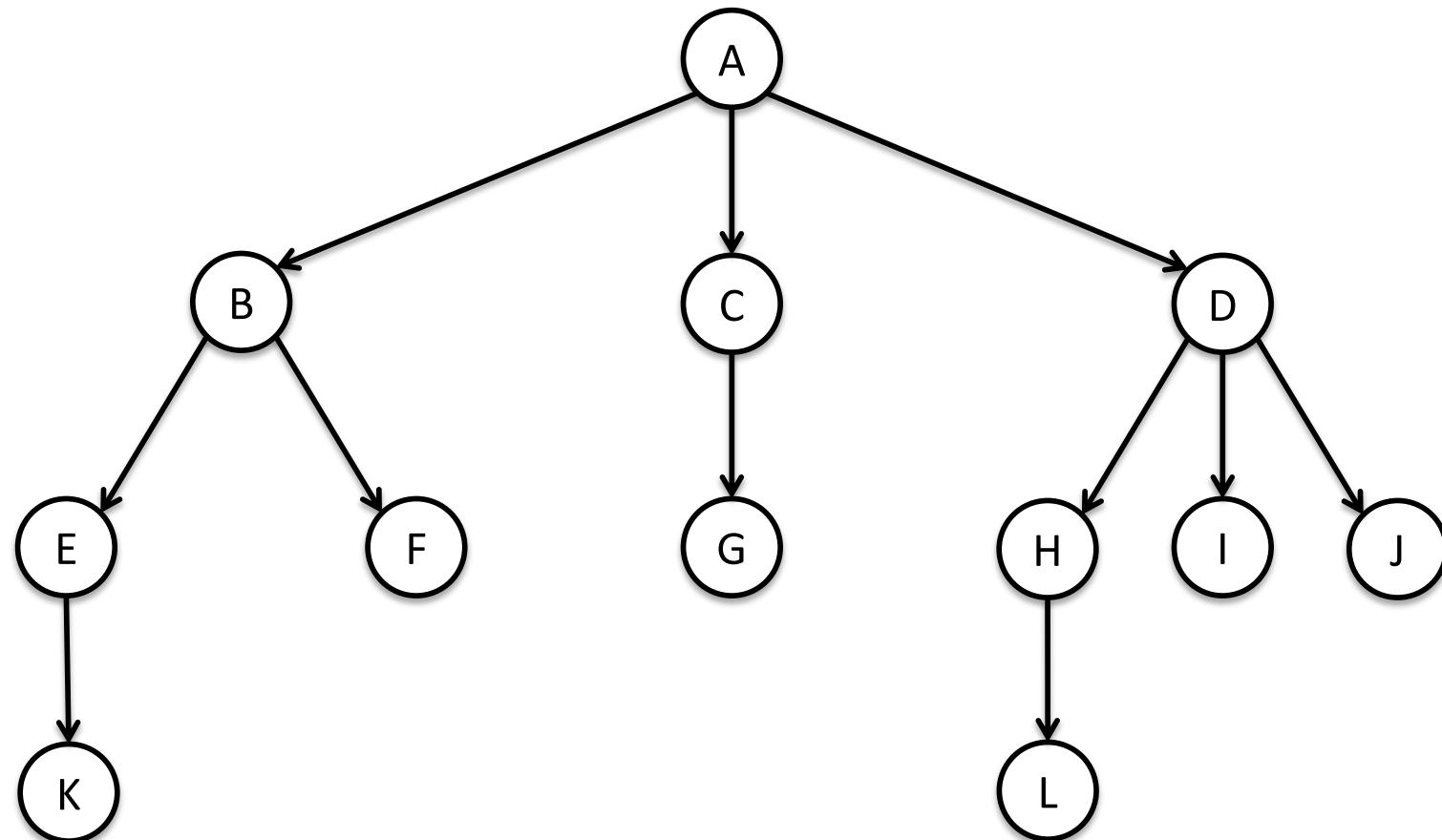
- Uma **árvore** é um conjunto finito de um ou mais nós tais que:
 - Existe um nó especial, denominado **raiz (árvore enraizada)**
 - Os demais nós encontram-se desdobrados em $n \geq 0$ conjuntos disjuntos T_1, \dots, T_n sendo que cada conjunto se constitui numa árvore
 - T_1, \dots, T_n são denominadas subárvores da raiz
- Utilizaremos grafos para representar árvores
- Todavia, existem outras representações equivalentes para árvores: conjuntos aninhados (diagrama de inclusão), parênteses aninhados, paragrafação

REPRESENTAÇÃO

Representação

Grafo

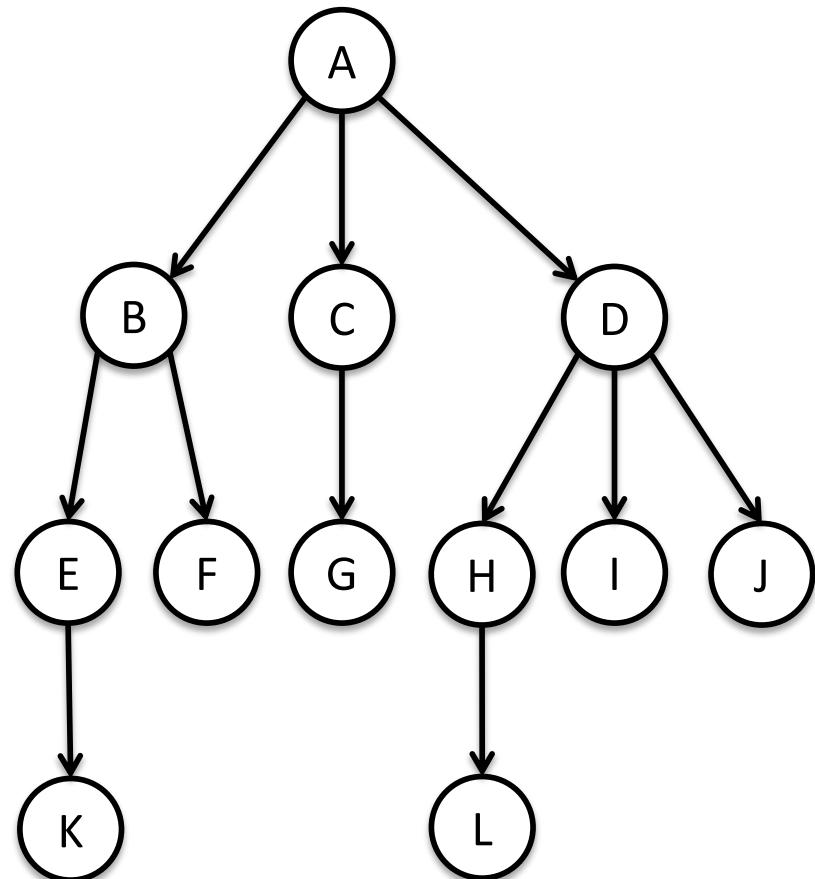
- Uma árvore é um grafo conexo e acíclico



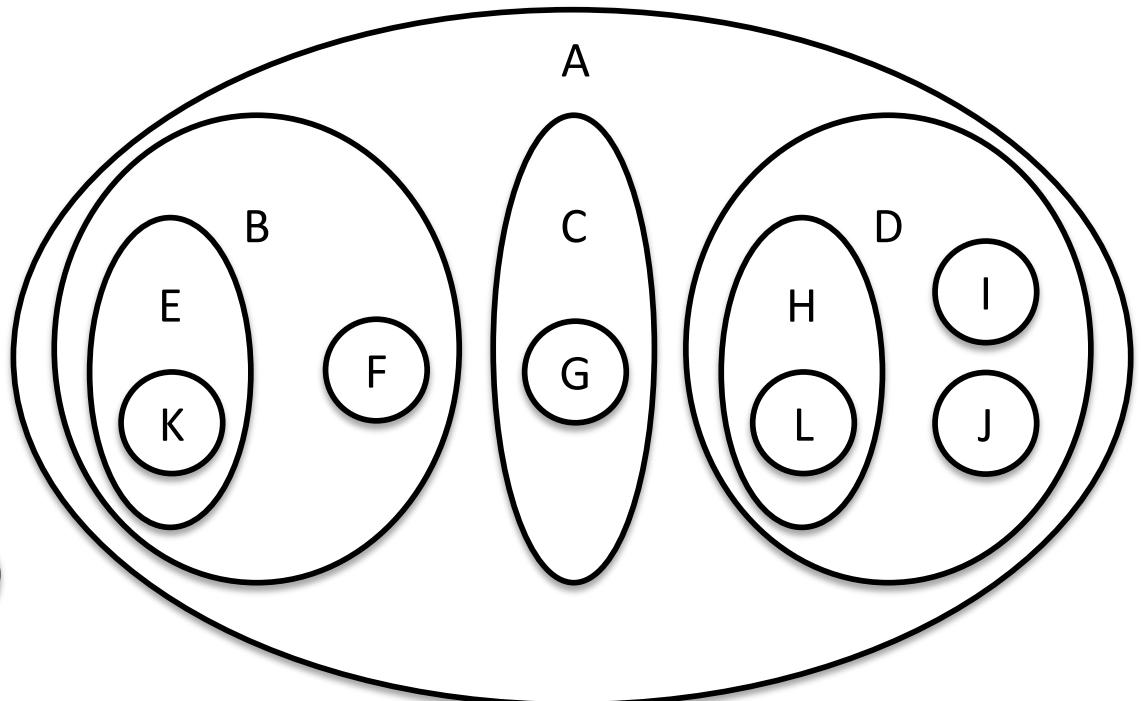
Representação

Conjuntos Aninhados

- Grafo



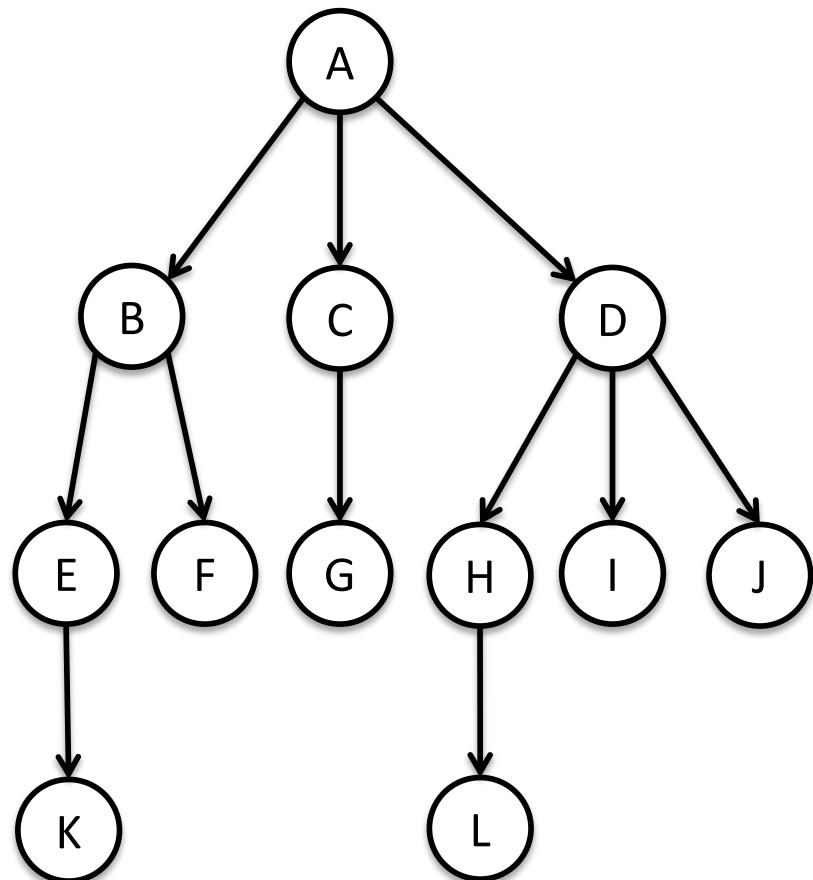
- Conjuntos aninhados



Representação

Parênteses aninhados

- Grafo



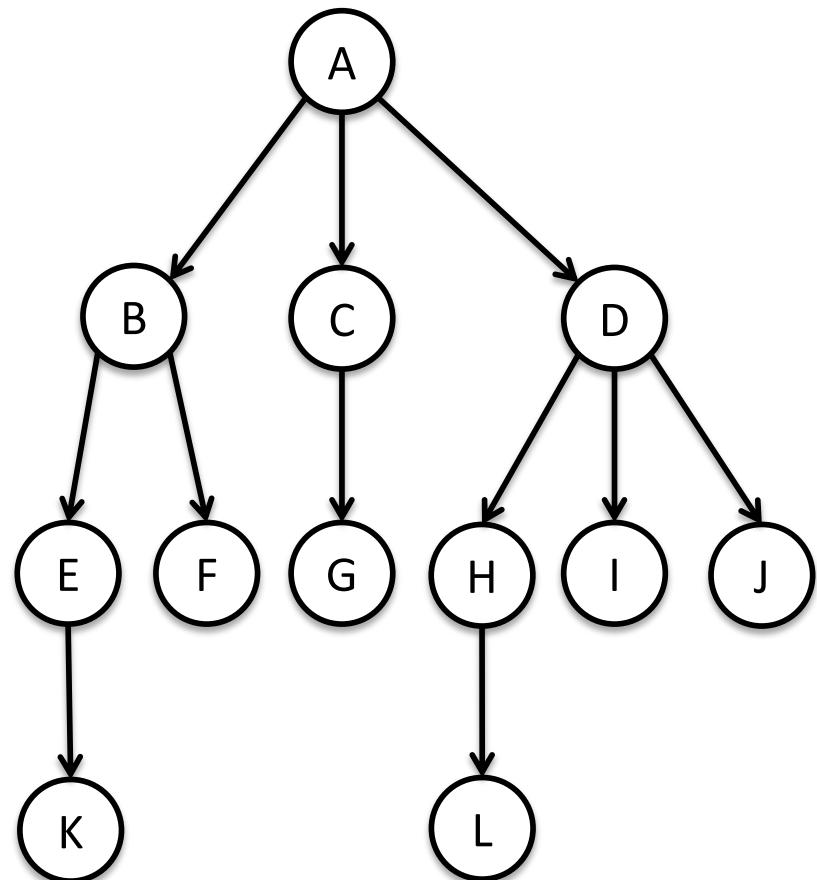
- Parênteses aninhados

- (A (B (E (K) F) C (G) D (H (L) I J)))

