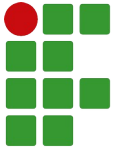


INSTITUTO FEDERAL
Sertão Pernambucano
Campus Salgueiro

Estrutura de Dados e Algoritmos com Java

Prof. Heraldo Gonçalves Lima Junior
heraldo.junior@ifsertao-pe.edu.br

1. Árvores

1.1. Introdução

- **Estrutura de dados muito utilizada**
  - Permite a representação de dados de maneira hierárquica;
  - Fornece maneiras eficientes de busca;

1.1. Introdução

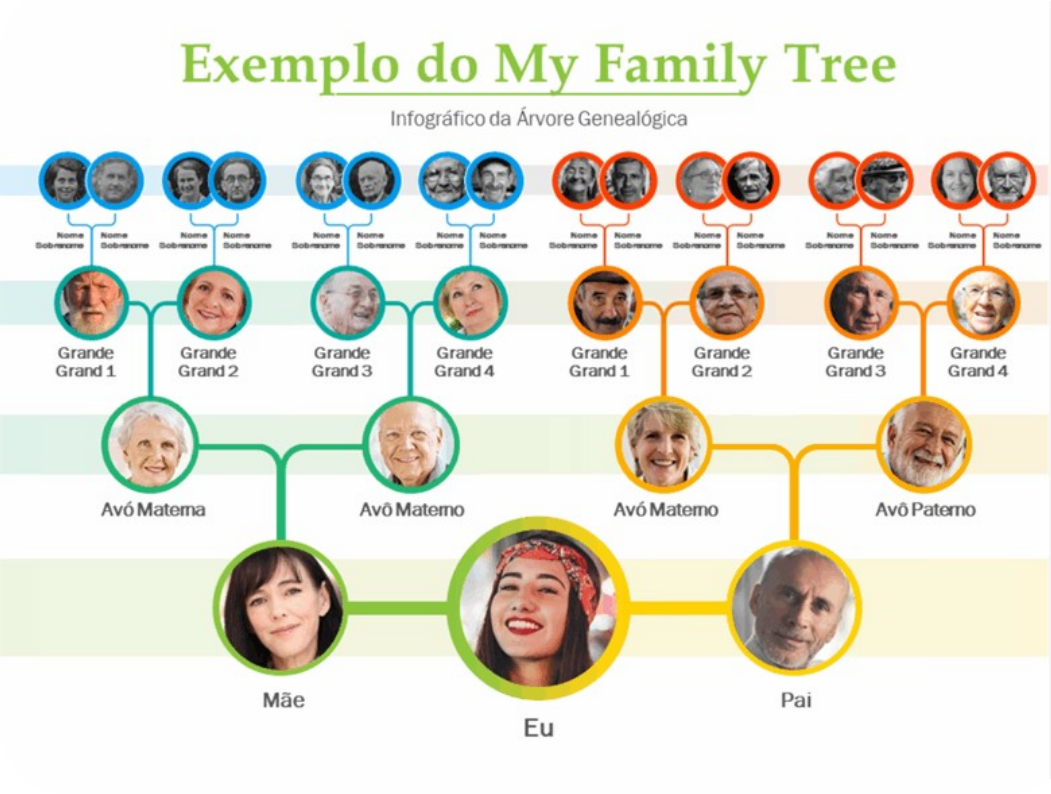
- Árvores são amplamente utilizadas e possuem diversas estratégias de implementação;

- Cada estratégia tem um conjunto específico de algoritmos para armazenamento e recuperação dos dados;

1.1. Introdução

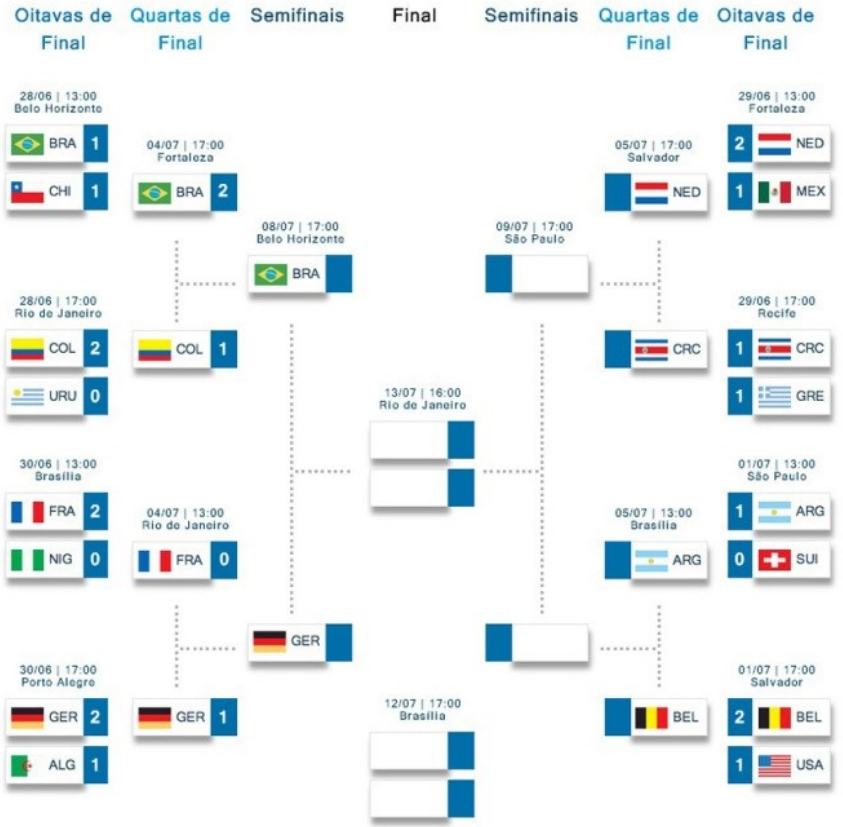
- Árvores não são lineares, ou seja, os elementos não são dispostos em forma sequencial;
- **Quais são seus usos comuns?**
  - Manipular dados hierárquicos
  - Manipular listar ordenadas de dados
  - Em algoritmos de roteamento de pacotes

