# ÁRVORES

Prof. Muriel Mazzetto

Estrutura de Dados

# Árvores



- Entre as estruturas de dados mais importantes.
- **Estrutura de dados não linear e hierárquica.**
- Seus elementos encontram-se “acima” ou “abaixo” de outros elementos da árvore.

# Árvores



- Entre as estruturas de dados mais importantes.
- **Estrutura de dados não linear e hierárquica.**
- Seus elementos encontram-se “acima” ou “abaixo” de outros elementos da árvore.
- Os elementos são chamados de “nós” ou “nodos”.
- Cada nó possui um campo de *identificação* e um campo para os *dados*.

# Árvores



- Entre as estruturas de dados mais importantes.
- **Estrutura de dados não linear e hierárquica.**
- Seus elementos encontram-se “acima” ou “abaixo” de outros elementos da árvore.
- Os elementos são chamados de “nós” ou “nodos”.
- Cada nó possui um campo de *identificação* e um campo para os *dados*.
- A hierarquia é definida em termos de descendência, ou seja, **um nodo pode ter vários filhos**.

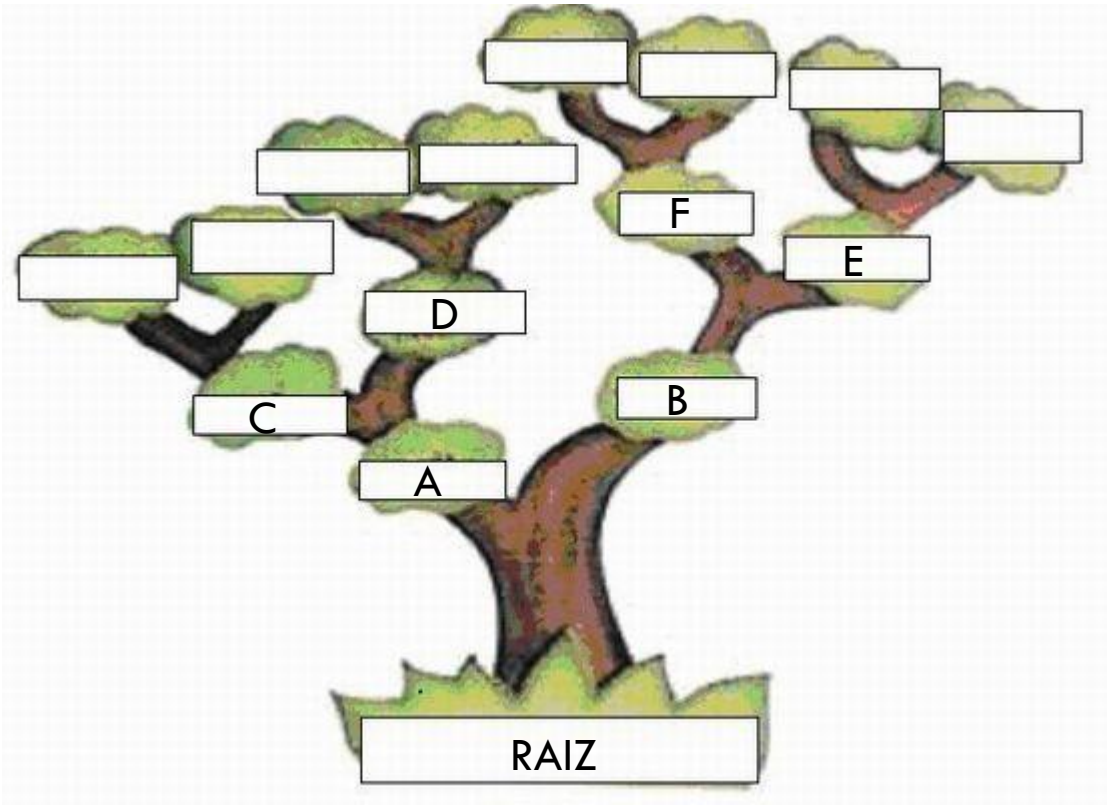
# Árvores



- Definição:
  - Estrutura de dado não linear e hierárquica.
  - Formada por vértices e arestas (nós e conexões).
  - Não possui ciclos (caminhos fechados).
  - Uma árvore sem nós é denominada *vazia*.
  - Uma árvore não vazia consiste em uma raiz e potenciais níveis de nós que formam a hierarquia.