Representação

Paragrafação

■ Grafo



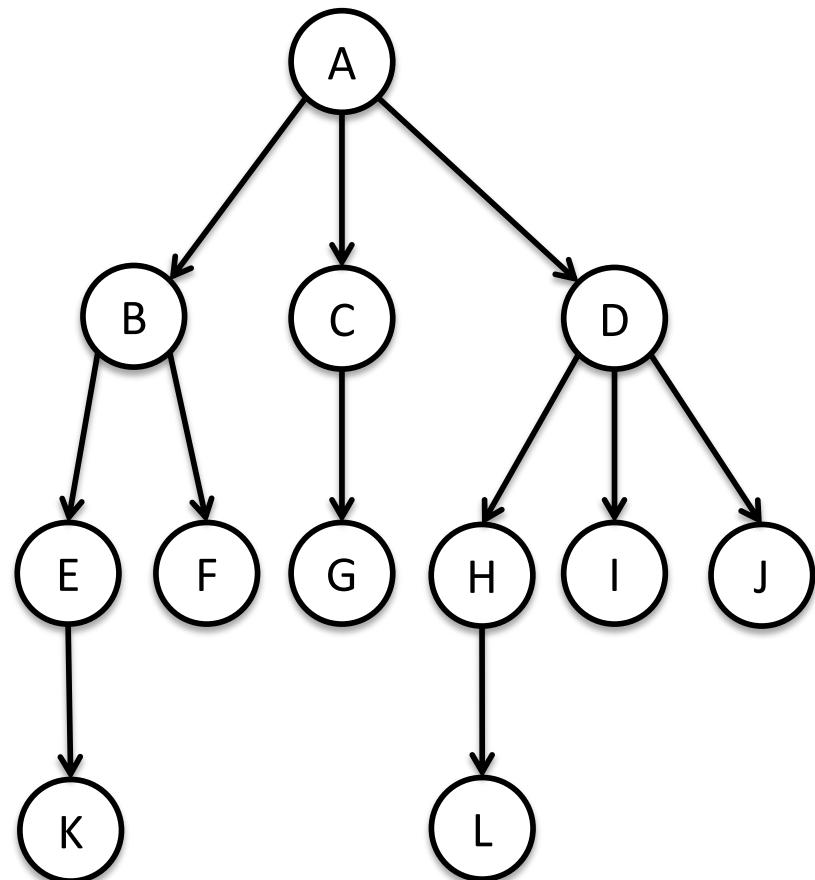
■ Paragrafação

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

Representação

Paragrafação

■ Grafo

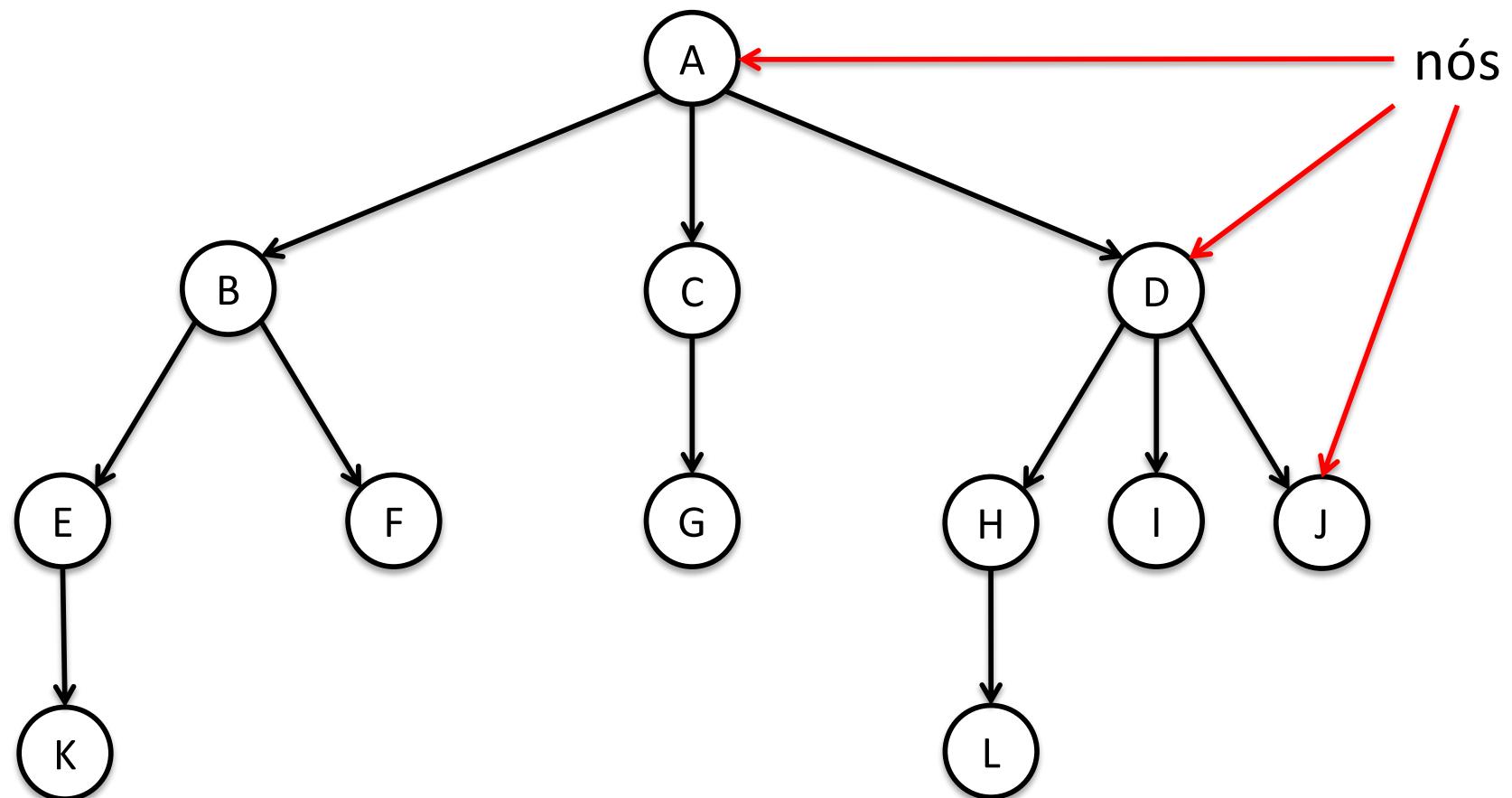


■ Paragrafação

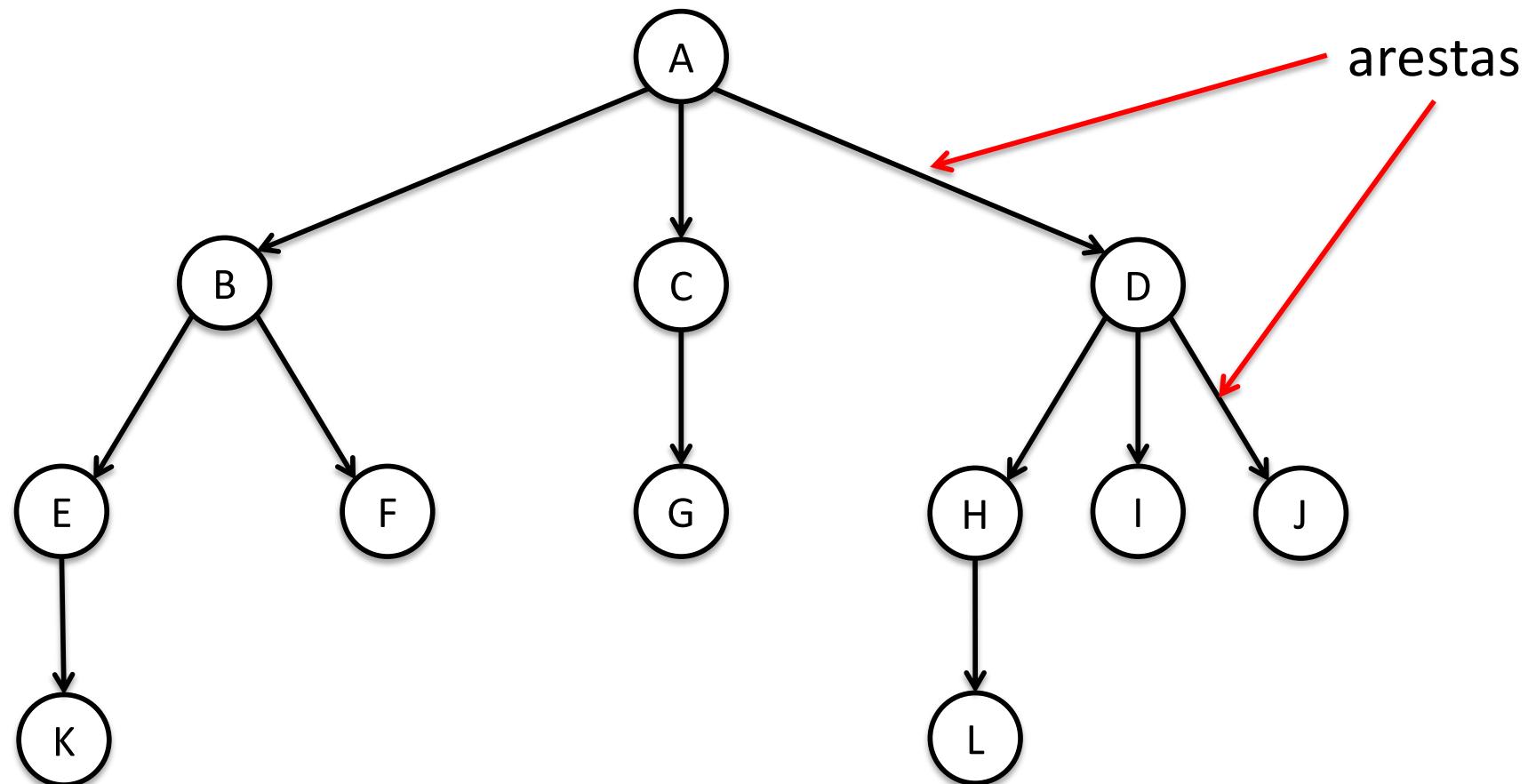
A
..B
....E
.....K
....F
..C
....G
..D
....H
.....L
....I
....J

CONCEITOS BÁSICOS

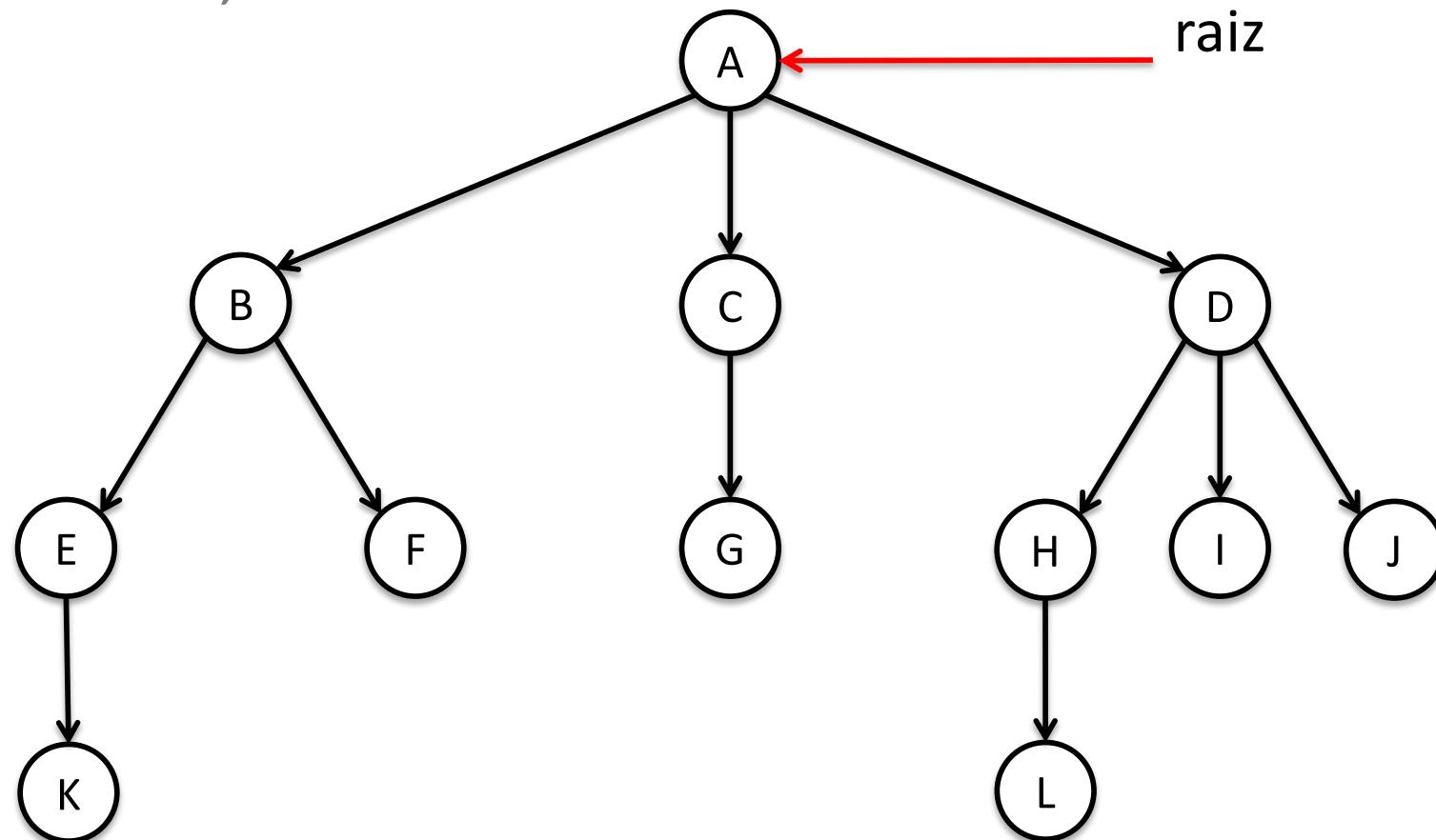
- Esta árvore possui 12 **nós** (ou **vértices**)



- Uma **aresta (arco)** liga um nó a outro



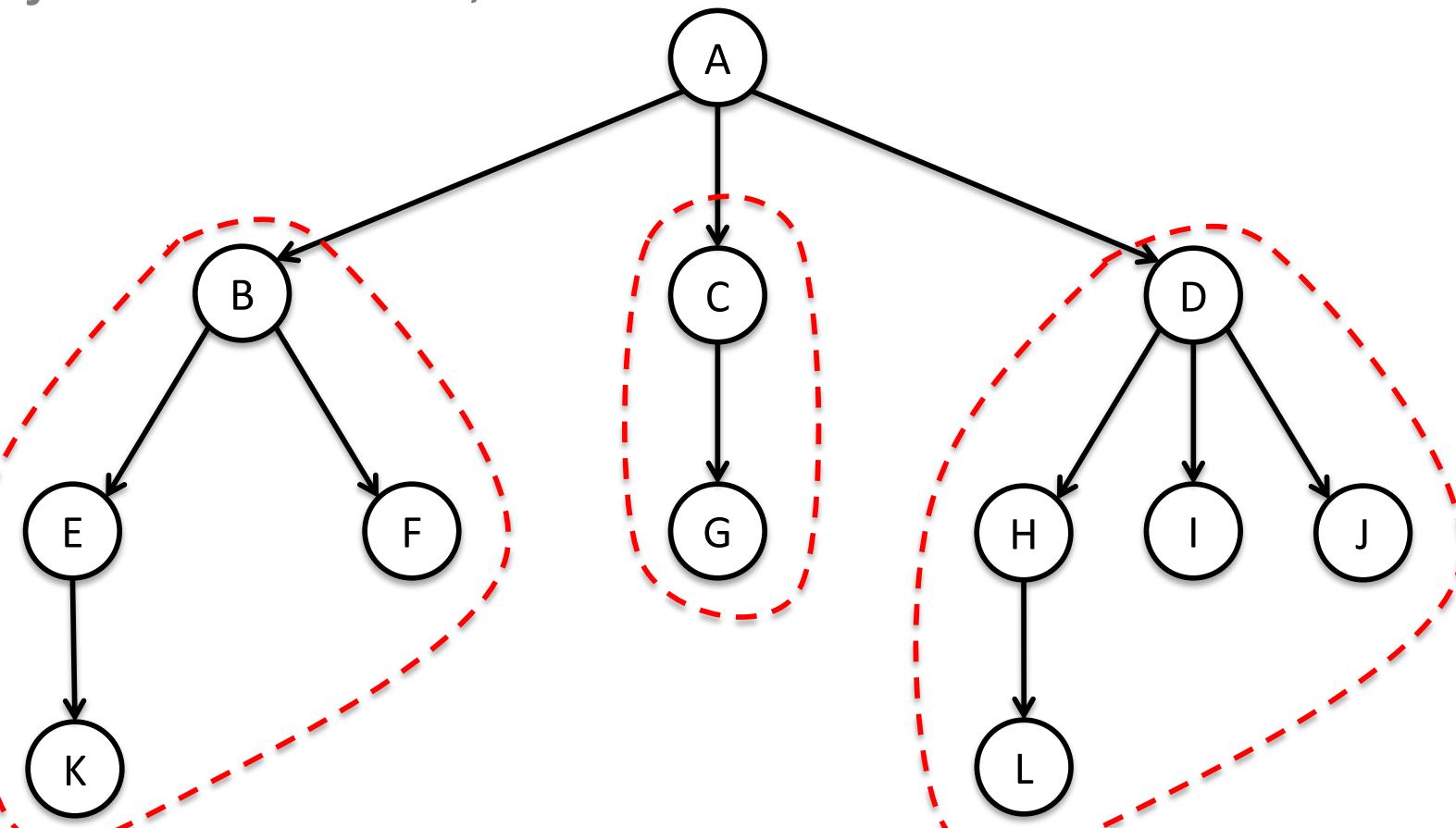
- Normalmente as árvores são desenhadas de forma invertida, com a **raiz** em cima



Conceitos Básicos

Subárvores

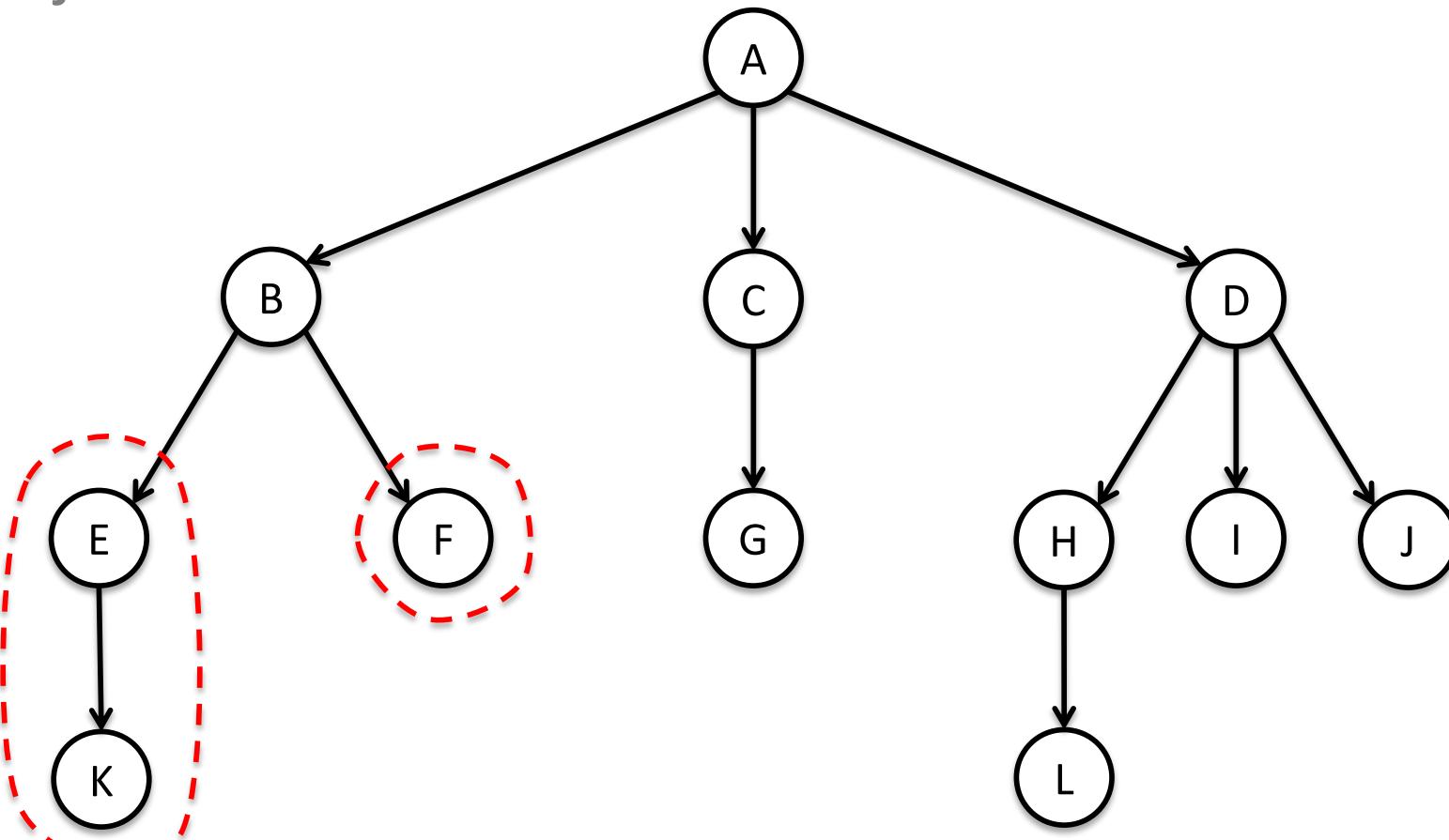
- No exemplo, o nó A possui três **subárvores** (*ramos*) cujas raízes são B, C e D