1.2. Exemplos do nosso cotidiano



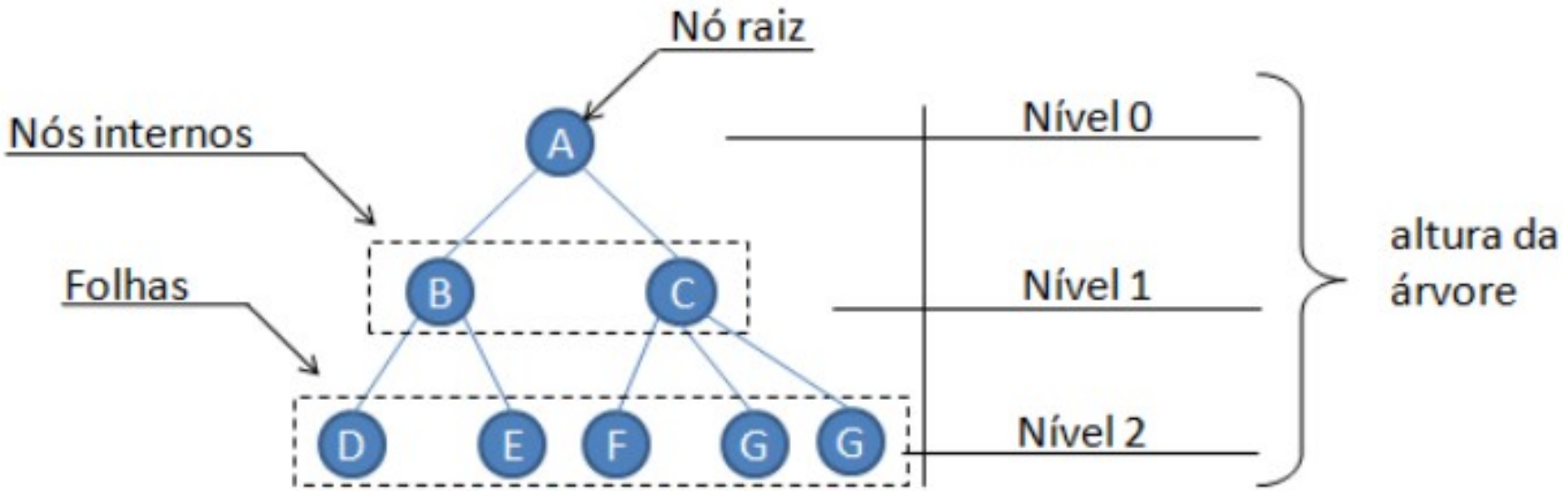
1.2. Exemplos do nosso cotidiano

1.2. Exemplos do nosso cotidiano



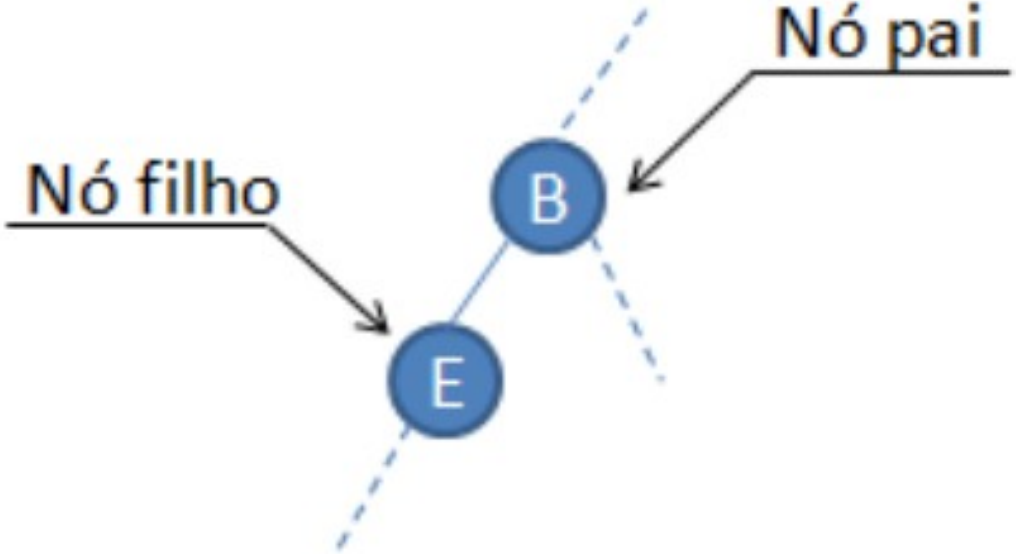
1.2. Exemplos do nosso cotidiano

1.3. Composição

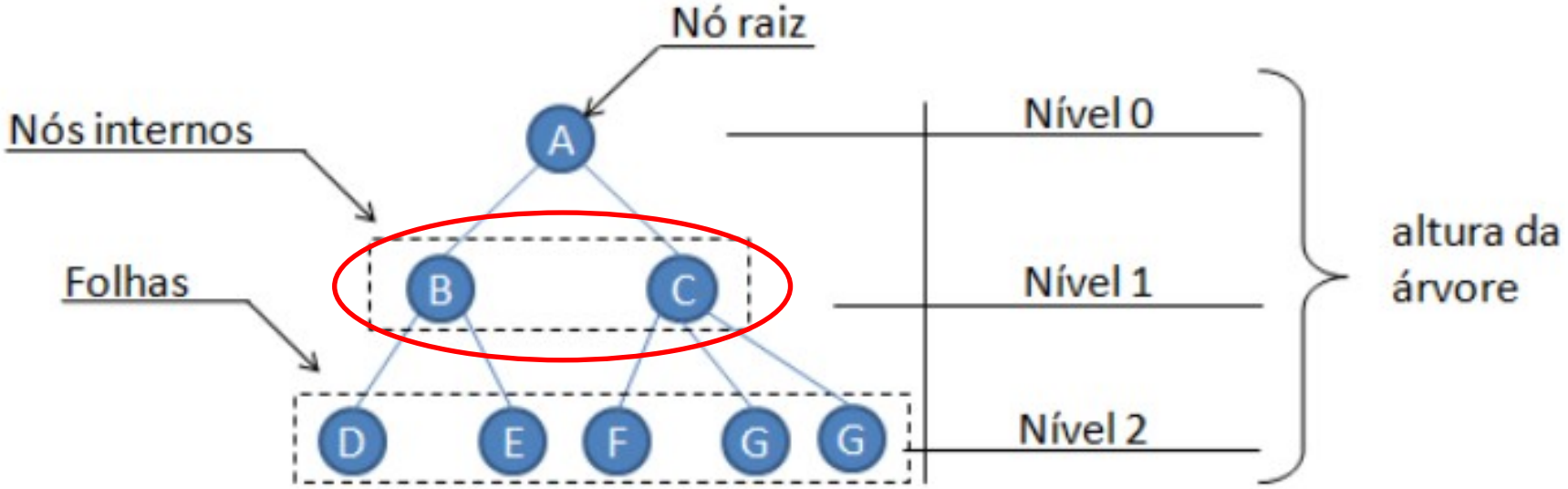


Nodos/Nós: são os elementos inseridos em uma árvore.