# Árvores

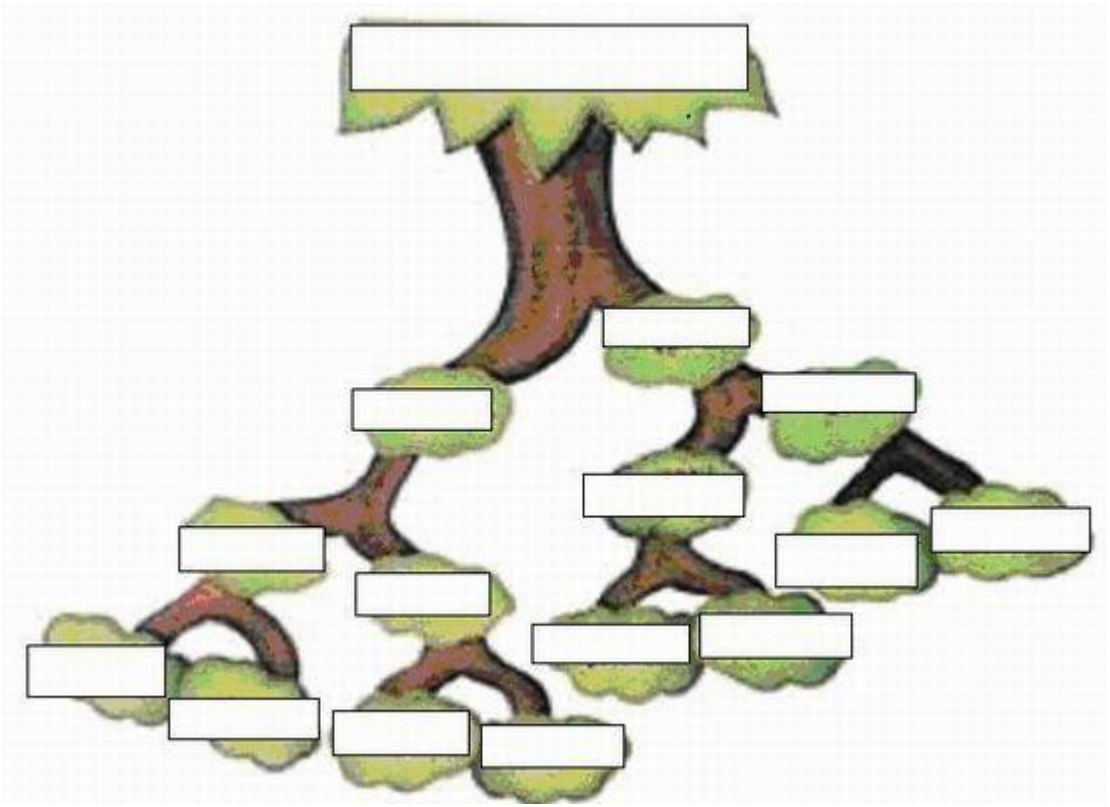


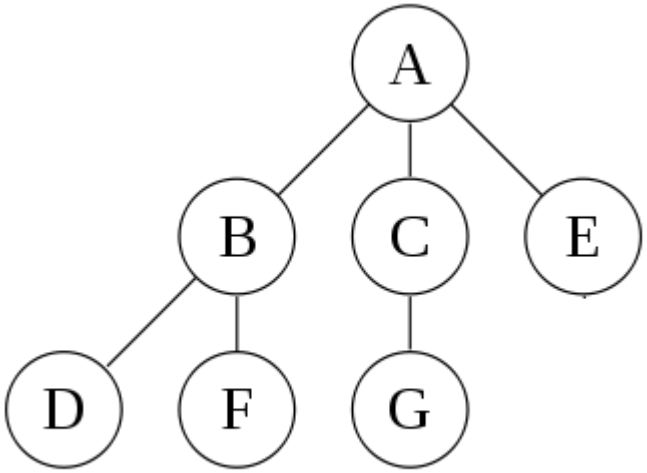



- RAIZ é pai de A e B.
- C é filho de A.
- A é pai de C.
- C é descendente de RAIZ.
- RAIZ é ancestral de todos.

# Árvores

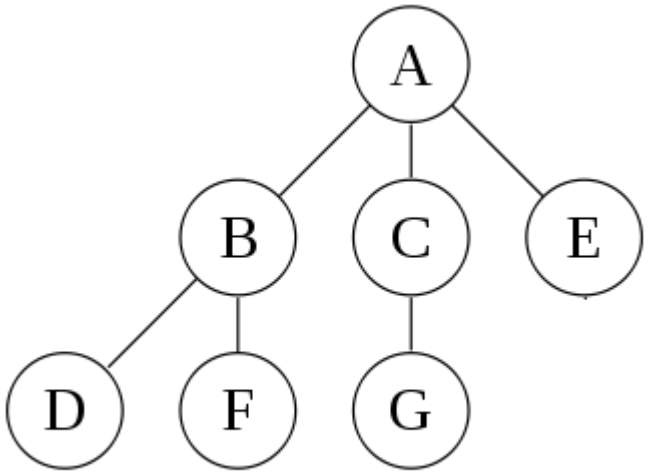






# Árvores






- A é a **raiz**.
- B, C e E são **filhos** de A.
- B é **pai** de D e F.
- G é descendente de A.
- A é ancestral de todos.

# Árvores: Representação



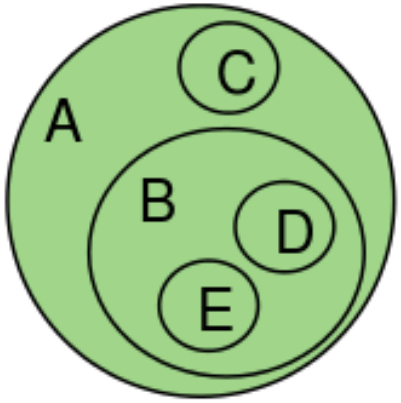



Diagrama de inclusão (Venm)

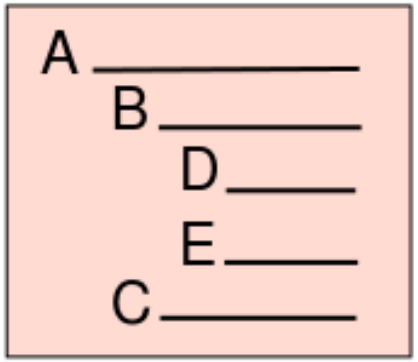
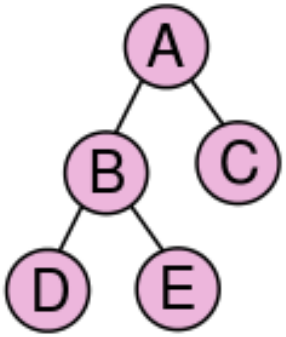