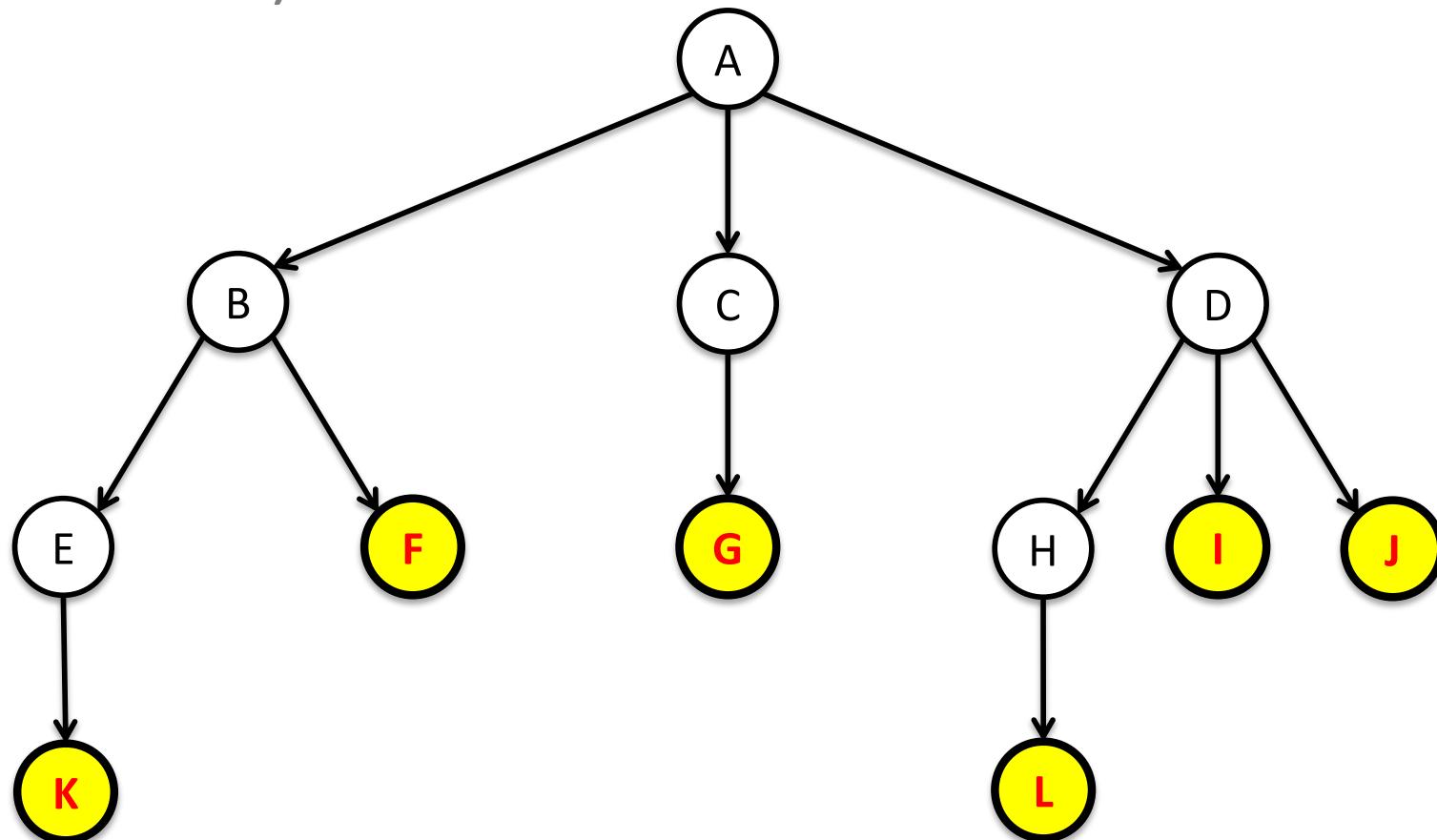
Conceitos Básicos

Subárvores

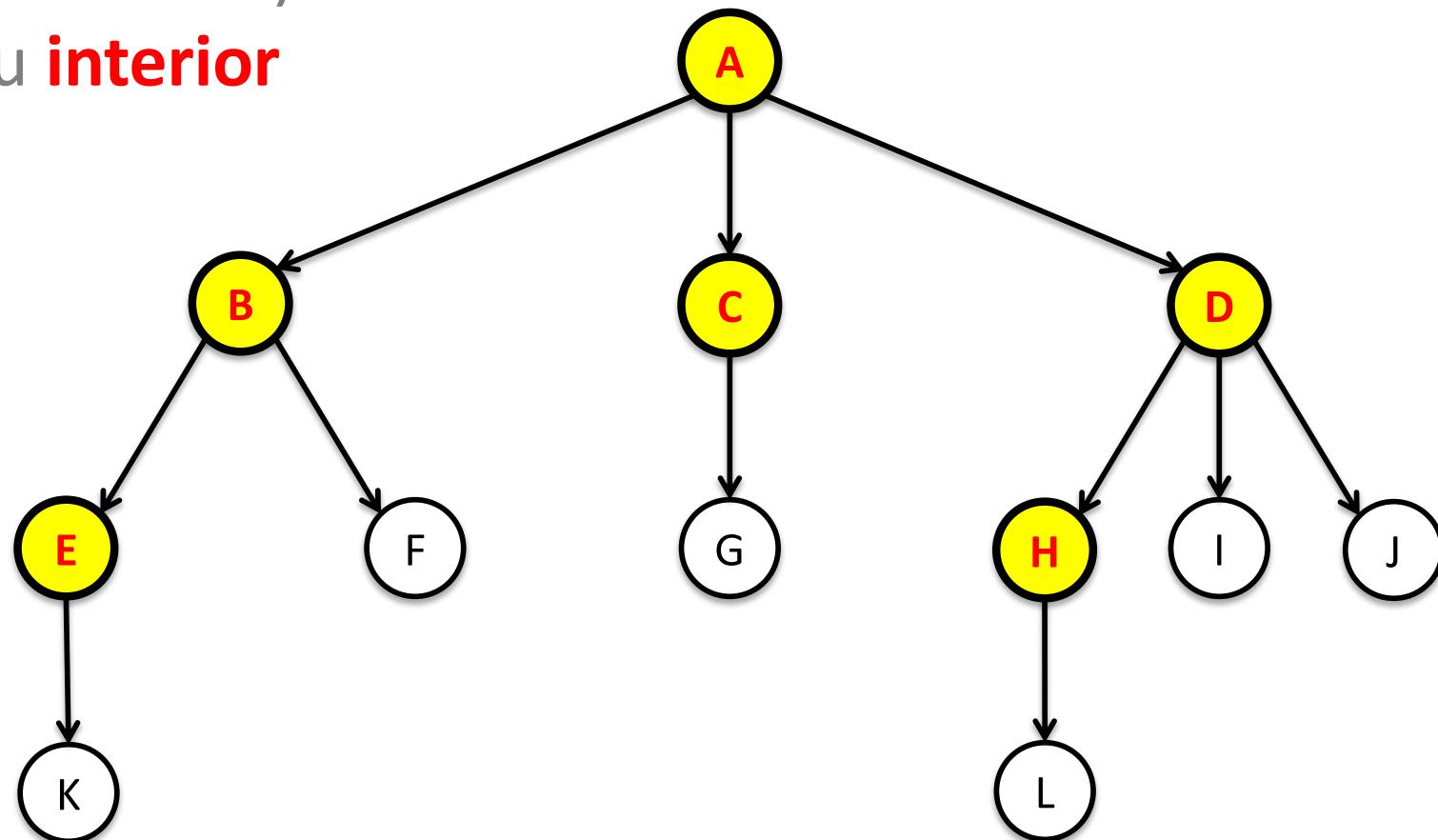
- No exemplo, o nó B possui duas **subárvores** (*ramos*) cujas raízes são E e F



- Um **nó** sem *descendentes* (sem *filhos* ou sem *sucessores*) é denominado **terminal** ou **folha**



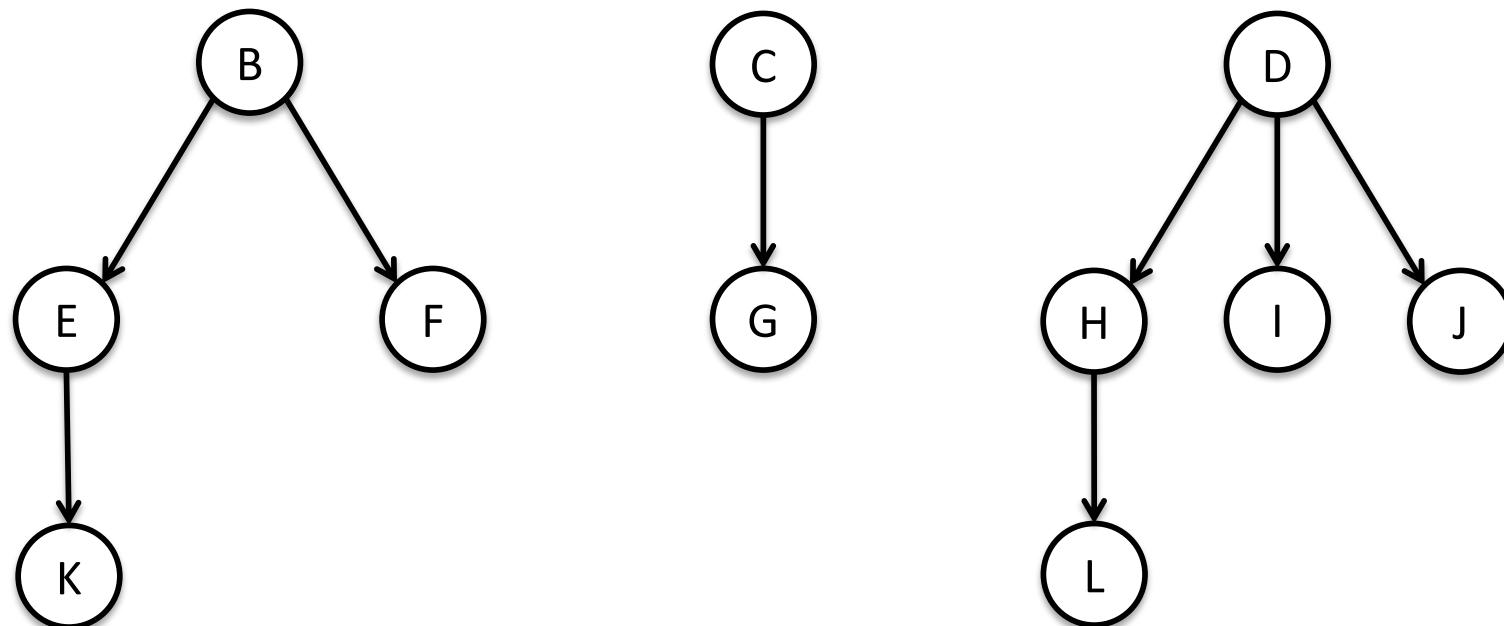
- Um **nó** com *descendentes* (com *filhos* ou com *sucessores*) é denominado **não-terminal** ou **não-folha** ou **interior**



Conceitos Básicos

Floresta

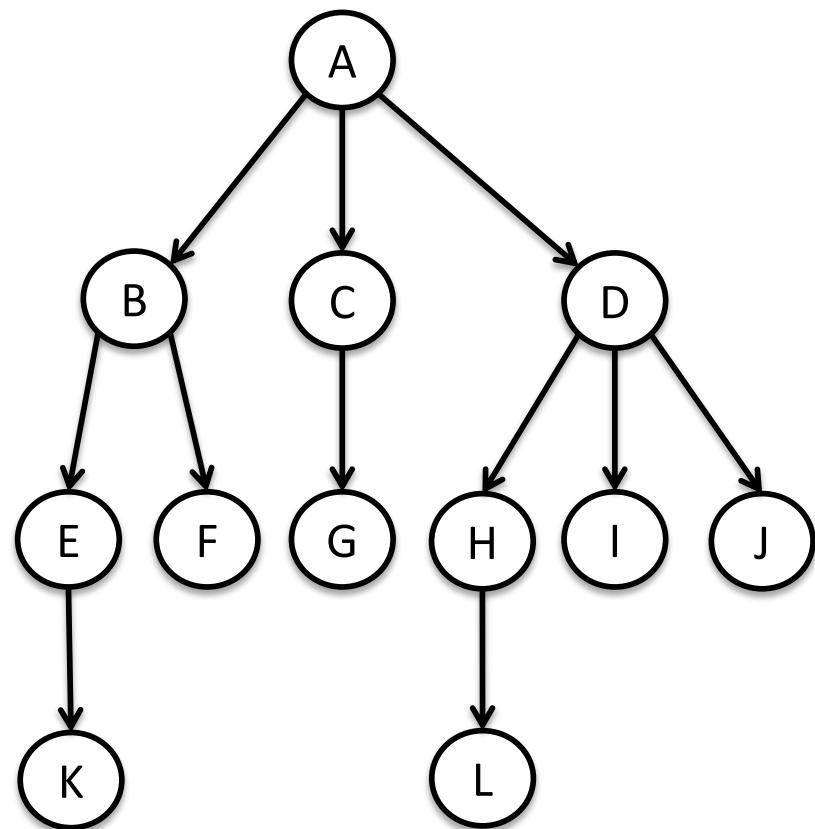
- Uma **floresta** é um conjunto de zero ou mais árvores
- No exemplo, temos 3 árvores que compõem uma floresta