1.3. Composição

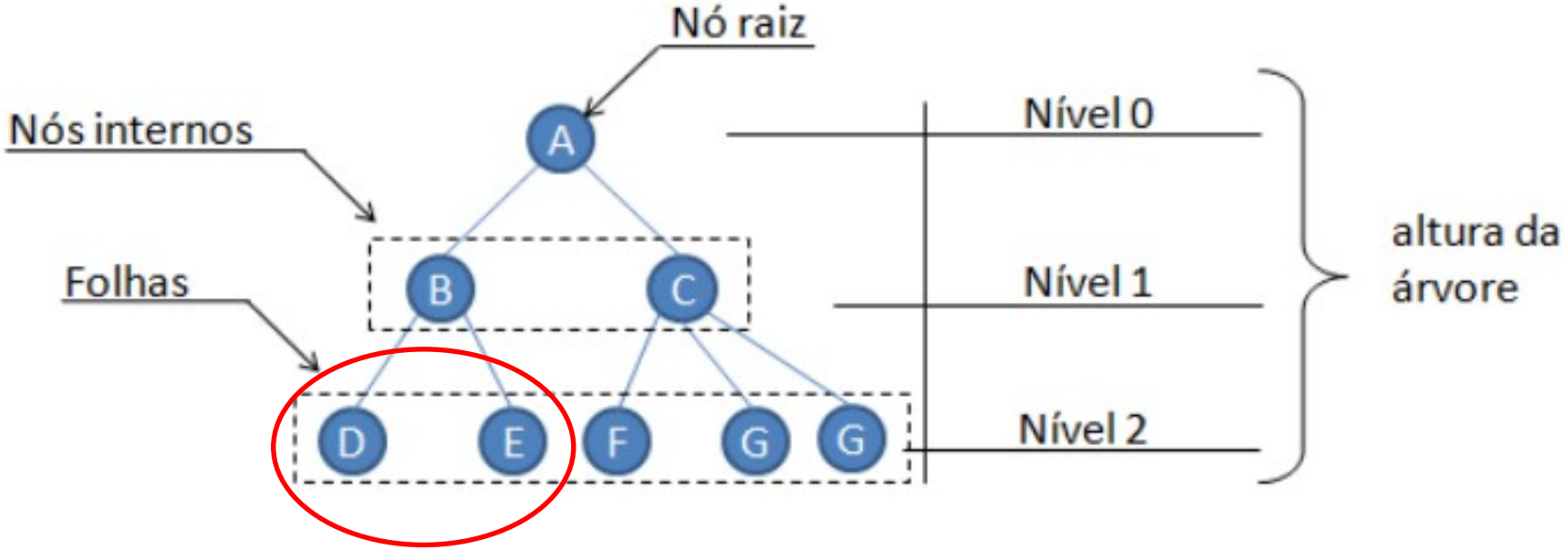


1.3. Composição



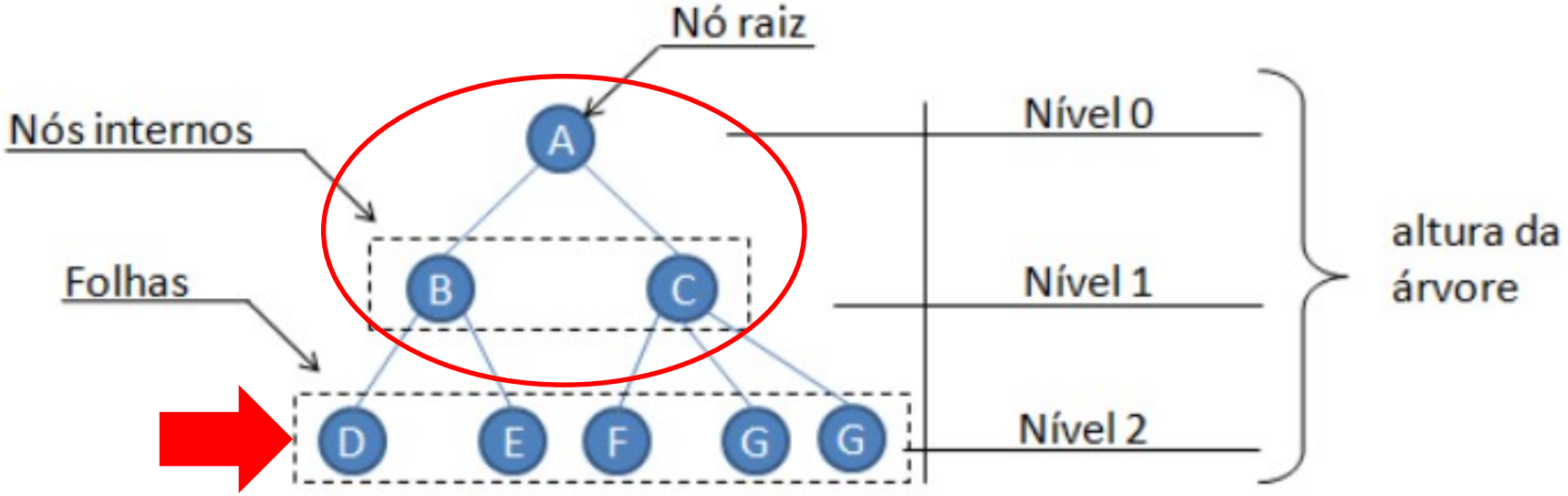
Nodos Galhos: Nós intermediários entre a raiz e as folhas.

1.3. Composição