Diagrama de barras




Hierárquica

1A; 1.1B; 1.1.1D; 1.1.2E; 1.2C

Numeração por níveis

(A(B(D)(E))(C))

Aninhamento

# Árvores: definições

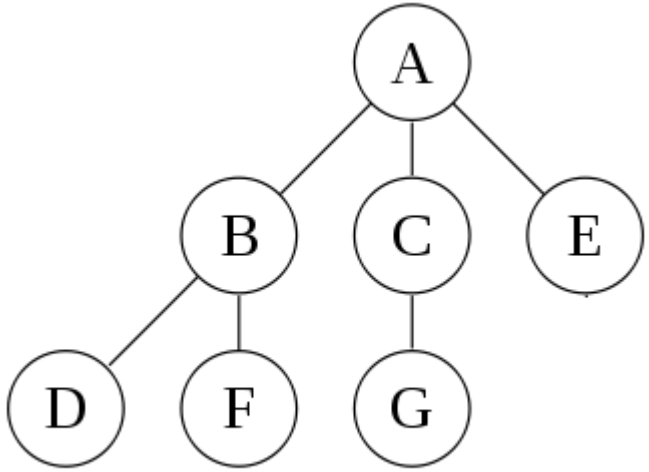


- Raiz: primeiro nó, aquele que não possui pai.
- Nível: raiz possui nível 1, soma-se um cada vez que descer para um filho.
- Profundidade/Altura: maior nível da árvore.
- Grau: quantidade de filhos de um nó (subárvores).
- Folha: nó que não possui filhos.
- Caminho: sequencia de nós de *x* até *y*.
- Subárvore: conjunto do nó e seus descendentes.

# Árvores: definições



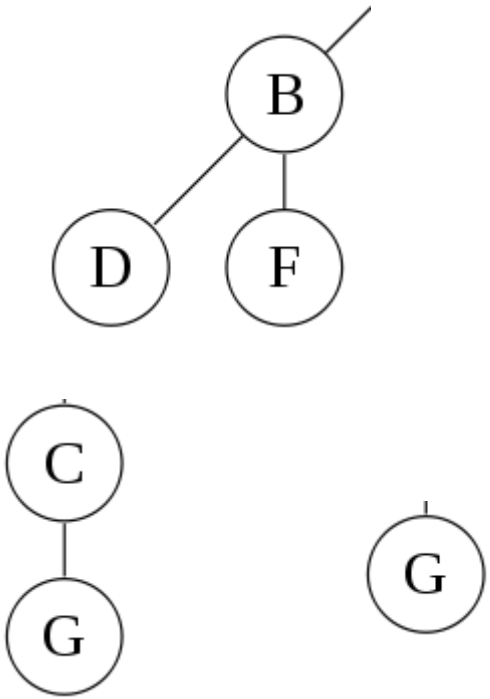
- Raiz: A.
- Nível:
  - B = 2;
  - A = 1;
  - F = 3;
- Profundidade: 3.
- Grau:
  - A = 3;
  - B = 2;
  - C = 1;
- Folha: D, F, G e E.
- Caminho: A – B – D.

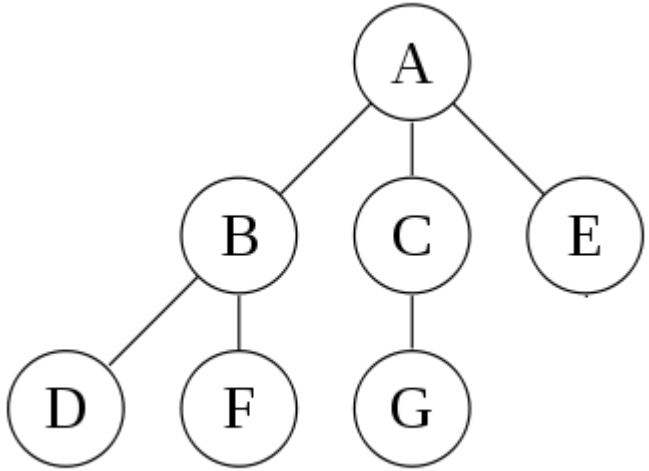


# Árvores: definições



- Subárvore:

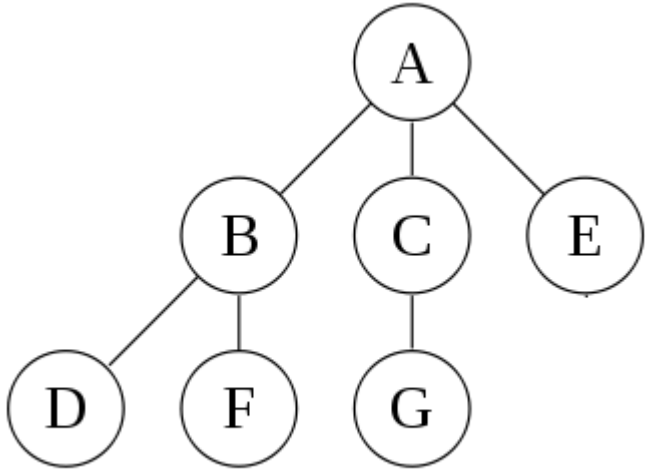


# Árvores: definições



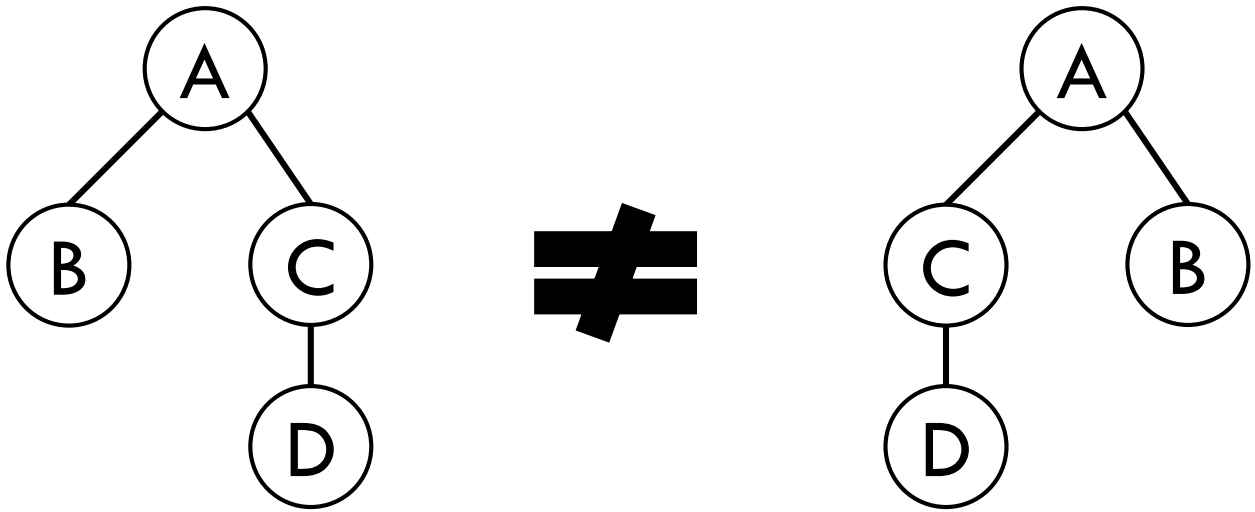
- Pai:
  - B pai de D;
  - C pai de G;
- Irmão:
  - D irmão de F;
  - B irmão de C e E.
- Descendente:
  - F é descendente de B e A.
- Ancestral:
  - A é ancestral de G.



# Árvores: definições



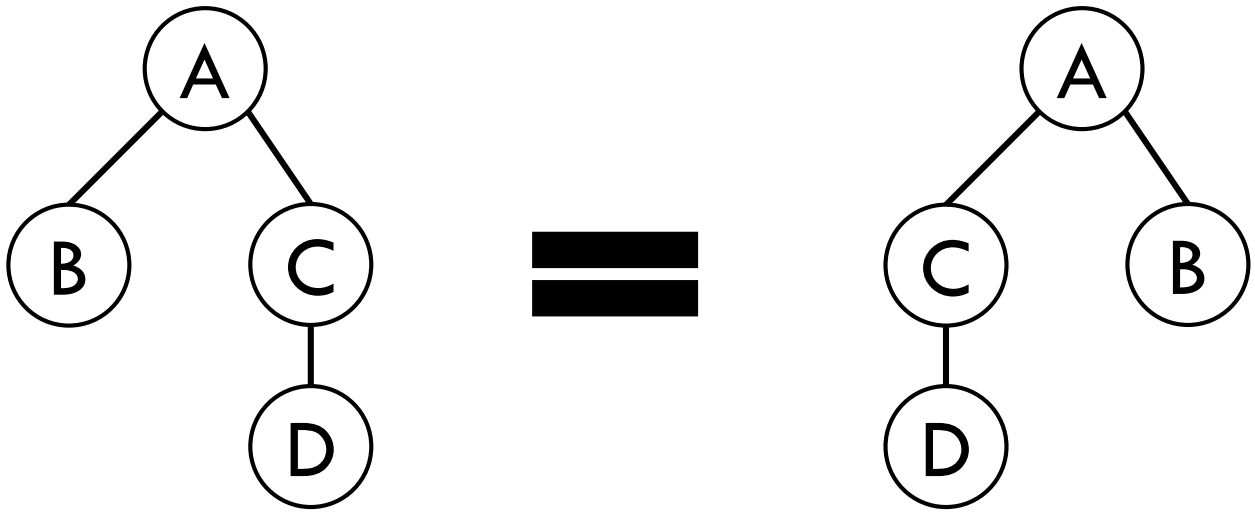
- Uma árvore é **ordenada** se a forma com os filhos estão dispostos é importante.



# Árvores: definições



- Uma árvore é **orientada** se apenas hierarquia dos nós é importante.



# Árvores: implementação



- As árvores podem ser implementadas de diferentes formas:
  - Matriz de adjacência;
  - Listas encadeadas;
  - Nós encadeados;

# Árvores: implementação

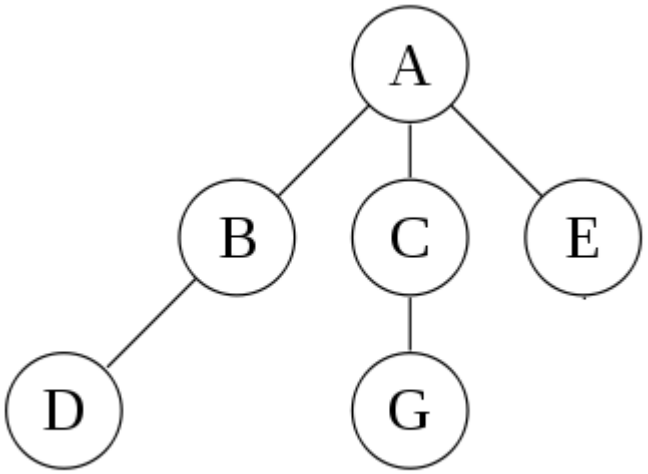


- Matriz de adjacência:
  - Aloca-se uma matriz quadrada com a quantidade de nós existentes.
  - Marca-se os índices dos nós que estão conectados.