Conceitos Básicos

Grau de um Nó

- O número de descendentes (immediatos) de um nó é denominado **grau** deste nó

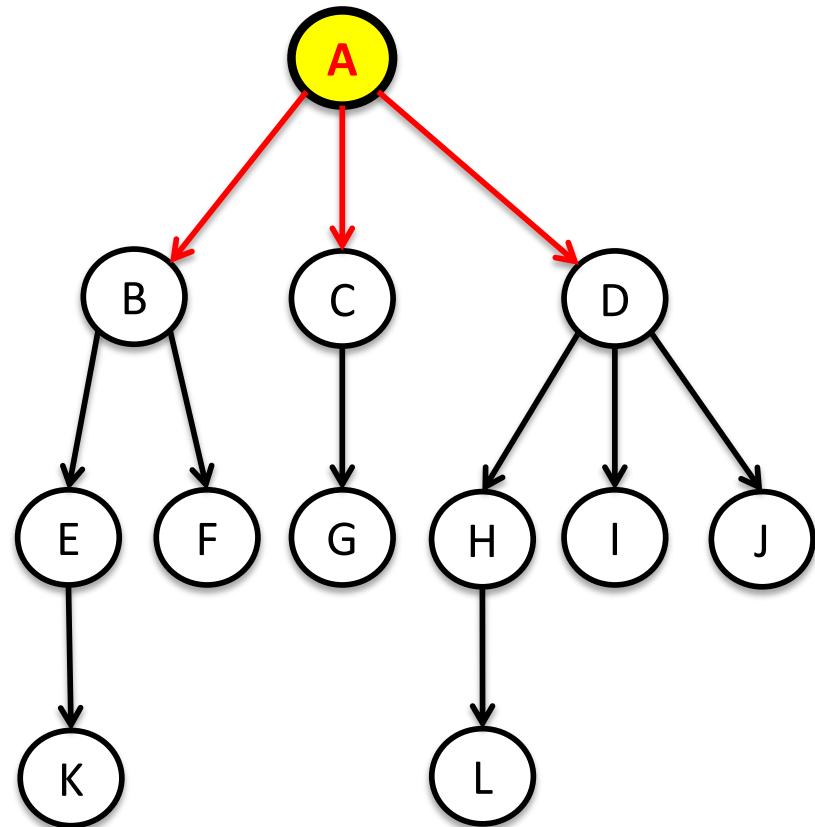


Nó	Grau
A	
B	
C	
D	
E	
F	
G	
H	
I	
J	
K	
L	

Conceitos Básicos

Grau de um Nó

- O número de descendentes (immediatos) de um nó é denominado **grau** deste nó

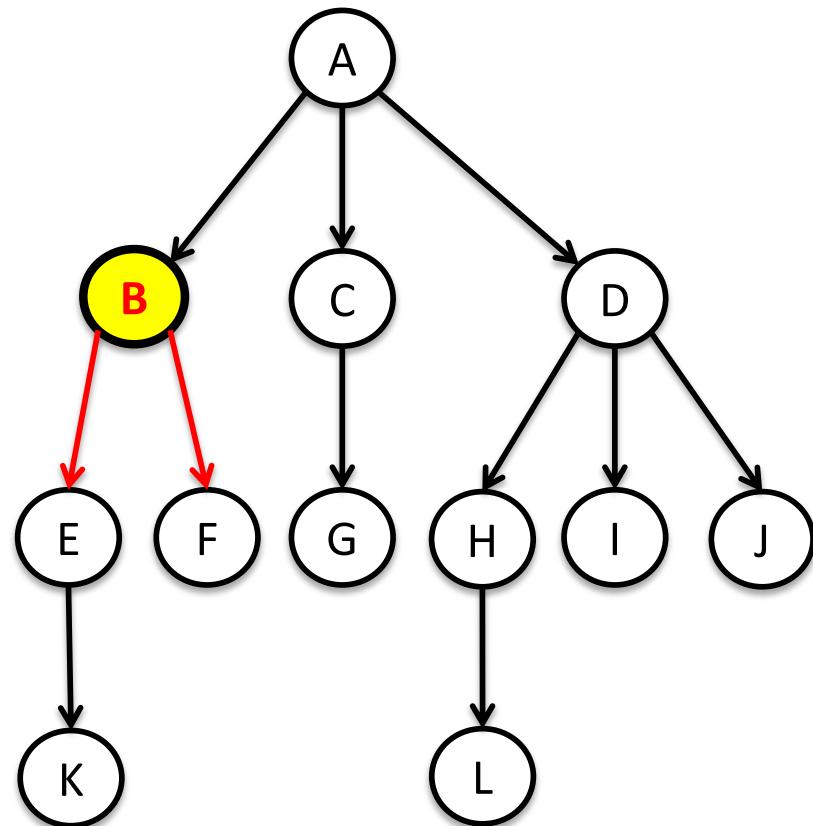


Nó	Grau
A	3
B	
C	
D	
E	
F	
G	
H	
I	
J	
K	
L	

Conceitos Básicos

Grau de um Nó

- O número de descendentes (immediatos) de um nó é denominado **grau** deste nó

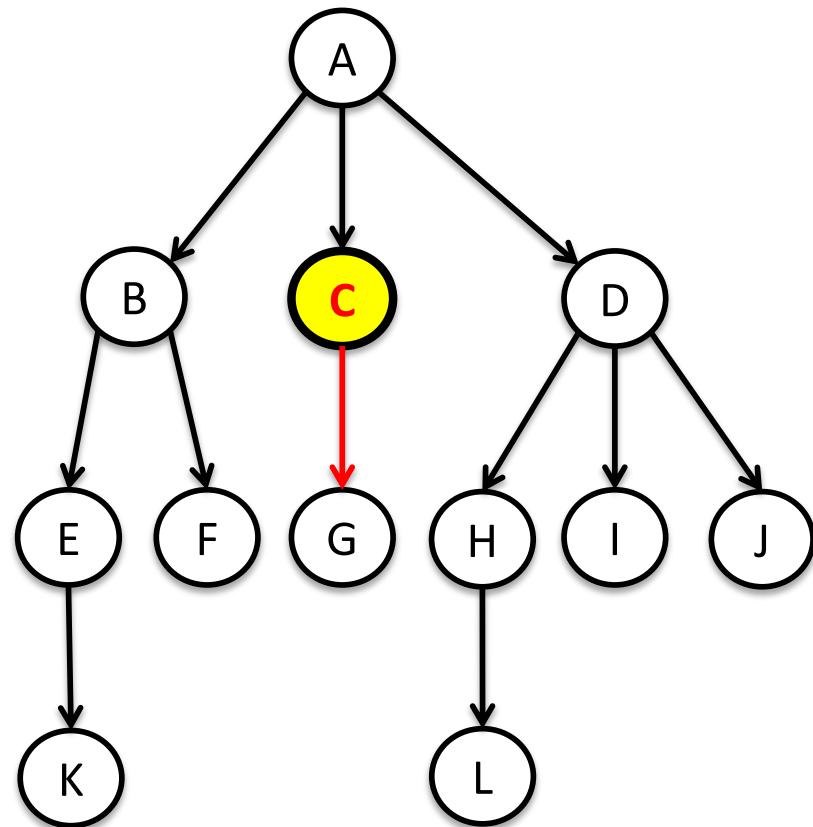


Nó	Grau
A	3
B	2
C	
D	
E	
F	
G	
H	
I	
J	
K	
L	

Conceitos Básicos

Grau de um Nó

- O número de descendentes (immediatos) de um nó é denominado **grau** deste nó

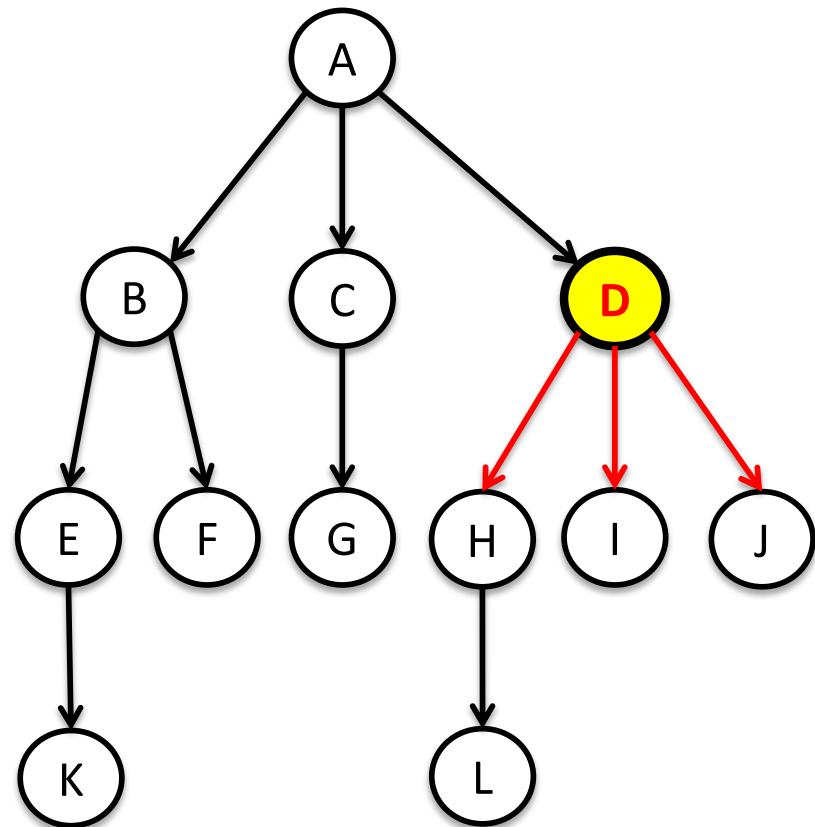


Nó	Grau
A	3
B	2
C	1
D	
E	
F	
G	
H	
I	
J	
K	
L	

Conceitos Básicos

Grau de um Nó

- O número de descendentes (immediatos) de um nó é denominado **grau** deste nó

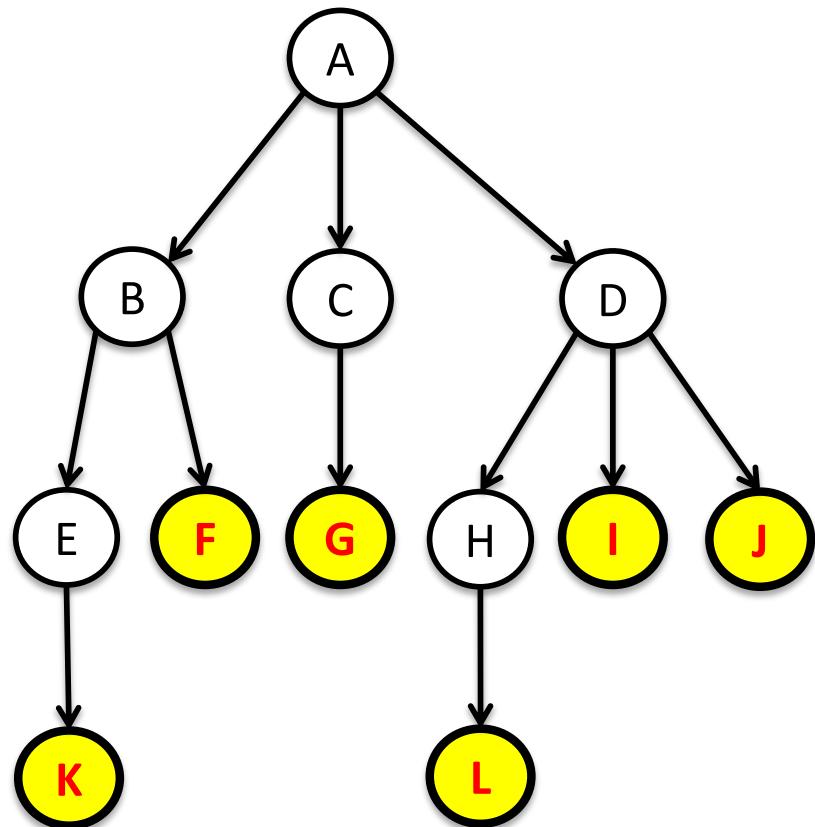


Nó	Grau
A	3
B	2
C	1
D	3
E	
F	
G	
H	
I	
J	
K	
L	

Conceitos Básicos

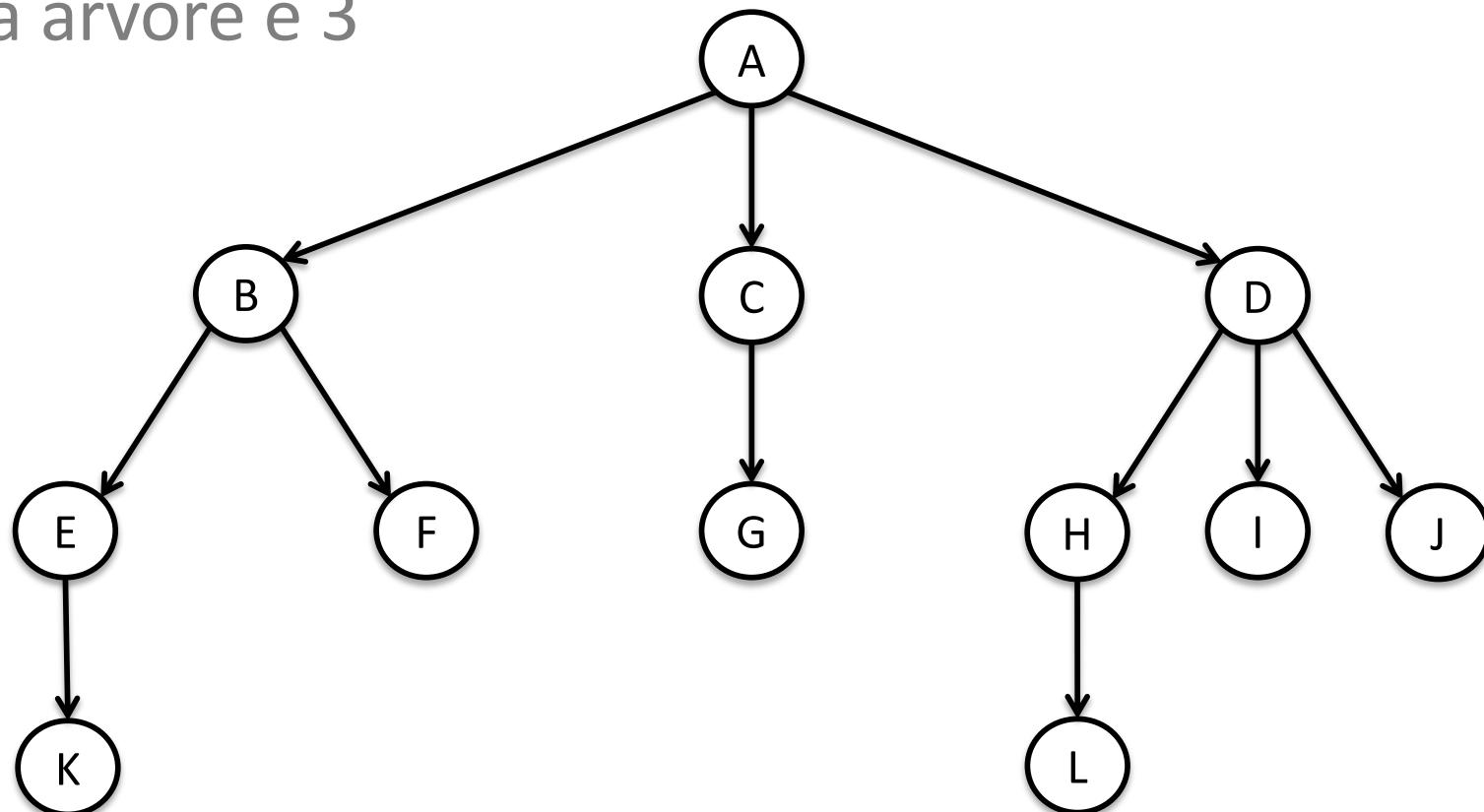
Grau de um Nó

- O **grau** de uma folha é zero

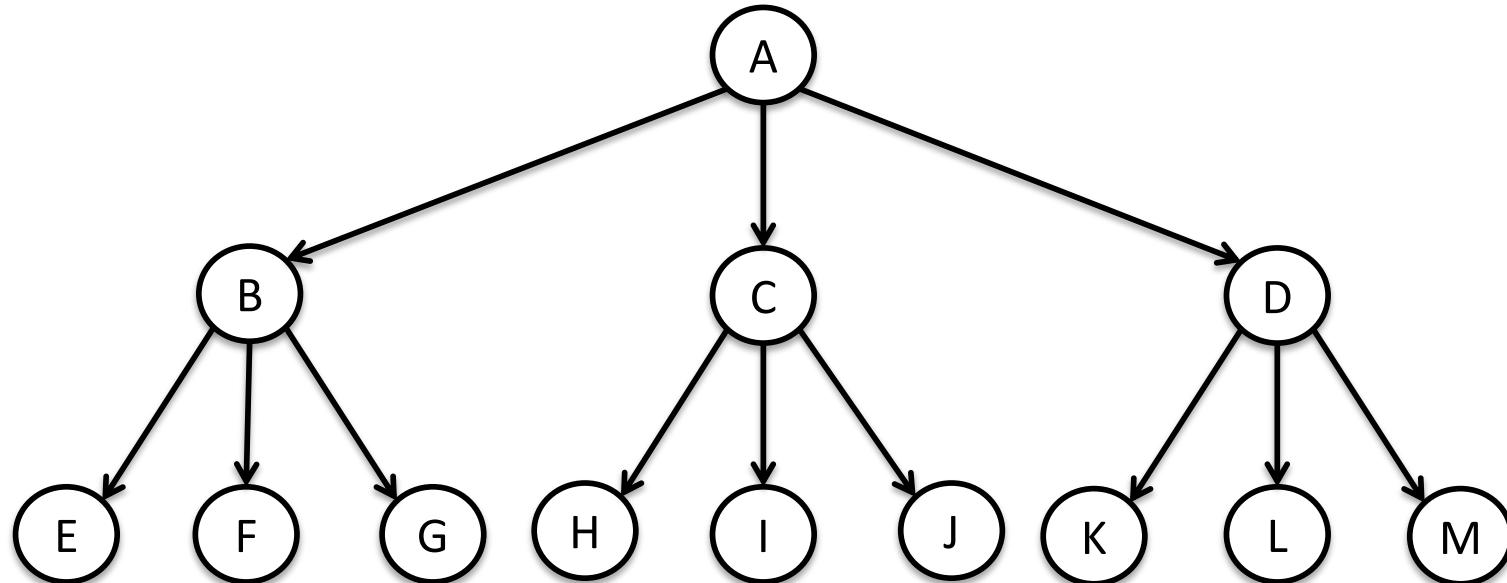


Nó	Grau
A	3
B	2
C	1
D	3
E	1
F	0
G	0
H	1
I	0
J	0
K	0
L	0

- O grau máximo atingido pelos nós de uma árvore é denominado **grau** desta árvore. No exemplo, o grau da árvore é 3



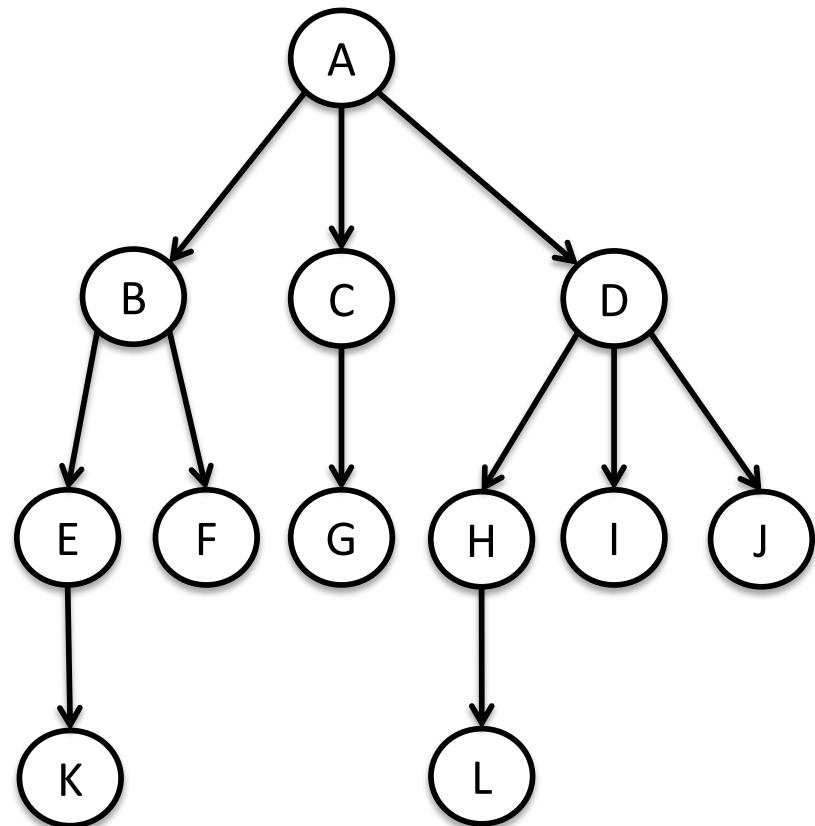
- Uma árvore de grau **d** é uma árvore **completa** se:
 - Todos os nós têm exatamente **d** filhos, exceto as folhas e
 - Todas as folhas estão na mesma altura