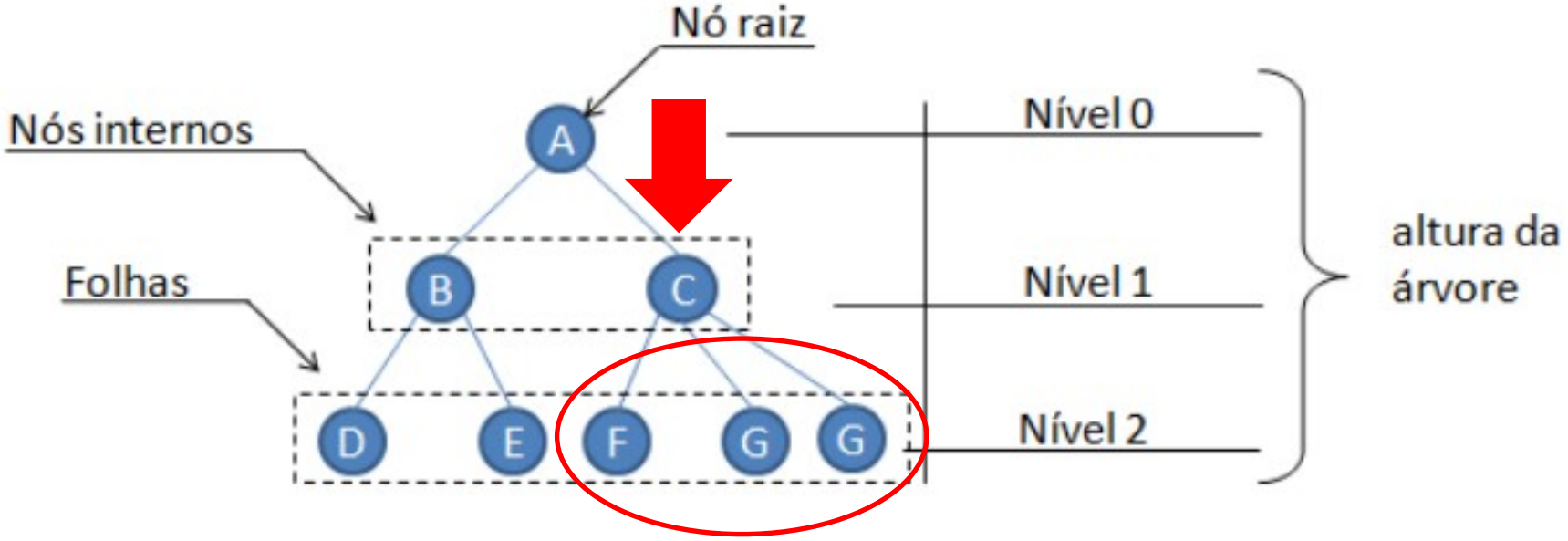
Nodos Irmãos: compartilham o mesmo nodo pai.

1.3. Composição



Nodos Ancestrais: são todos os nodos que estão acima de um nodo, com ligação direta ou indireta.