# Árvores: implementação



- Matriz de adjacência:
  - Aloca-se uma matriz quadrada com a quantidade de nós existentes.
  - Marca-se os índices dos nós que estão conectados.



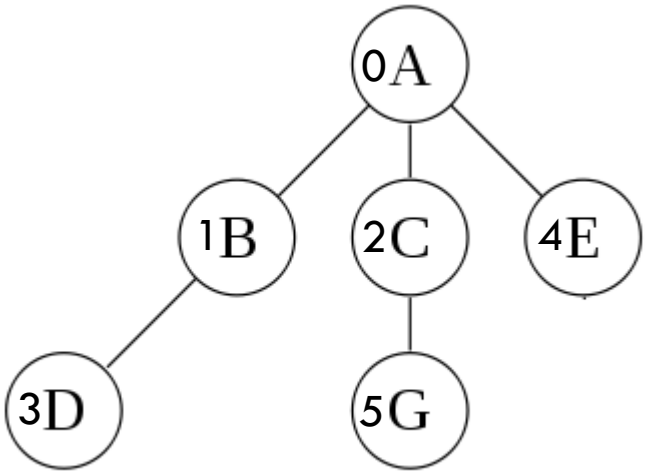

| | A | B | C | D | E | G |
|---|---|---|---|---|---|---|
| A | 0 | 1 | 1 | 0 | 1 | 0 |
| B | 0 | 0 | 0 | 1 | 0 | 0 |
| C | 0 | 0 | 0 | 0 | 0 | 1 |
| D | 0 | 0 | 0 | 0 | 0 | 0 |
| E | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 0 | 0 | 0 | 0 | 0 | 0 |

# Árvores: implementação



- Matriz d
  - Aloca- m a quantidade de nós ex
  - Marca- stão conectados.

Necessário adaptar para que os nós sejam interpretados como os índices da matriz.

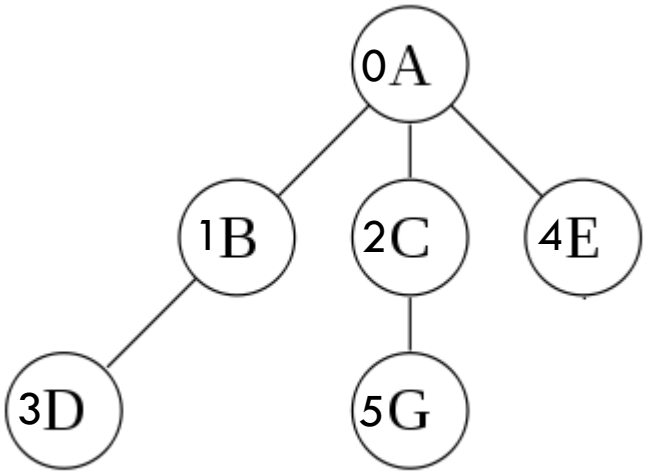



| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 |

# Árvores: implementação



- Matriz de adjacência:
  - Desvantagem: conhecer o máximo de nós; gasto de memória; muitas posições sem uso.

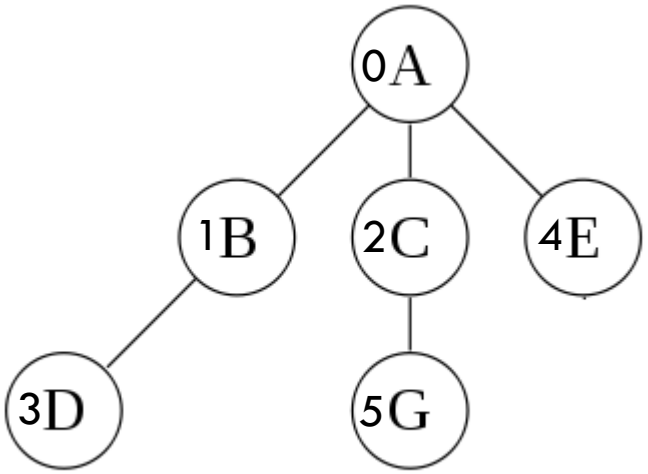



| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 1 | 0 | 1 | 0 |
| **1** | 0 | 0 | 0 | 1 | 0 | 0 |
| **2** | 0 | 0 | 0 | 0 | 0 | 1 |
| **3** | 0 | 0 | 0 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 0 | 0 |

# Árvores: implementação



- Matriz de adjacência:
  - Sugestão: utilizar colunas para armazenar apenas os índices dos filhos.