Conceitos Básicos

Pai

- As raízes das subárvores de um nó **X** são os **filhos** de **X**; **X** é o **pai** dos **filhos**

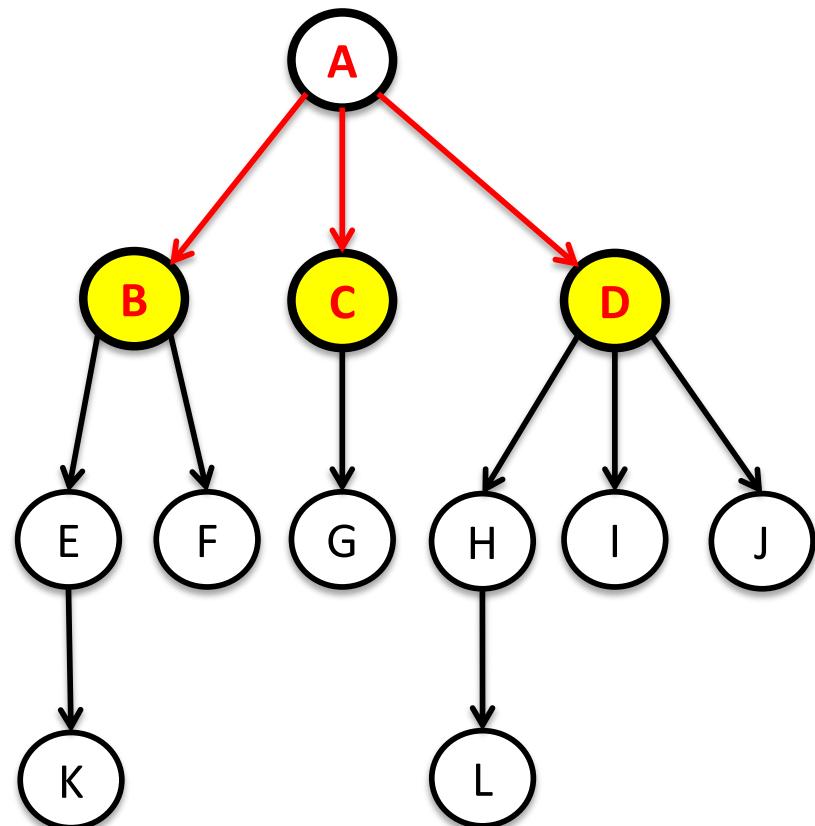


Nó Pai	Nós Filhos
A	
B	
C	
D	
E	
F	
G	
H	
I	
J	
K	
L	

Conceitos Básicos

Pai

- As raízes das subárvores de um nó **X** são os **filhos** de **X**; **X** é o **pai** dos **filhos**

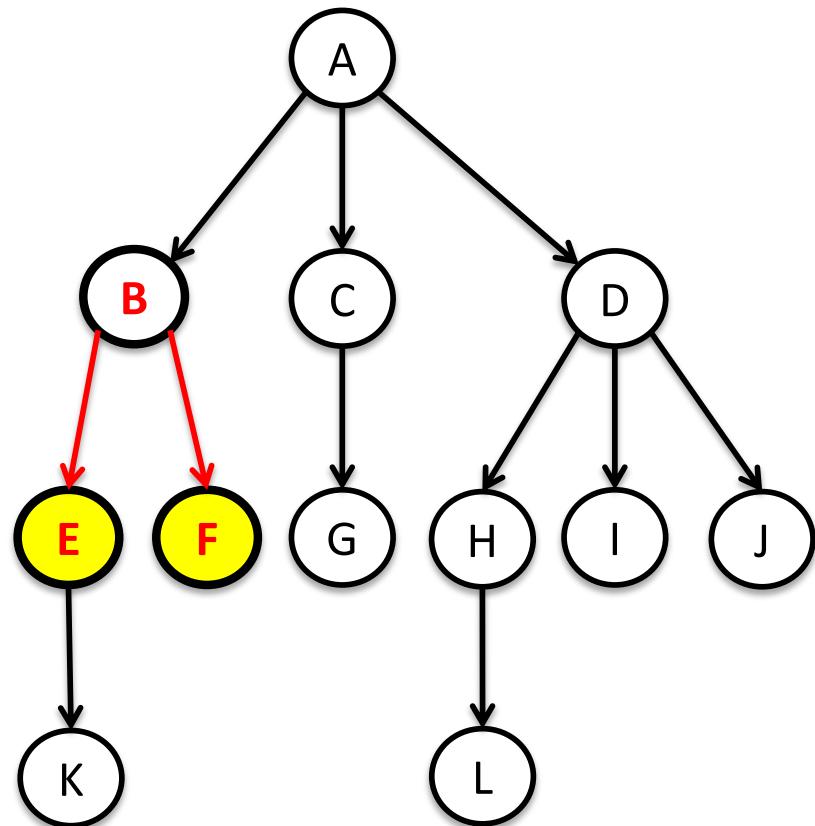


Nó Pai	Nós Filhos
A	B, C, D
B	
C	
D	
E	
F	
G	
H	
I	
J	
K	
L	

Conceitos Básicos

Pai

- As raízes das subárvores de um nó **X** são os **filhos** de **X**; **X** é o **pai** dos **filhos**

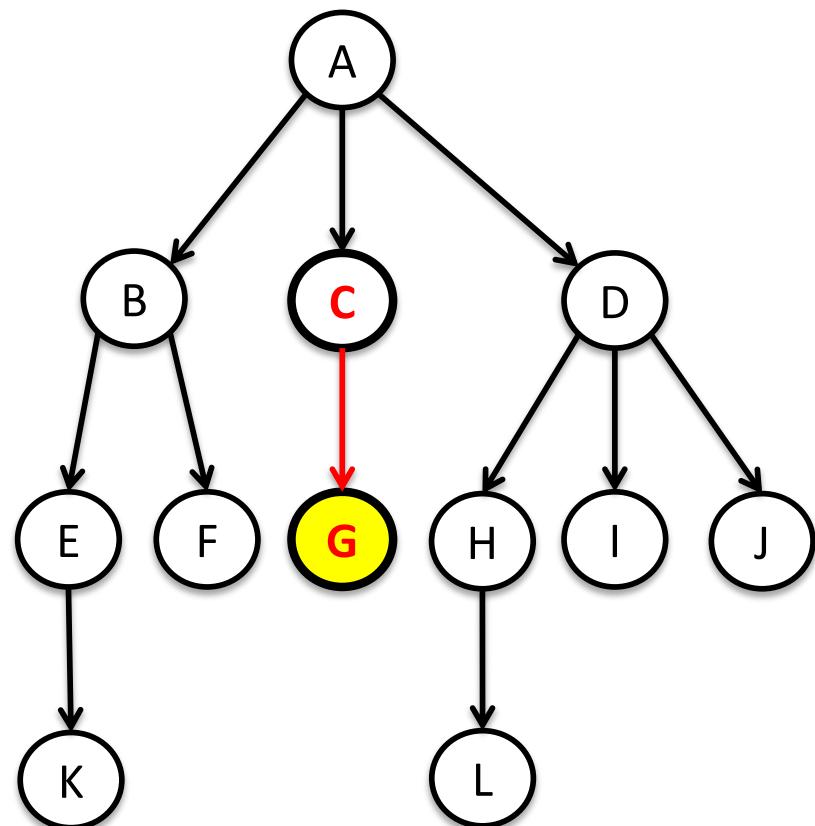


Nó Pai	Nós Filhos
A	B, C, D
B	E, F
C	
D	
E	
F	
G	
H	
I	
J	
K	
L	

Conceitos Básicos

Pai

- As raízes das subárvores de um nó **X** são os **filhos** de **X**; **X** é o **pai** dos **filhos**

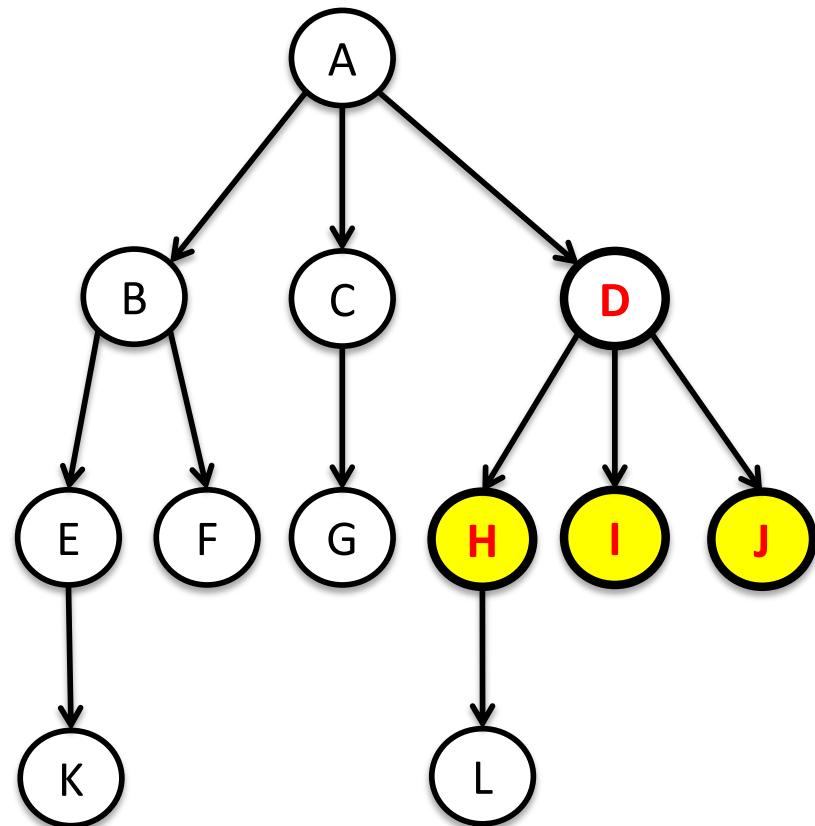


Nó Pai	Nós Filhos
A	B, C, D
B	E, F
C	G
D	
E	
F	
G	
H	
I	
J	
K	
L	

Conceitos Básicos

Pai

- As raízes das subárvores de um nó **X** são os **filhos** de **X**; **X** é o **pai** dos **filhos**

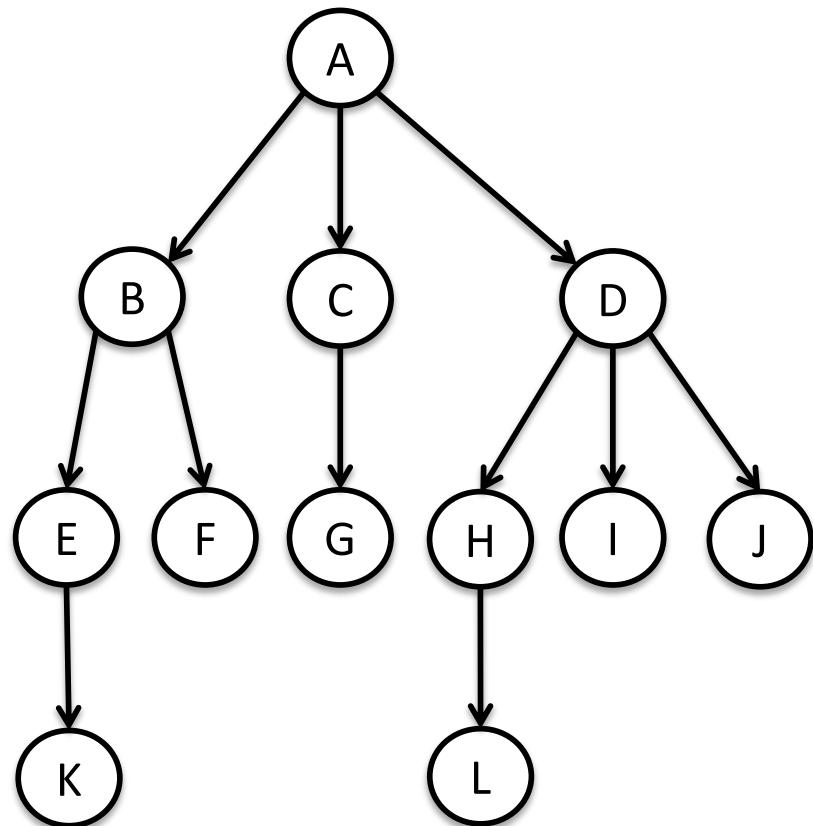


Nó Pai	Nós Filhos
A	B, C, D
B	E, F
C	G
D	H, I, J
E	
F	
G	
H	
I	
J	
K	
L	

Conceitos Básicos

Pai

- As raízes das subárvores de um nó **X** são os **filhos** de **X**; **X** é o **pai** dos **filhos**

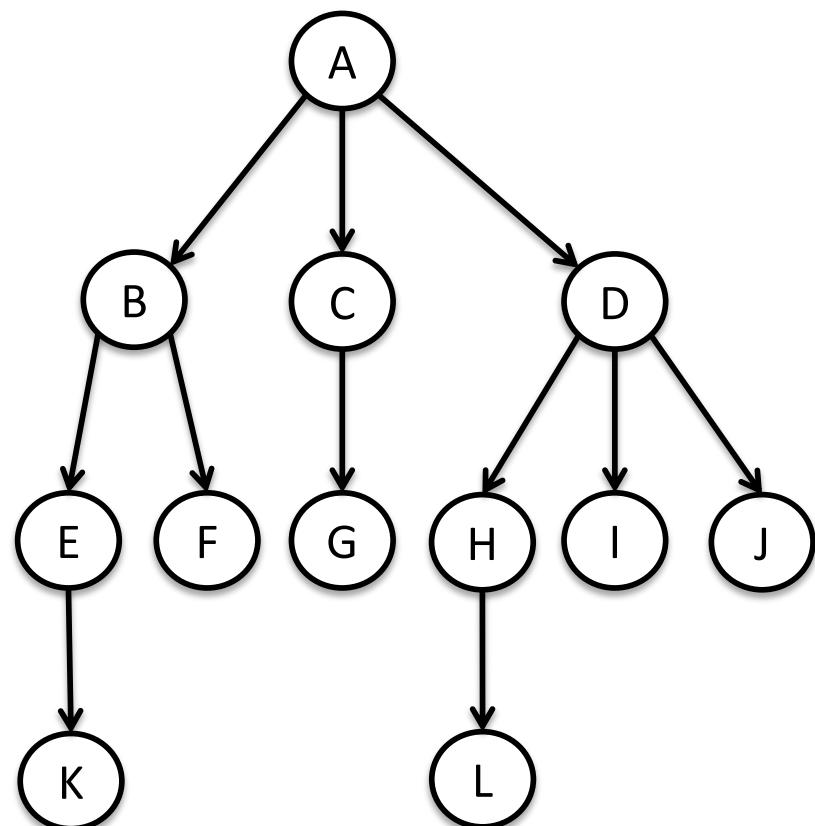


Nó Pai	Nós Filhos
A	B, C, D
B	E, F
C	G
D	H, I, J
E	K
F	-
G	-
H	L
I	-
J	-
K	-
L	-

Conceitos Básicos

Irmão

- Os filhos (descendentes) de um mesmo nó pai (antecessor) são denominados **irmãos**