1.3. Composição



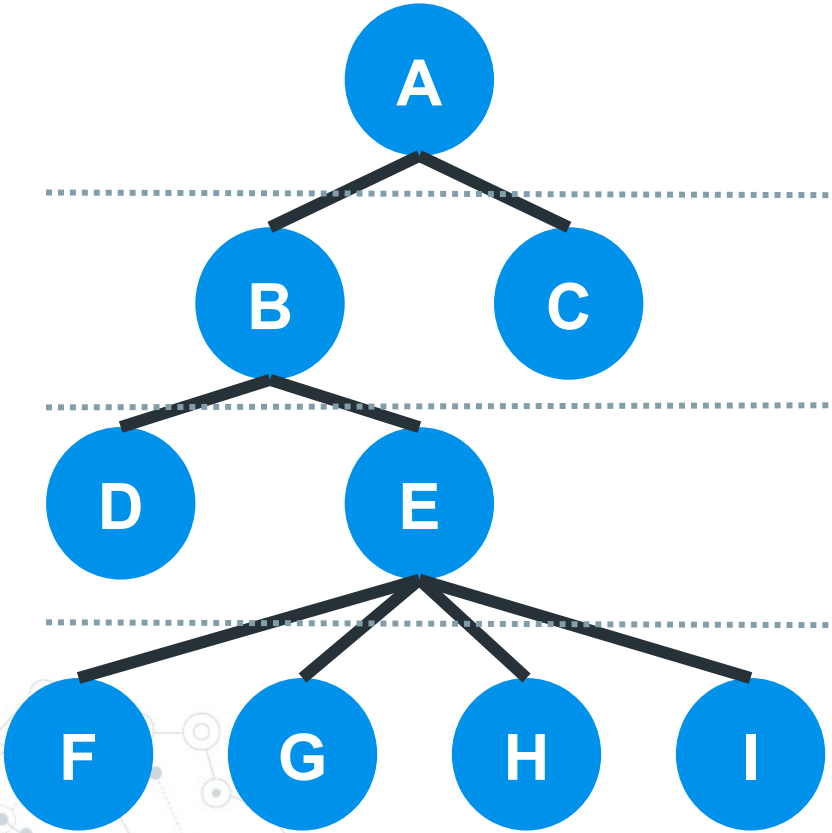
Nodos Descendentes: são todos os nodos que estão abaixo de um nodo, com ligação direta ou indireta.

1.4. Propriedades

- As principais propriedades de cada nodo são:
  - **Ordem/Grau:** Indica o número de filhos que o nodo possui;
  - **Nível/Altura/Profundidade:** Indica a distância do nodo em relação à raiz;
- A árvore também possuí valores de ordem e de nível.
  - Ambos os valores serão assumidos com base no maior valor encontrado em qualquer nodo.