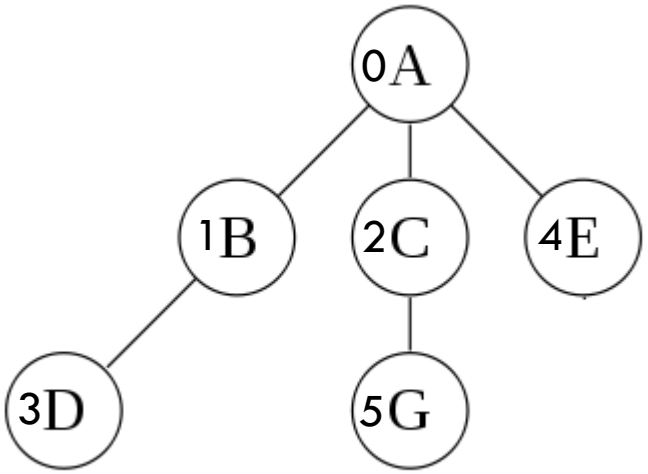



| | 0 | 1 | 2 |
|---|---|---|---|
| A:0 | 1 | 2 | 4 |
| B:1 | 3 | -1 | -1 |
| C:2 | 5 | -1 | -1 |
| D:3 | -1 | -1 | -1 |
| E:4 | -1 | -1 | -1 |
| F:5 | -1 | -1 | -1 |

# Árvores: implementação



- Matriz de adjacência:
  - Sugestão: utilizar colunas para armazenar apenas os índices dos filhos.
  - Desvantagens: necessário conhecer o maior grau.




| | 0 | 1 | 2 |
|---|---|---|---|
| A:0 | 1 | 2 | 4 |
| B:1 | 3 | -1 | -1 |
| C:2 | 5 | -1 | -1 |
| D:3 | -1 | -1 | -1 |
| E:4 | -1 | -1 | -1 |
| F:5 | -1 | -1 | -1 |

# Árvores: implementação

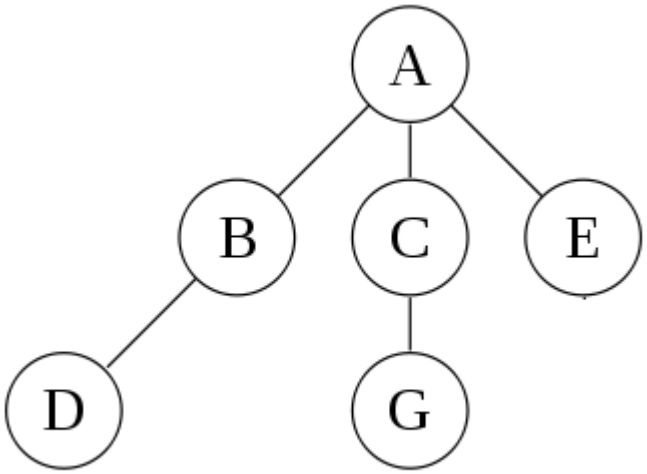


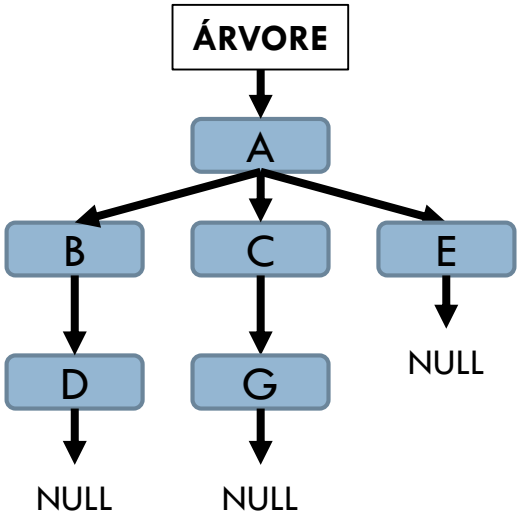
- Nós encadeados:
  - Cada nó aponta diretamente para a posição de memória de seus filhos (e pai).
  - Similar às listas encadeadas.

# Árvores: implementação



- Nós encadeados:
  - Cada nó aponta diretamente para a posição de memória de seus filhos (e pai).
  - Similar às listas encadeadas.

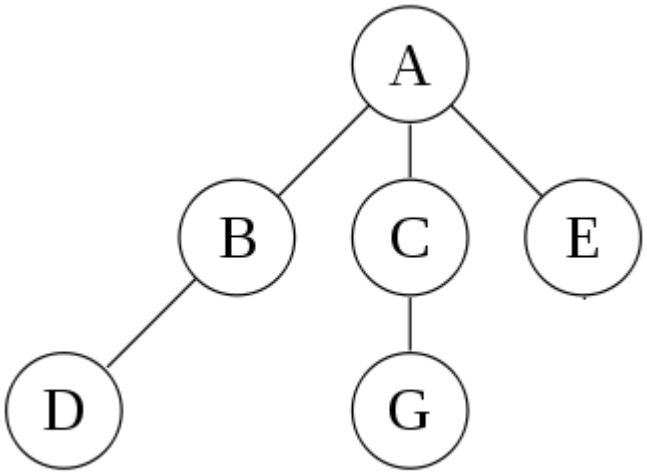


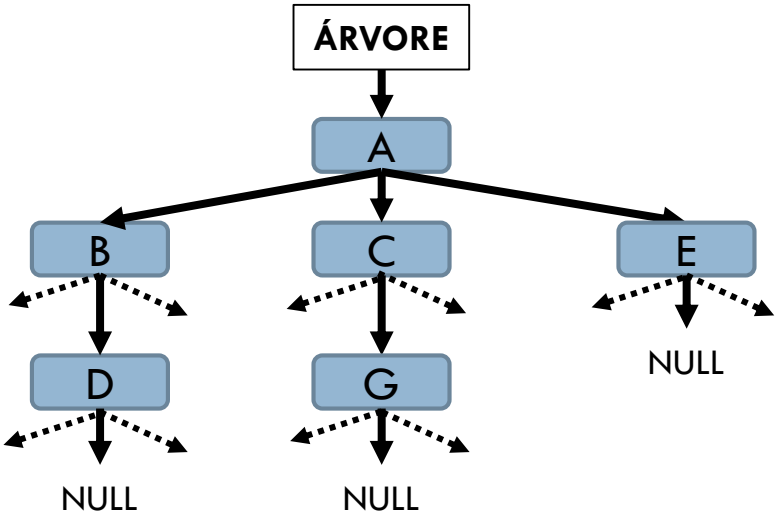



# Árvores: implementação



- Nós encadeados:
  - Desvantagem: necessário conhecer o grau máximo da árvore; implementar a *struct* com a quantidade de ponteiros necessários.