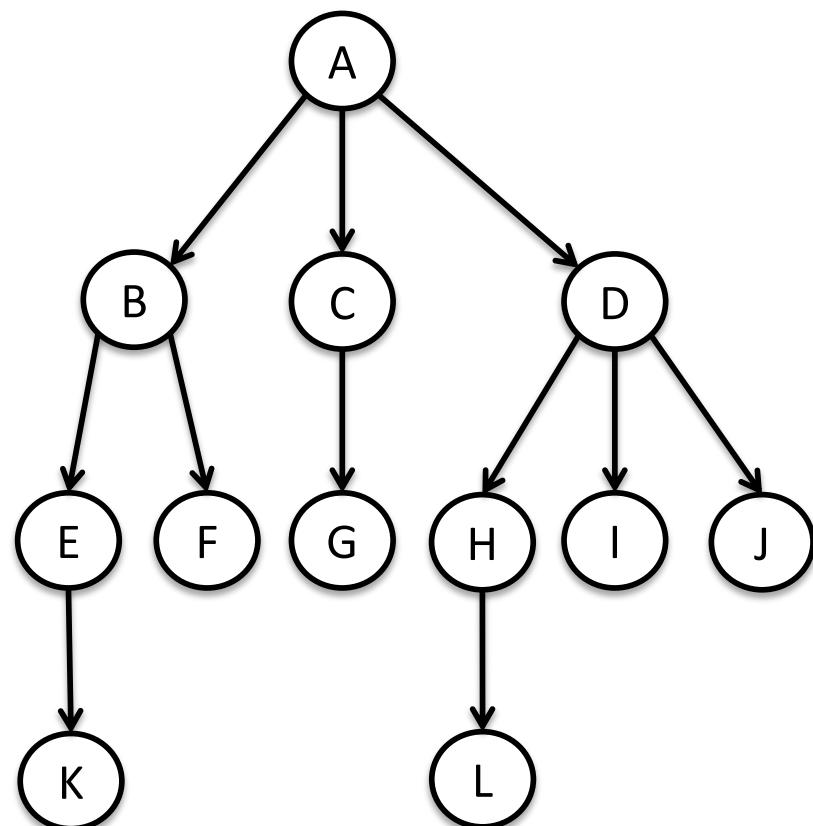


Irmãos
B, C, D
E, F
H, I, J

Conceitos Básicos

Avô e demais Parentes

- Podemos estender essa terminologia para **avô**, **bisavô**, e demais parentes



Nós	Avô
E, F, G, H, I, J	A
K	B
L	D

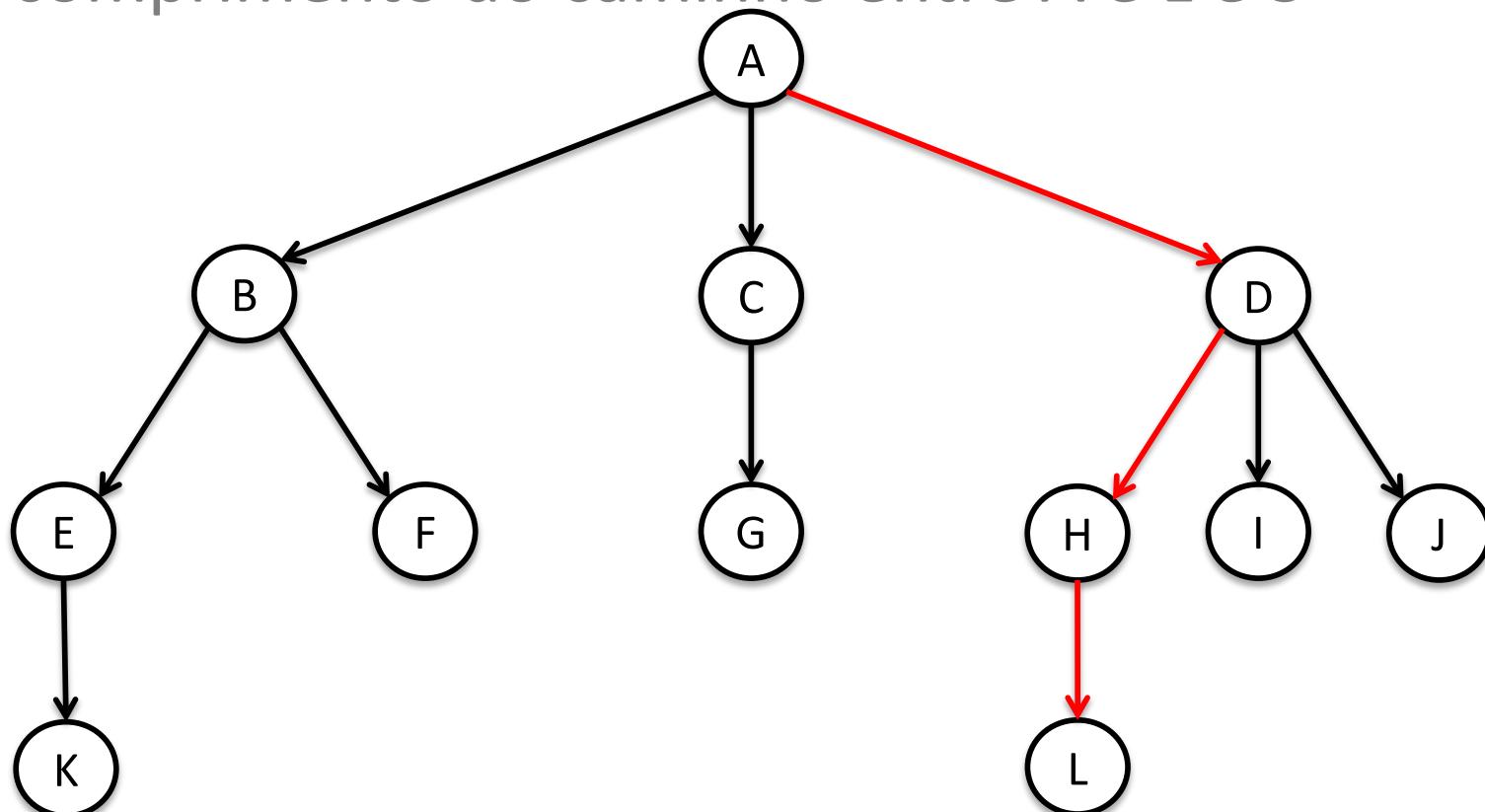
Nós	Bisavô
K, L	A

- Uma sequência de nós distintos v_1, v_2, \dots, v_k tal que sempre existe a relação
 - “ v_{i+1} é filho de v_i ” ou “ v_i é pai de v_{i+1} ”, $1 \leq i < k$ é denominada um **caminho** entre v_1 e v_k
- Diz-se que v_1 alcança v_k ou que v_k é alcançado por v_1
- Um caminho de k vértices v_1, v_2, \dots, v_k é formado pela sequência de $k-1$ pares de nós $(v_1, v_2), (v_2, v_3), \dots, (v_{k-2}, v_{k-1}), (v_{k-1}, v_k)$
 - $k-1$ é o **comprimento** do caminho
 - Cada par (v_i, v_{i+1}) é uma **aresta** ou **arco**, $1 \leq i < k$

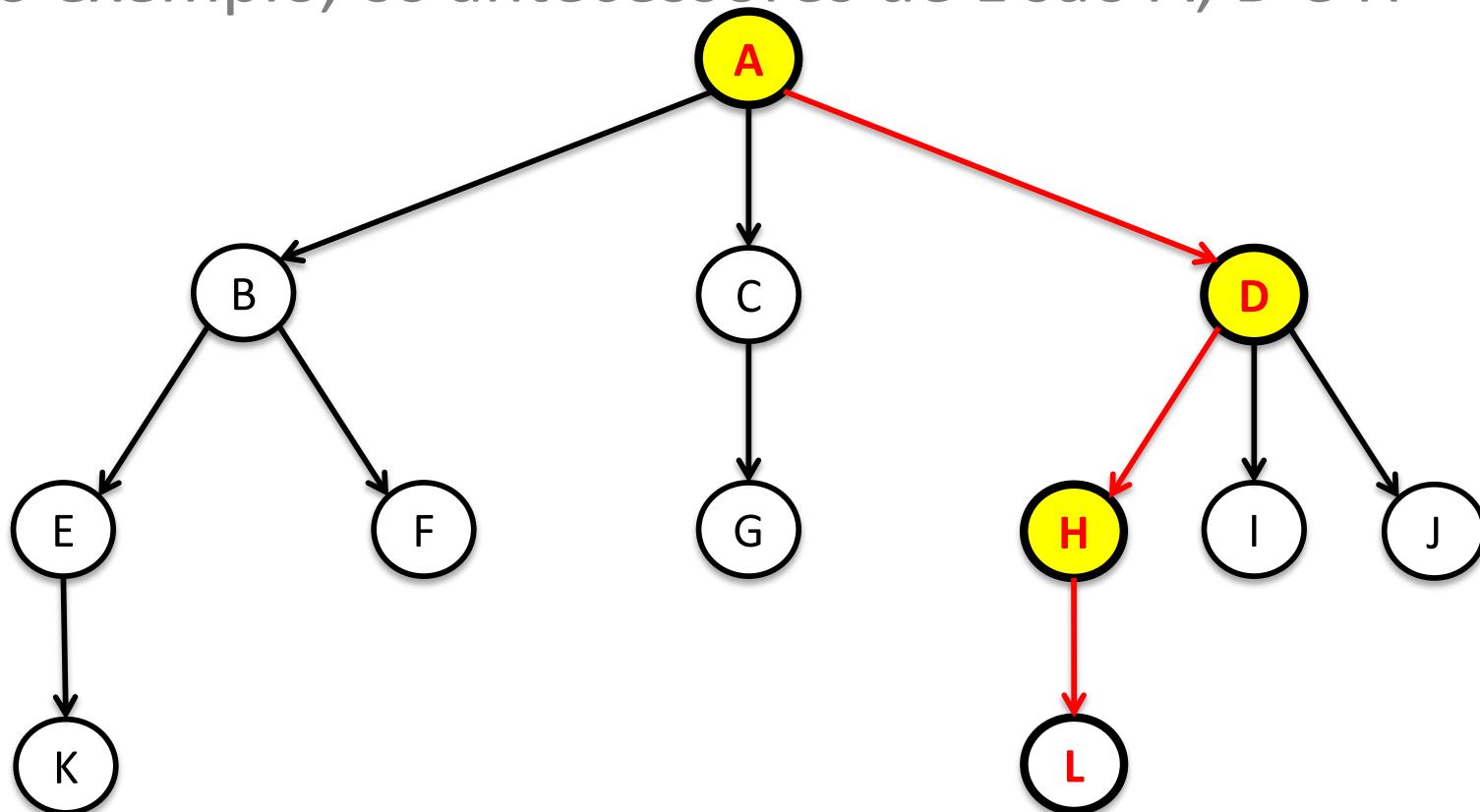
Conceitos Básicos

Caminho

- A, D, H, L é um caminho entre A e L, formado pela sequência de arestas (A, D), (D, H), (H, L)
- O comprimento do caminho entre A e L é 3



- Os **antecessores (antepassados)** de um nó são todos os nós no caminho entre a raiz e o respectivo nó
- No exemplo, os antecessores de L são A, D e H

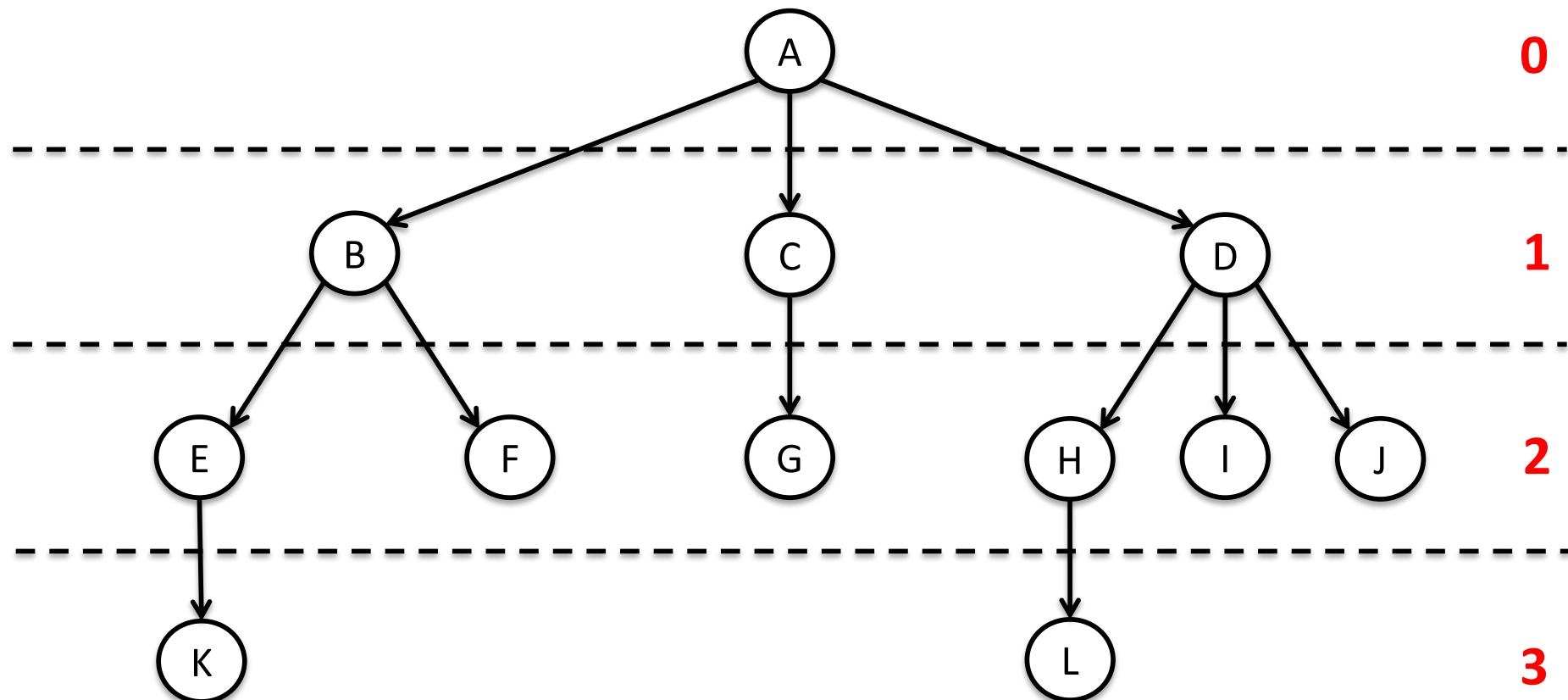


- O **nível** (ou **profundidade**) de um nó é definido admitindo-se que a raiz está no nível zero (nível um)
- Se um nó tem **nível i**, seus filhos terão **nível i+1**
- Não existe um padrão quanto ao nível adotado para a raiz, que determina o nível dos demais nós
- Assim, a raiz pode ser admitida como estando
 - No **nível zero**
 - Alternativamente, no **nível um**
- No restante desta aula, vamos adotar a raiz no nível 0

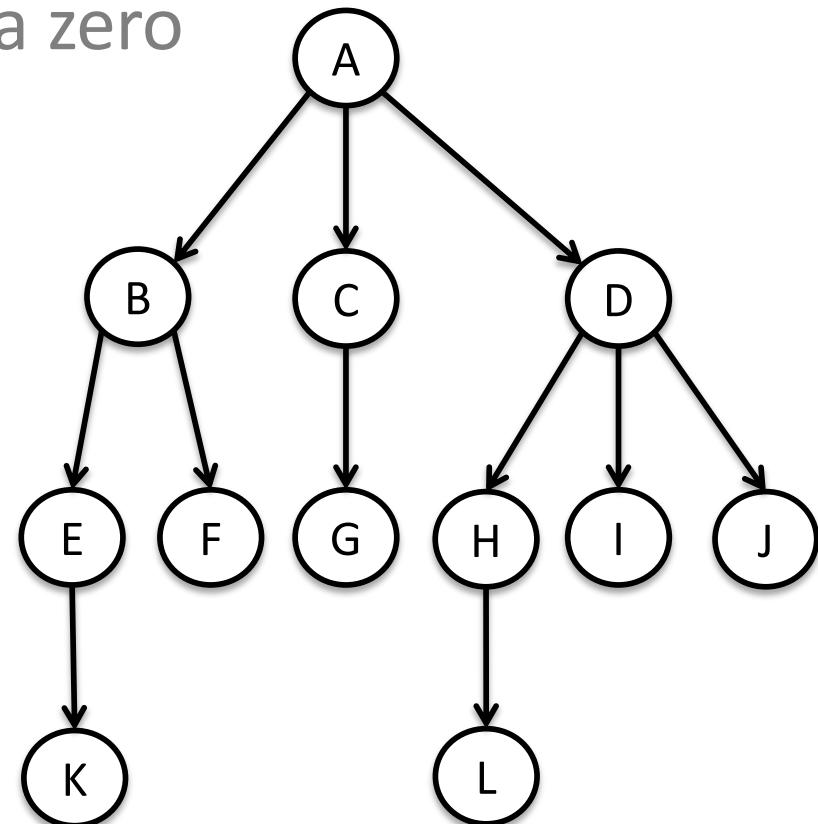
Conceitos Básicos

Nível

- B, C e D estão no nível 1
- K e L estão no nível 3



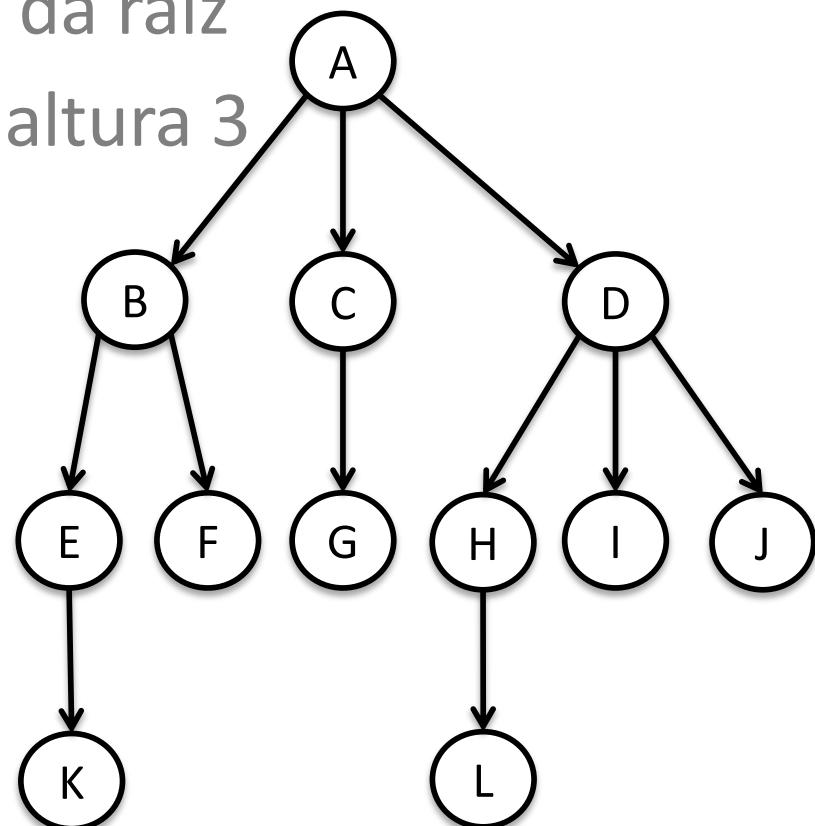
- A **altura de um nó** é o número de arestas no maior caminho desde o nó até um de seus descendentes
- Portanto, as folhas têm altura zero
- No exemplo, os nós:
 - K, F, G, L, I, J têm altura 0
 - E, C, H têm altura 1
 - B e D têm altura 2
 - A tem altura 3