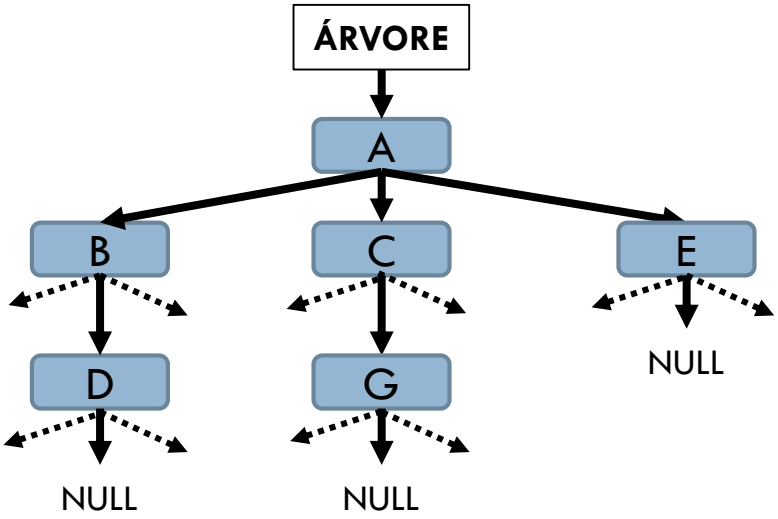



# Árvores: implementação



- Nós encadeados:
  - Desvantagem: necessário conhecer o grau máximo da árvore; implementar a *struct* com a quantidade de ponteiros necessários.

```
struct NO{
    struct dados *info;
    struct NO *p1;
    struct NO *p2;
    struct NO *p3;
};
```



# Árvores: implementação

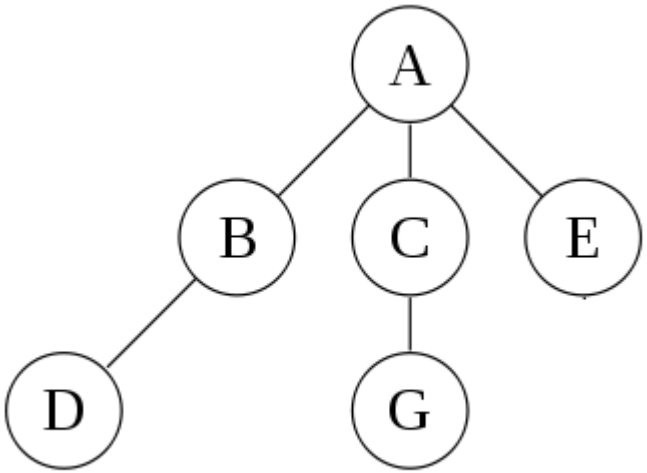


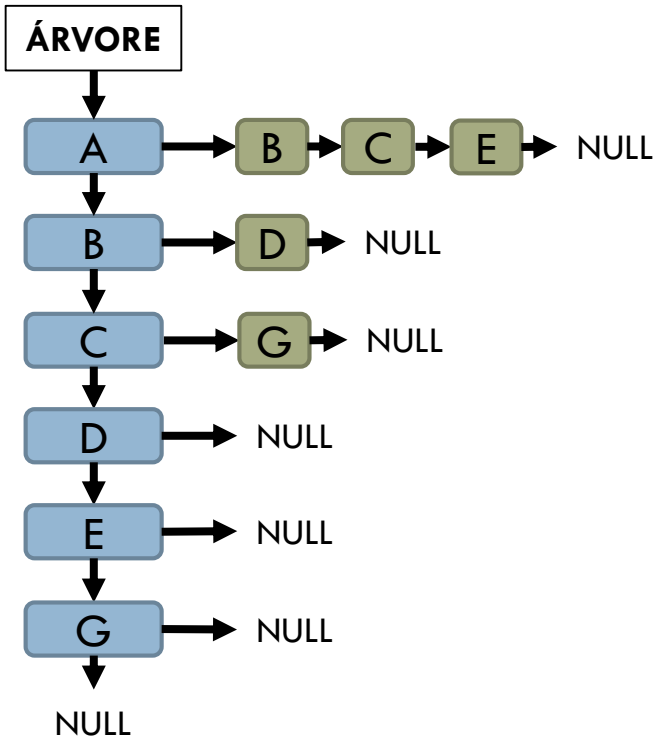
- Lista encadeada:
  - Lista de nós (vetor ou lista encadeada).
  - Cada nó possui sua própria **lista encadeada de ponteiros** para os filhos.

# Árvores: implementação



- Lista encadeada:
  - Lista de nós (vetor ou lista encadeada).
  - Cada nó possui sua própria **lista encadeada de ponteiros** para os filhos.

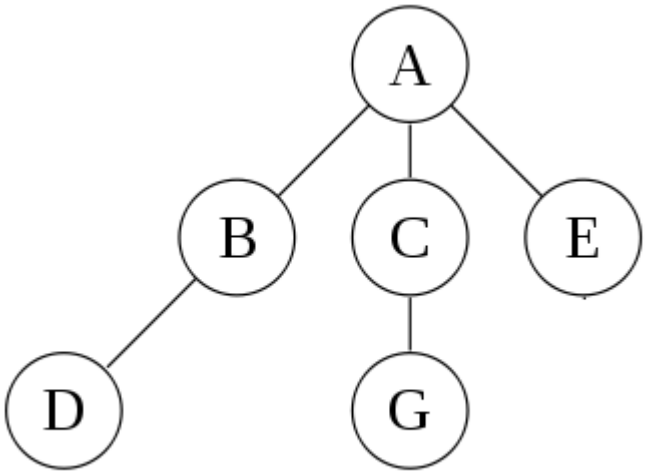


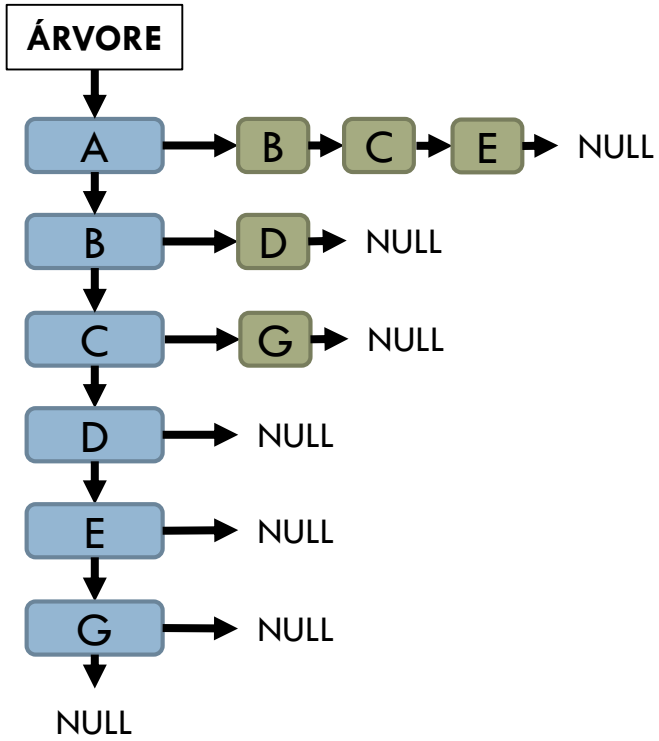



# Árvores: implementação



- Lista encadeada:
  - Forma mais generalizada de implementação.
  - Vantagem: maior gerenciamento e economia de memória.



# Árvores



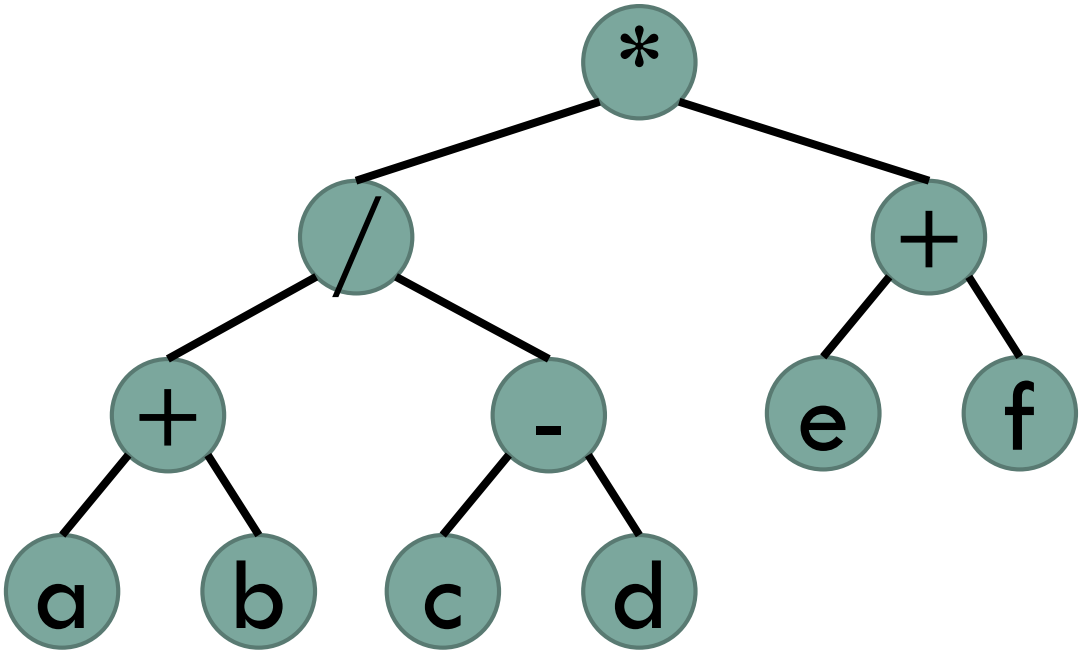
- Usados para organização:
  - Pastas de SO;
  - Interfaces gráficas (menus);
  - Bancos de dados.
- Usados para buscas:
  - Ordenação e pesquisa de dados alocados dinamicamente.
- Aplicações em Inteligência Artificial:
  - Árvore de Pesquisa.
- Compiladores (análise sintática).

# Árvores



- Representação de uma expressão aritmética:

$$\left(\frac{a+b}{c-d}\right) * (e+f)$$



# Árvores



- Variações:
  - Árvores n-árias;
  - Árvores binárias de busca;
  - Árvores balanceadas:
    - AVL;
    - RedBlack;
    - Árvore B e B+;