Conceitos Básicos

Altura de uma Árvore

- A **altura** (ou **profundidade**) **de uma árvore** é o nível máximo entre todos os nós da árvore ou, equivalentemente, é a altura da raiz
- No exemplo, a árvore possui altura 3



Conceitos Básicos

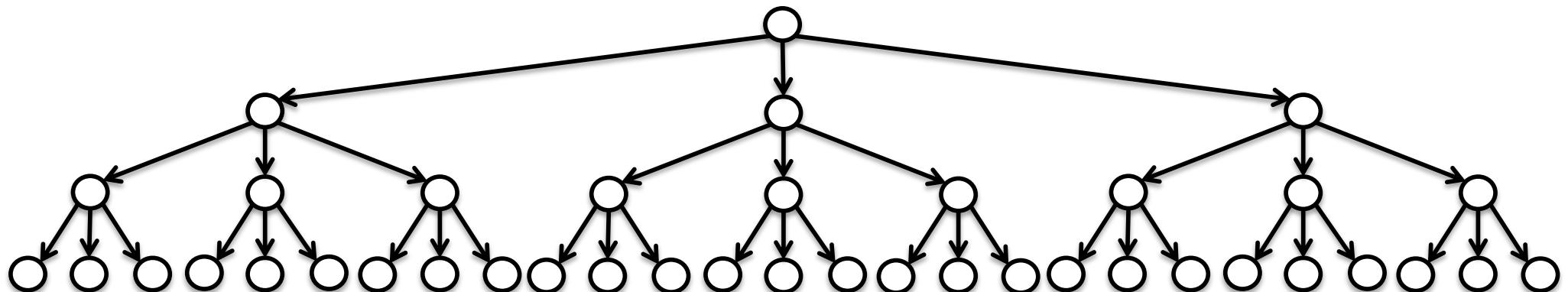
Número Máximo de Nós

- O número máximo de nós $n(h, d)$ em um árvore de altura h é atingido quando todos os nós possuírem d subárvore, exceto o nível h , que não possui subárvore
- Para uma árvore de grau d
 - Nível **0** contém d^0 (um) nó (raiz)
 - Nível **1** contém d^1 descendentes da raiz
 - Nível **2** contém d^2 descendentes
 - ...
 - Nível **i** contém d^i descendentes

Conceitos Básicos

Número Máximo de Nós

- Assumindo $d = 3$
 - Nível 0: 1 nó (raiz)
 - Nível 1: 3 nós
 - Nível 2: $3^2 = 9$ nós
 - Nível 3: $3^3 = 27$ nós
- $n(3, 3) = 1 + 3 + 9 + 27 = 40$ nós



Conceitos Básicos

Número Máximo de Nós

- Portanto, o número máximo de nós $n = n(h, d)$ é a soma do número de nós em cada nível, ou seja:

$$n = n(h, d) = \sum_{i=0}^h d^i = d^0 + d^1 + d^2 + \cdots + d^h$$

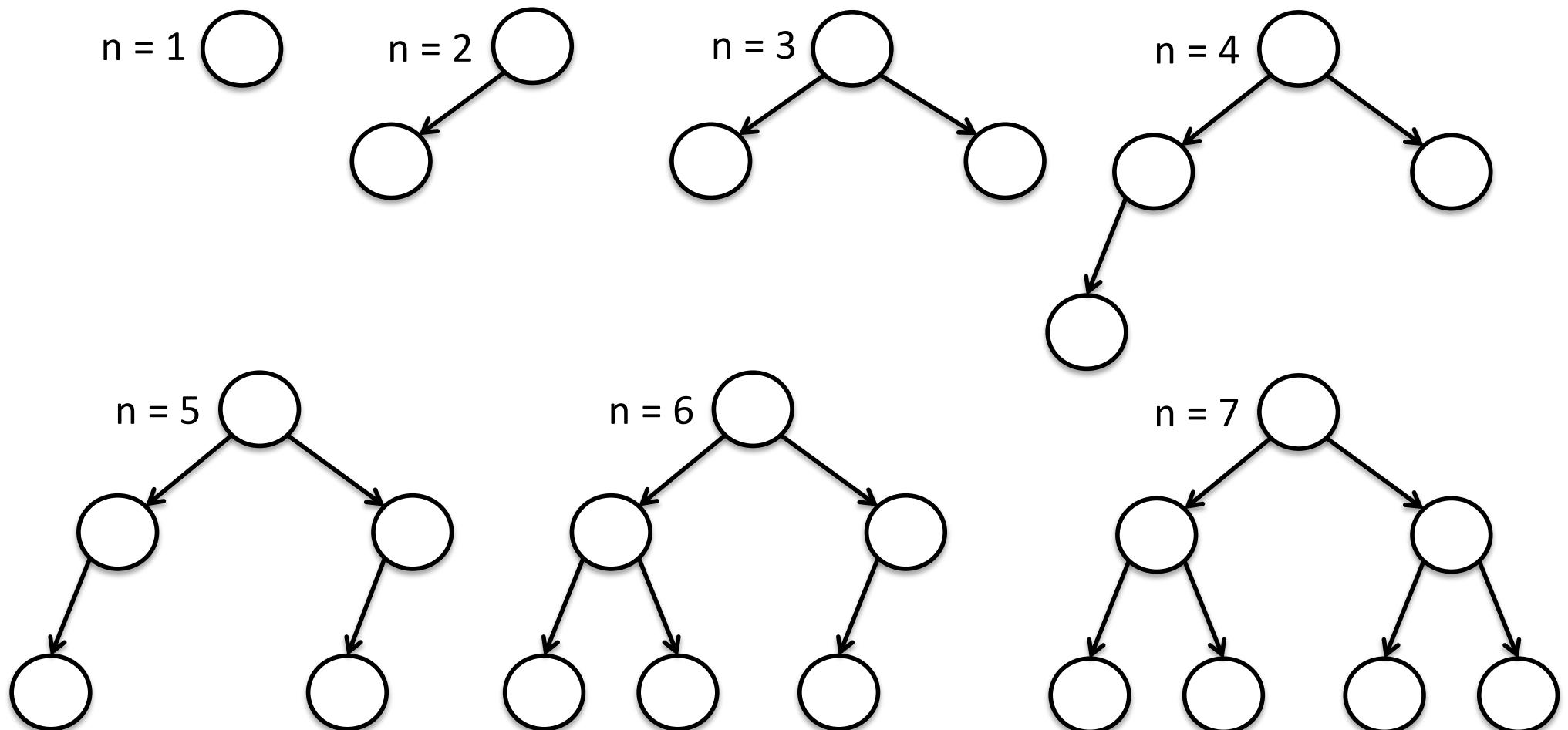
$$\sum_{i=0}^h d^i = \frac{d^{h+1} - 1}{d - 1}, d > 1$$

- Uma árvore é **balanceada** se, para cada nó, a *altura* de suas subárvorese diferem, no máximo, de uma unidade
- Uma árvore é **perfeitamente balanceada** se, para cada nó, os *números de nós* em suas subárvorese diferem, no máximo, de uma unidade
- Todas as árvores perfeitamente balanceadas também são árvores balanceadas

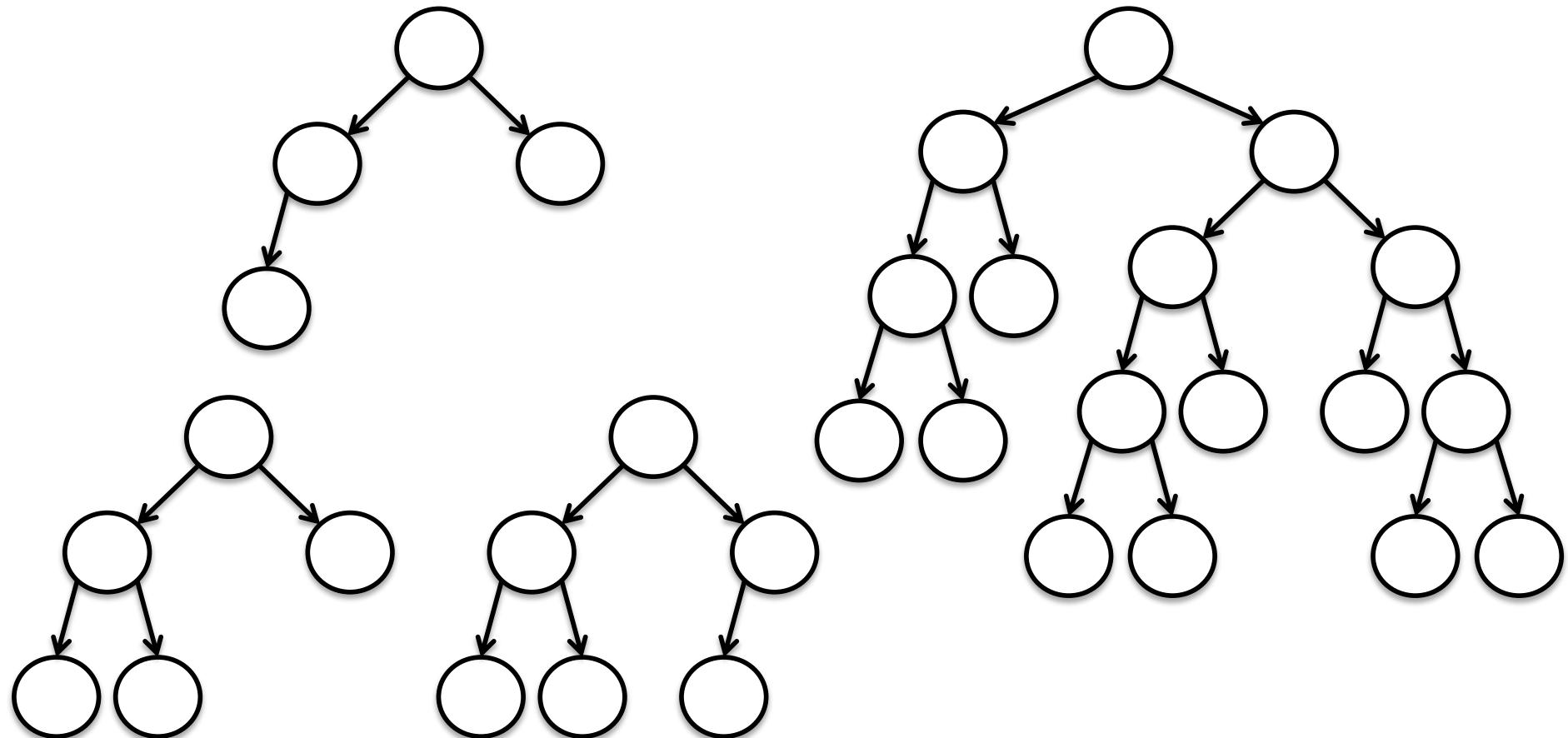
Conceitos Básicos

Árvore Balanceada

- Árvores perfeitamente平衡adas de grau 2



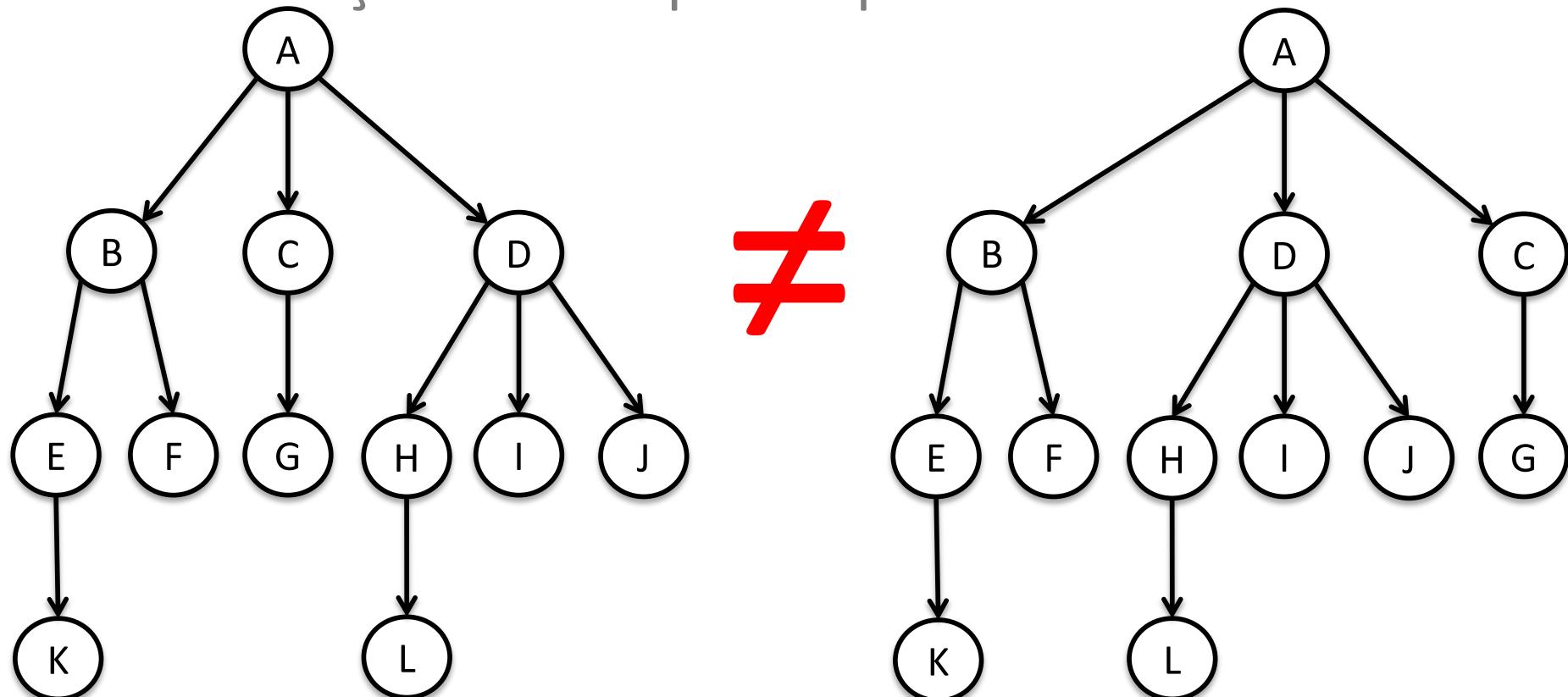
- Árvores平衡adas de grau 2



Conceitos Básicos

Árvore Orientada

- Uma árvore **orientada (ordenada)** é uma árvore na qual os filhos de cada nó são orientados (ordenados)
- A orientação é da esquerda para a direita



TAD ÁRVORE

- **O que o TAD Árvore deveria conter?**
 - Representação do tipo da árvore
 - Conjunto de operações que atuam sobre a árvore

- **Quais operações deveriam fazer parte da TAD?**

- **O que o TAD Árvore deveria conter?**
 - Representação do tipo da árvore
 - Conjunto de operações que atuam sobre a árvore

- **Quais operações deveriam fazer parte da TAD?**

**O conjunto de operações a ser definido
depende de cada aplicação**

- Um conjunto de operações necessário à maioria das aplicações é:
 1. Criar uma árvore sem descendentes
 2. Inserir um novo descendente em uma árvore
 3. Verificar se uma dada árvore possui descendentes
 4. Obter a lista de descendentes de uma dada árvore

Exemplo de Protótipo

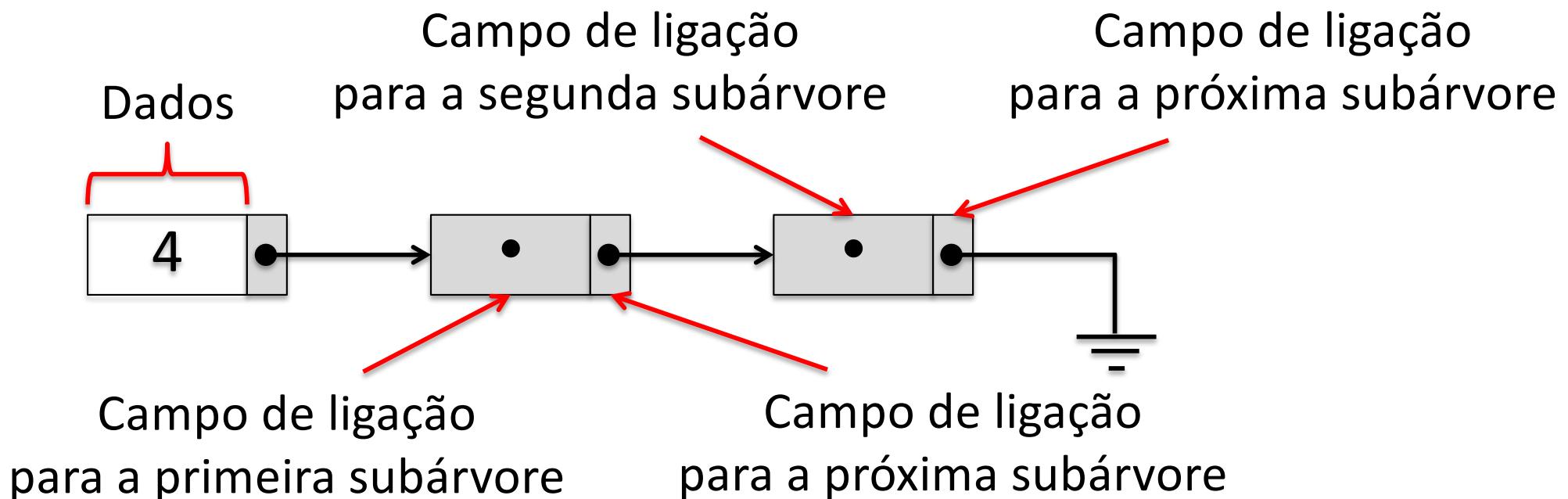
■ Exemplo de Conjunto de Operações

- **TArvore_Inicia(Arvore, x)**: Inicia uma árvore sem descendentes contendo o item x
- **TArvore_EhExterno(Arvore)**: Retorna *true* se a árvore não possuir nenhum descendente; caso contrário, retorna *false*
- **TArvore_Insere(Arvore, s)**: Insere a subárvore enraizada em s como descendente da raiz da árvore
- **TArvore_ListaDescendentes(Arvore)**: Retorna a lista de descendentes da raiz da árvore
- **TArvore_NDescendentes(Árvore)**: Retorna o número de descendentes da raiz da árvore

- Existem várias opções de estruturas de dados que podem ser usadas para representar árvores
- Uma das representações mais utilizadas é:
 - Implementação por meio de **listas lineares**

IMPLEMENTAÇÃO POR LISTAS LINEARES

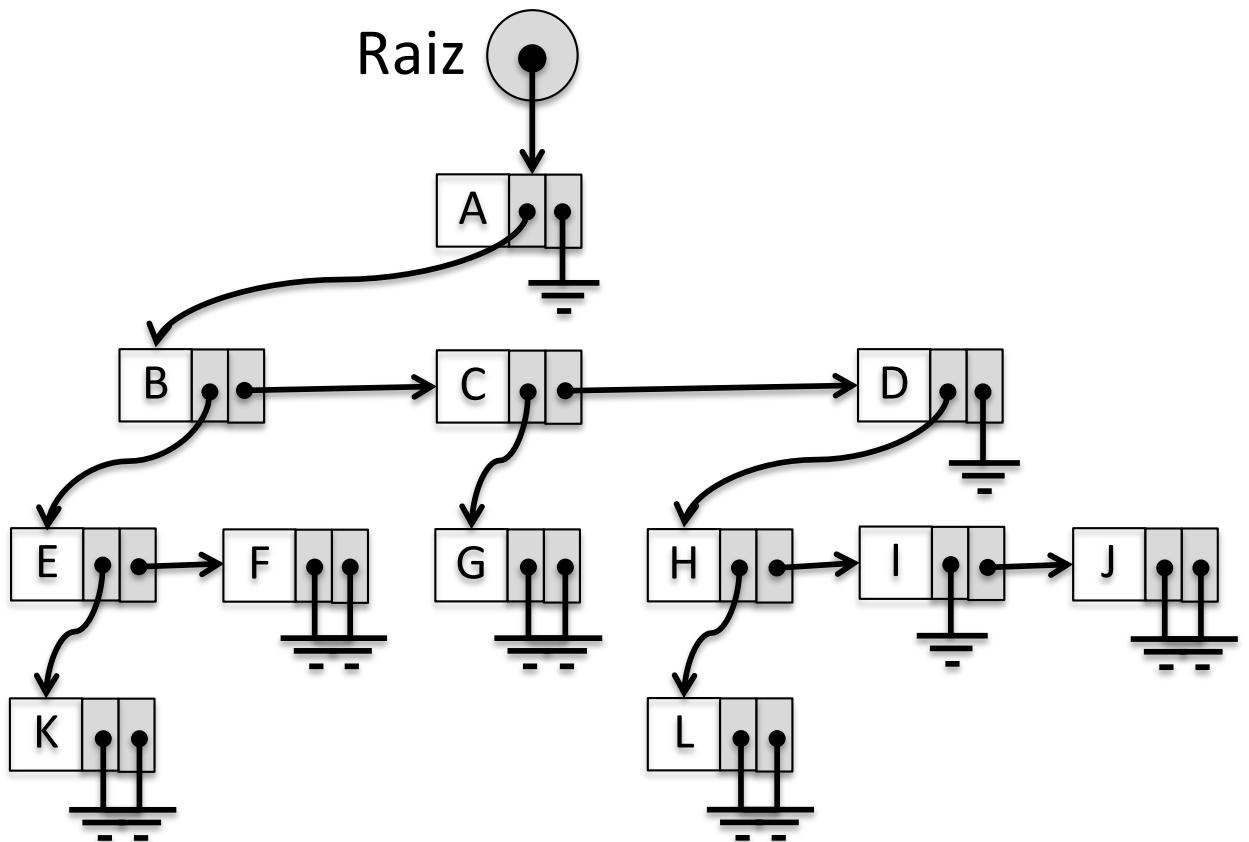
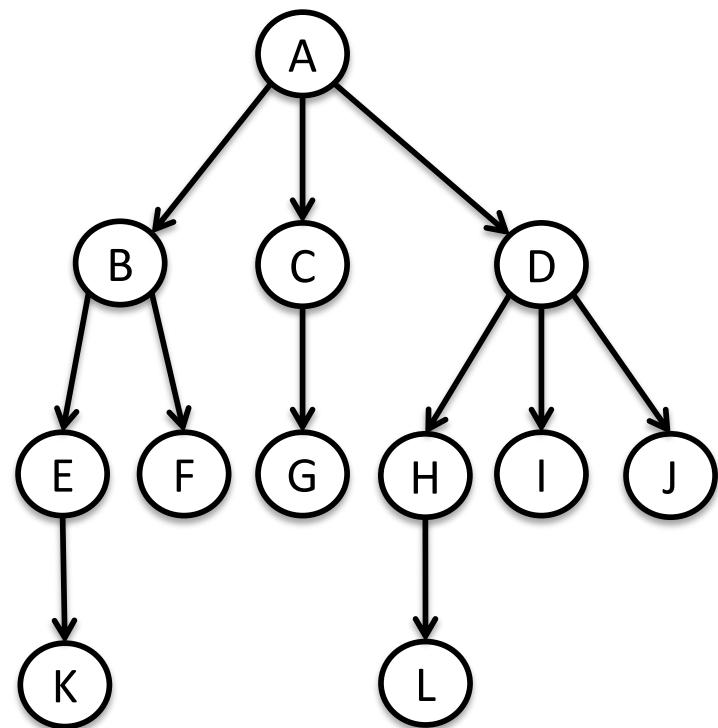
- Árvores podem ser implementadas por meio de listas lineares usando apontadores
 - Cada nó possui um campo de informação e uma série de campos de ligação, de acordo com o número de filhos



Implementação por Listas

Por exemplo, a árvore ternária seguinte ($d = 3$) ...

... pode ser implementada como mostrado abaixo:



Estrutura da Árvore por Listas

```
typedef struct SNo *TArvore;

typedef TArvore TItem;

typedef int TChave;

typedef struct {
    TChave Chave;
    /* outros componentes */
} TDados;

typedef struct SNo {
    TDados Dados;
    TLista Filhos;
} TNo;
```

Estrutura da Árvore por Listas

```
/* procedimentos e funcoes do TAD */  
void TArvore_Inicia(TArvore Arvore, TDados x);  
int TArvore_EhExterno(TArvore Arvore);  
int TArvore_NDescendentes(TArvore Arvore);  
int TArvore_Insere(TArvore Arvore, TArvore SubArv);  
TLista *TArvore_ListaDescendentes(TArvore Arvore);
```

Operações na Árvore por Listas

```
/* Inicia as variaveis da arvore */
void TArvore_Inicia(TArvore Arvore, TDados x)
{
    Arvore->Dados = x;
    TLista_Inicia(&Arvore->Filhos);
}

/* Retorna se a arvore nao possui nenhum descendente */
int TArvore_EhExterno(TArvore Arvore)
{
    return (TLista_Tamanho(&Arvore->Filhos) == 0);
}

/* Retorna o numero de descendentes da arvore */
int TArvore_NDescendentes(TArvore Arvore)
{
    return TLista_Tamanho(&Arvore->Filhos);
}

/* Insere uma subarvore como descendente da raiz da arvore */
int TArvore_Insere(TArvore Arvore, TArvore SubArv)
{
    return TLista_Insere(&Arvore->Filhos, TLista_Tamanho(&Arvore->Filhos), SubArv);
}
```

Operações na Árvore por Listas

```
/* Retorna a lista de descendentes da raiz da arvore */
TLista *TArvore_ListaDescendentes(TArvore Arvore)
{
    TLista *pLista, ListaArv;
    TArvore Arv;

    pLista = (TLista *) malloc(sizeof(TLista));
    TLista_Inicia(pLista);

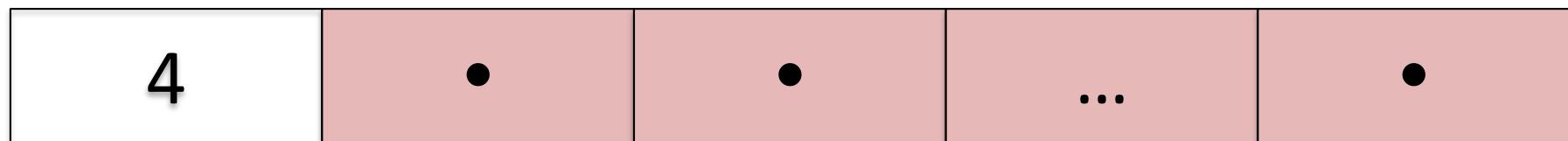
    TLista_Inicia(&ListaArv);
    while (TLista_Retira(&Arvore->Filhos, TLista_Tamanho(&Arvore->Filhos)-1, &Arv))
        TLista_Insere(&ListaArv, TLista_Tamanho(&ListaArv), Arv);

    while (TLista_Retira(&ListaArv, TLista_Tamanho(&ListaArv)-1, &Arv)) {
        TLista_Insere(&Arvore->Filhos, TLista_Tamanho(&Arvore->Filhos), Arv);
        TLista_Insere(pLista, TLista_Tamanho(pLista), Arv);
    }

    return pLista;
}
```

Implementação por Listas

- Entretanto, é mais simples o caso em que cada nó tem um número máximo de filhos d pré-estabelecido

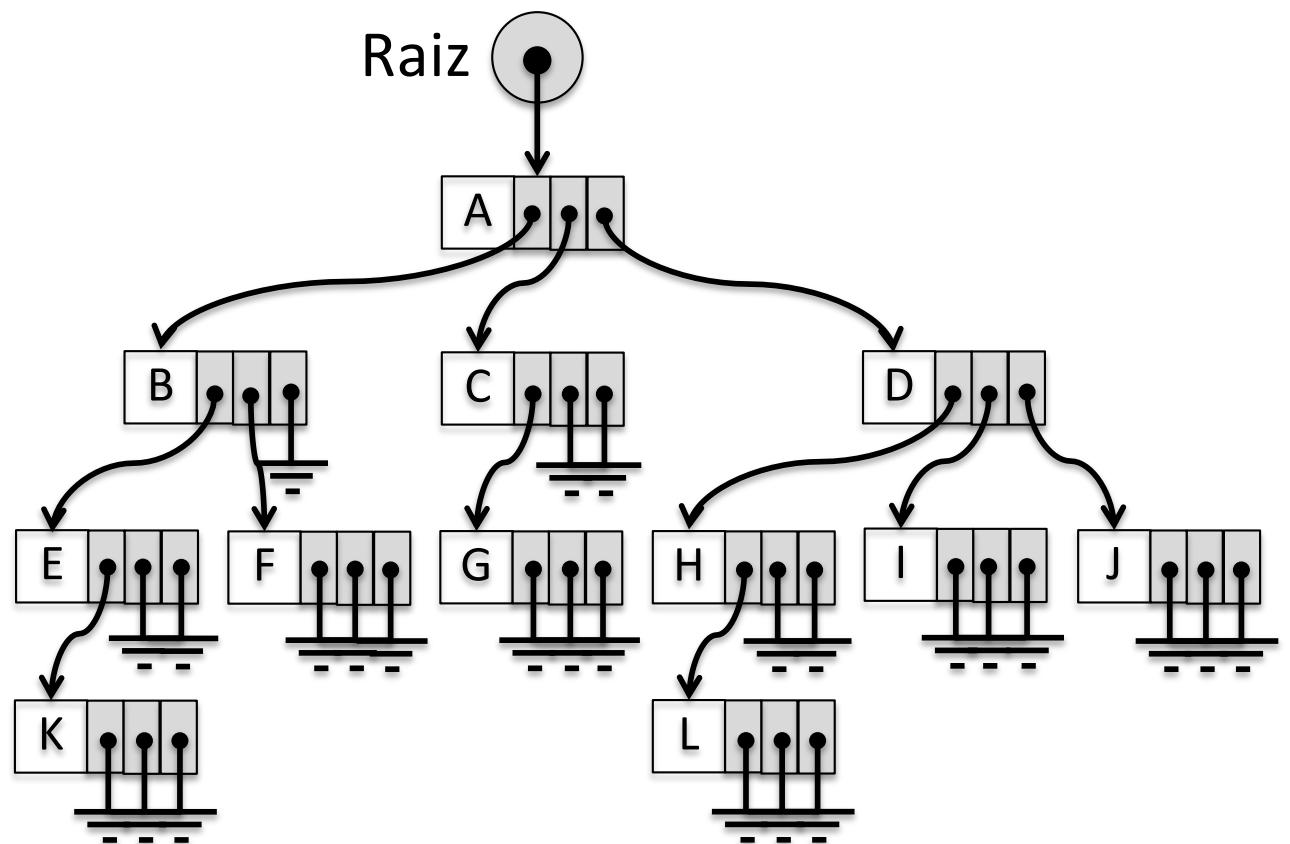
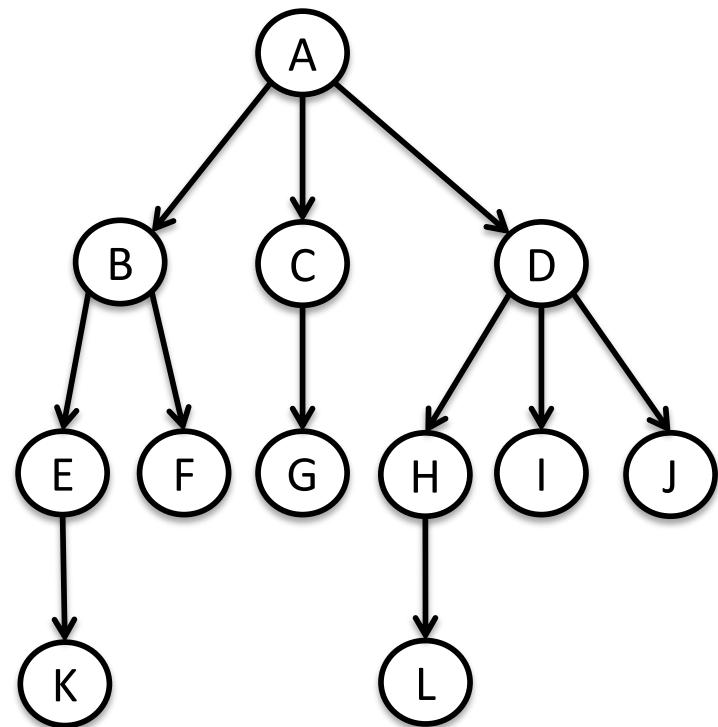


Dados	Campo de ligação para a primeira subárvore	Campo de ligação para a segunda subárvore	Campo de ligação para a d -ésima subárvore
-------	--	---	--

Implementação por Listas

Por exemplo, a árvore ternária seguinte ($d = 3$) ...

... pode ser implementada como mostrado abaixo:



Estrutura da Árvore por Listas

```
#define MAXGRAU 100

typedef struct SNo *TArvore;

typedef TArvore TItem;

typedef int TChave;

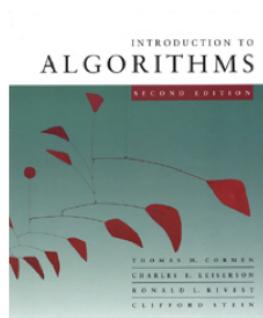
typedef struct {
    TChave Chave;
    /* outros componentes */
} TDados;

typedef struct SNo {
    TDados Dados;
    TLista Filhos;
} TNo;
```

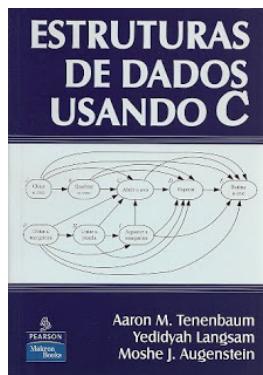
- Imagine que você descobriu uma biblioteca que implementa o TAD Árvore de maneira diferente, mas ainda oferece suporte as seguintes operações:

```
void TArvore_Inicia(TArvore Arvore, TDados x);
int TArvore_EhExterno(TArvore Arvore);
int TArvore_NDescendentes(TArvore Arvore);
int TArvore_Insere(TArvore Arvore, TArvore SubArv);
TLista *TArvore_ListaDescendentes(TArvore Arvore);
```

- Você possui apenas os códigos em binário dessa biblioteca e, portanto, não tem conhecimento de como esse TAD Árvore foi implementado
- Usando esse TAD Árvore, implemente uma função que compare se duas árvores são iguais



CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms**. 3^a Edição. MIT Press, 2009. **Seção 10.4**



TENENBAUM, A. M.; LANGSAM, Y.; AUGENSTEIN, M. J. **Estruturas de Dados usando C**. Pearson Makron Books, 2008.

Capítulo 5