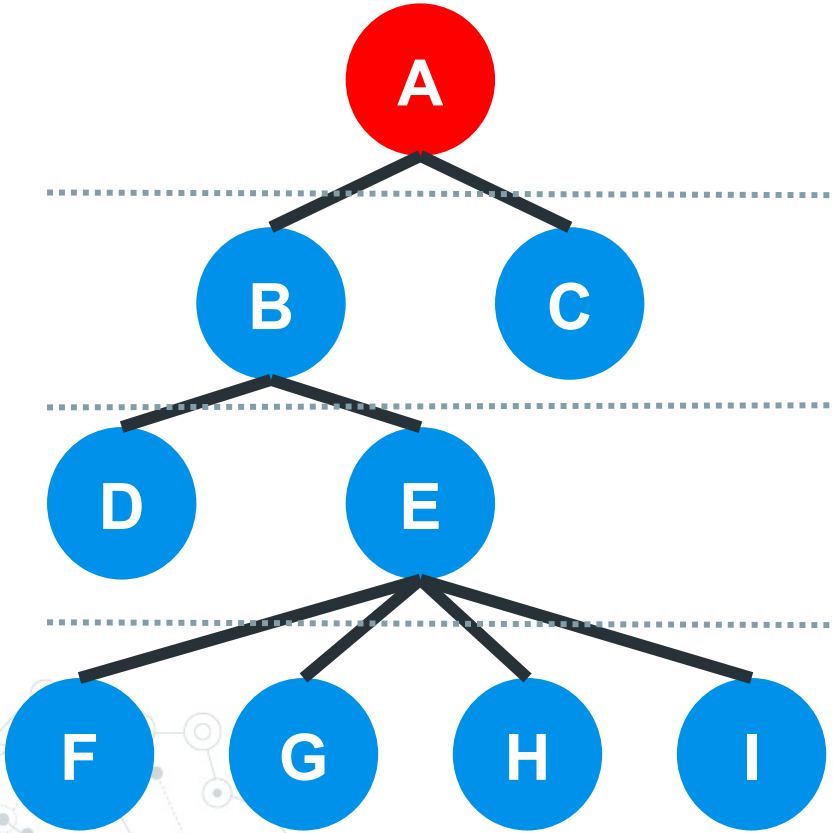
1.4. Propriedades

Árvore: Nível 3 e Ordem 4



1.4. Propriedades

Árvore: Nível 3 e Ordem 4



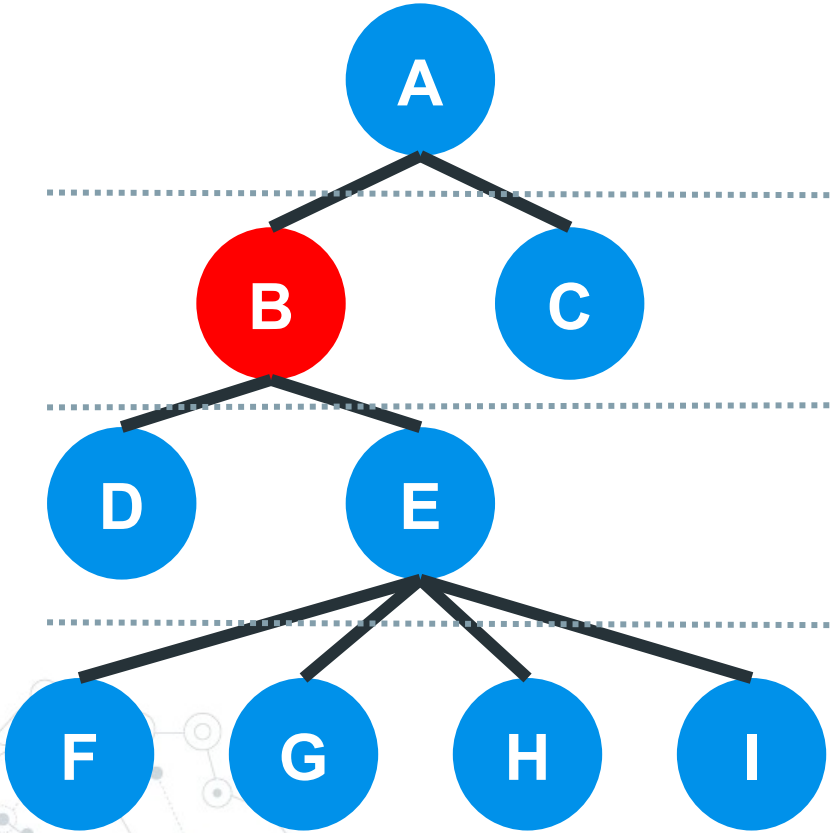
NÓ A:

Tipo: Raiz

Ordem: 2

1.4. Propriedades

Árvore: Nível 3 e Ordem 4



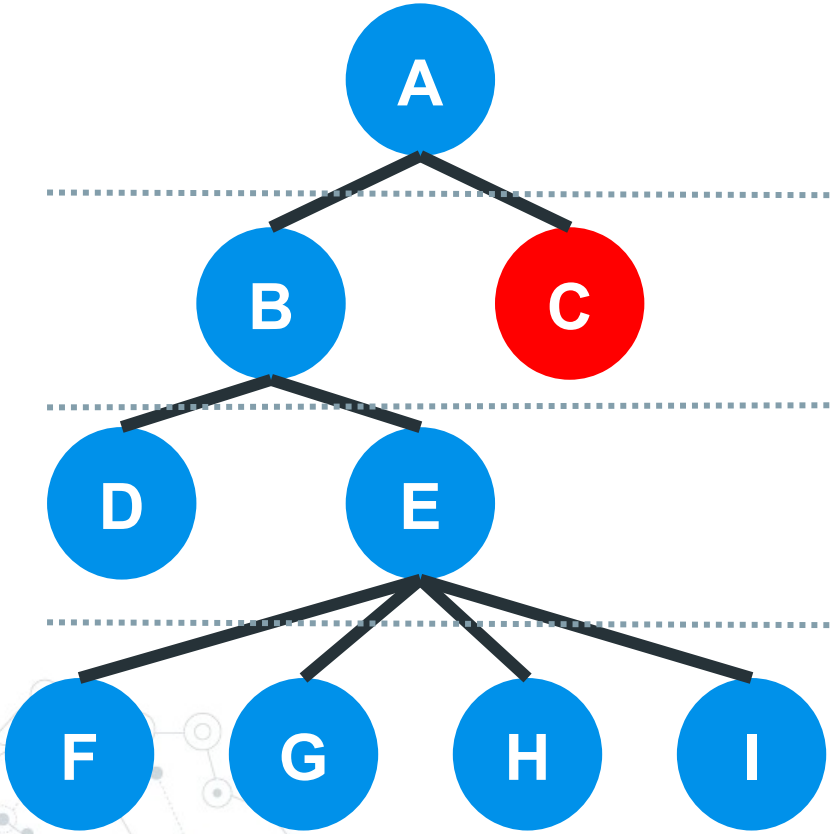
NÓ B:

Tipo: Galho

Ordem: 2

1.4. Propriedades

Árvore: Nível 3 e Ordem 4



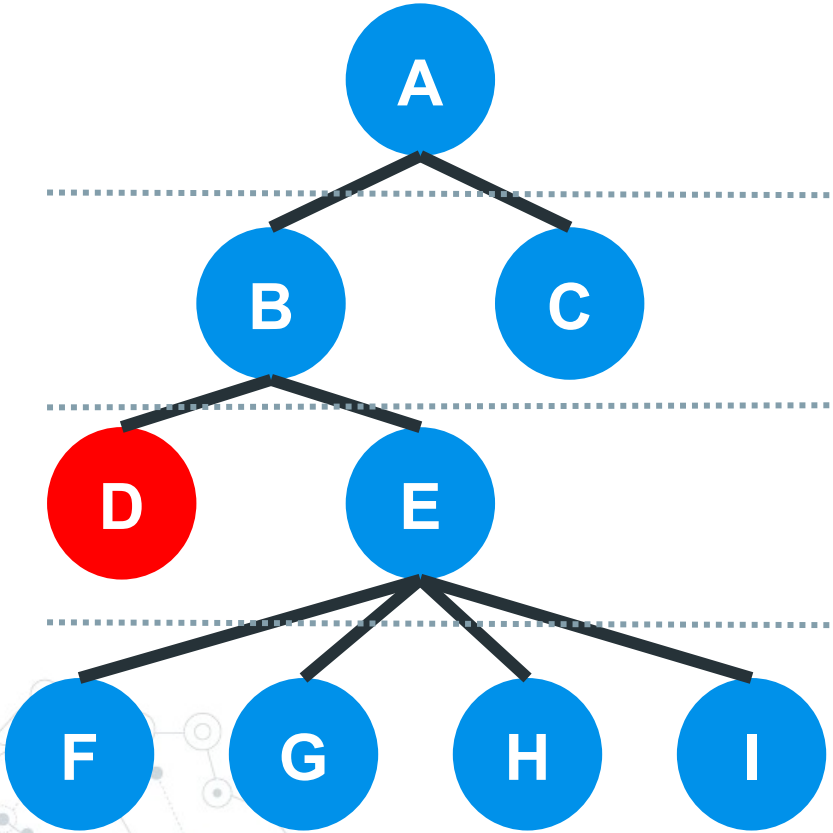
NÓ C:

Tipo:

Ordem:

1.4. Propriedades

Árvore: Nível 3 e Ordem 4



NÍVEL 0

NÍVEL 1

NÍVEL 2

NÍVEL 3

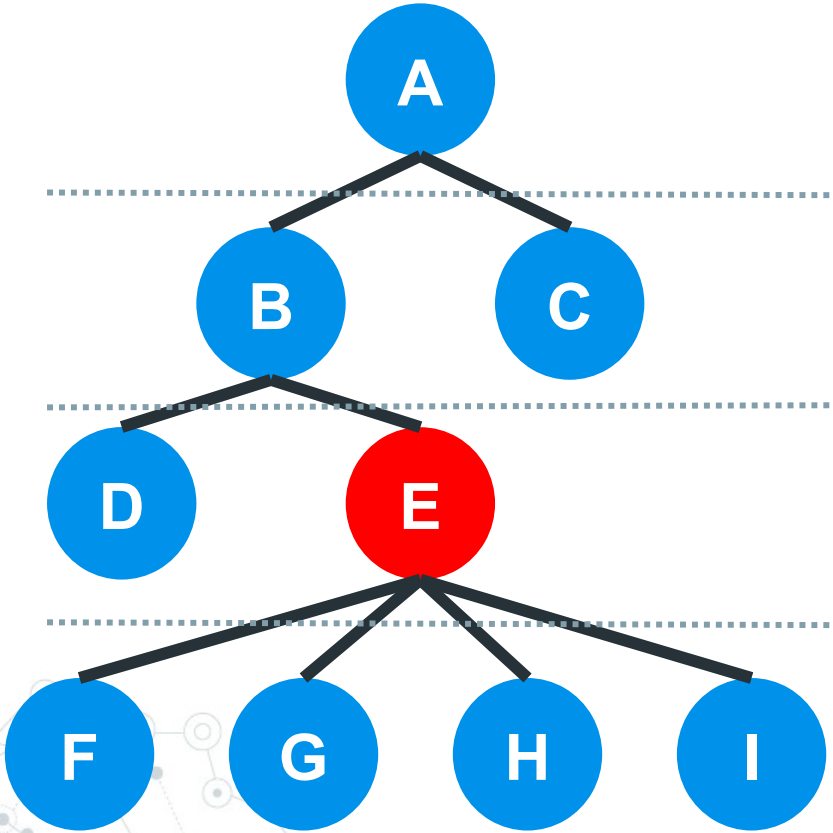
NÓ D:

Tipo: Folha

Ordem: 0

1.4. Propriedades

Árvore: Nível 3 e Ordem 4



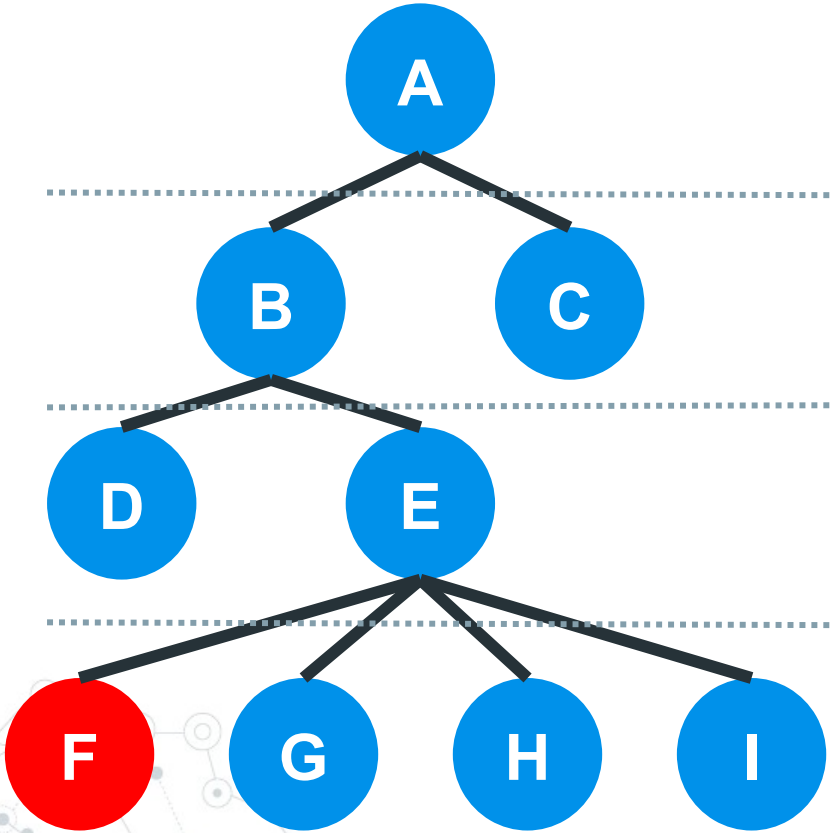
NÓ E:

Tipo:

Ordem:

1.4. Propriedades

Árvore: Nível 3 e Ordem 4



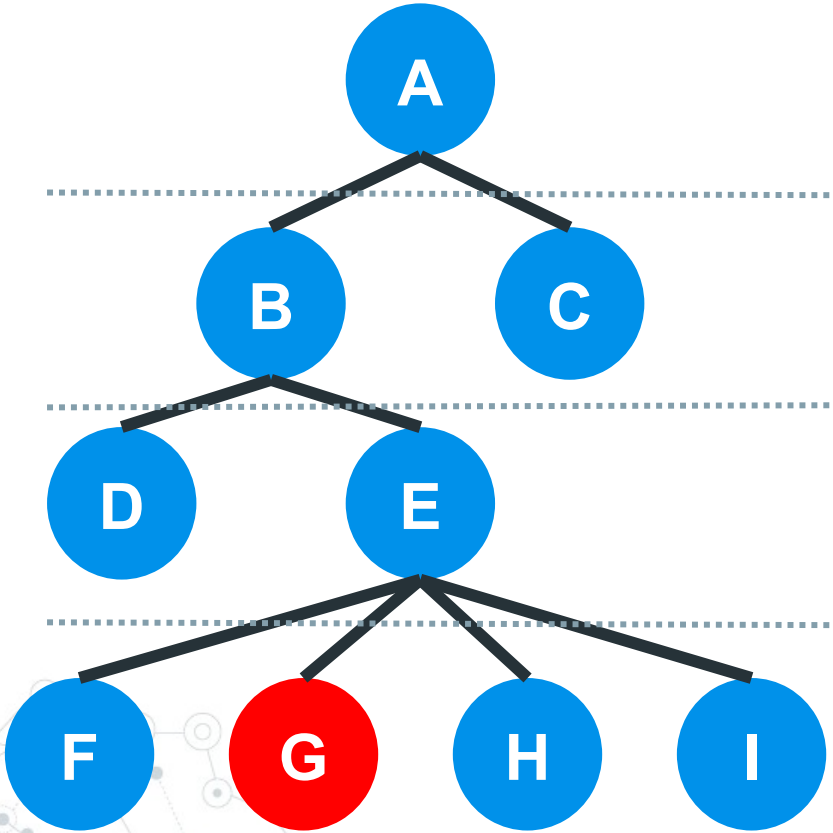
NÓ F:

Tipo: Folha

Ordem: 0

1.4. Propriedades

Árvore: Nível 3 e Ordem 4



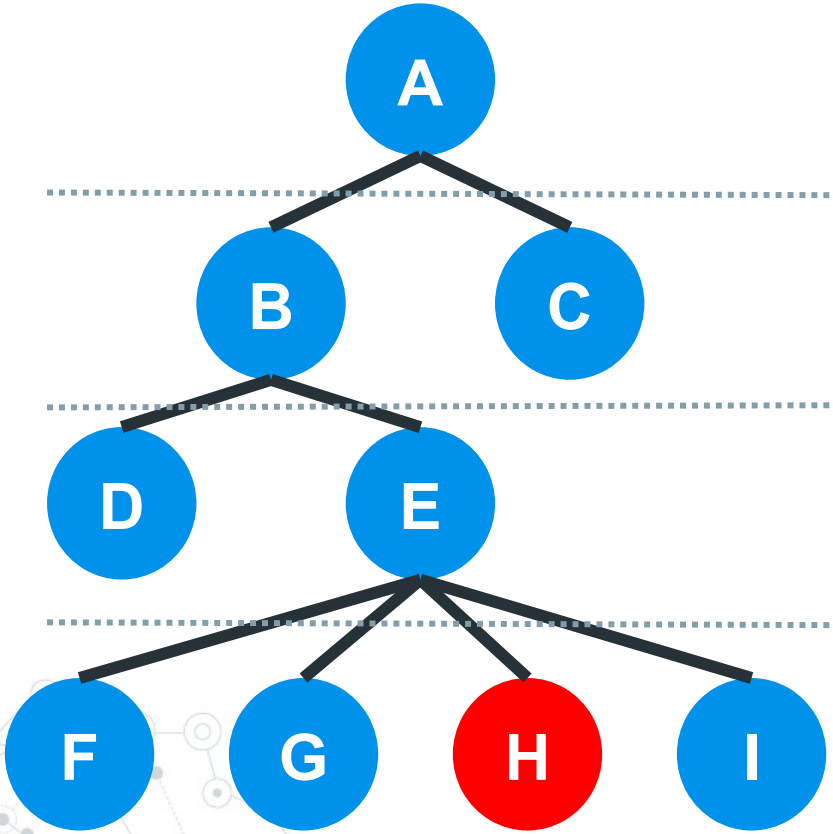
NÓ G:

Tipo: Folha

Ordem: 0

1.4. Propriedades

Árvore: Nível 3 e Ordem 4



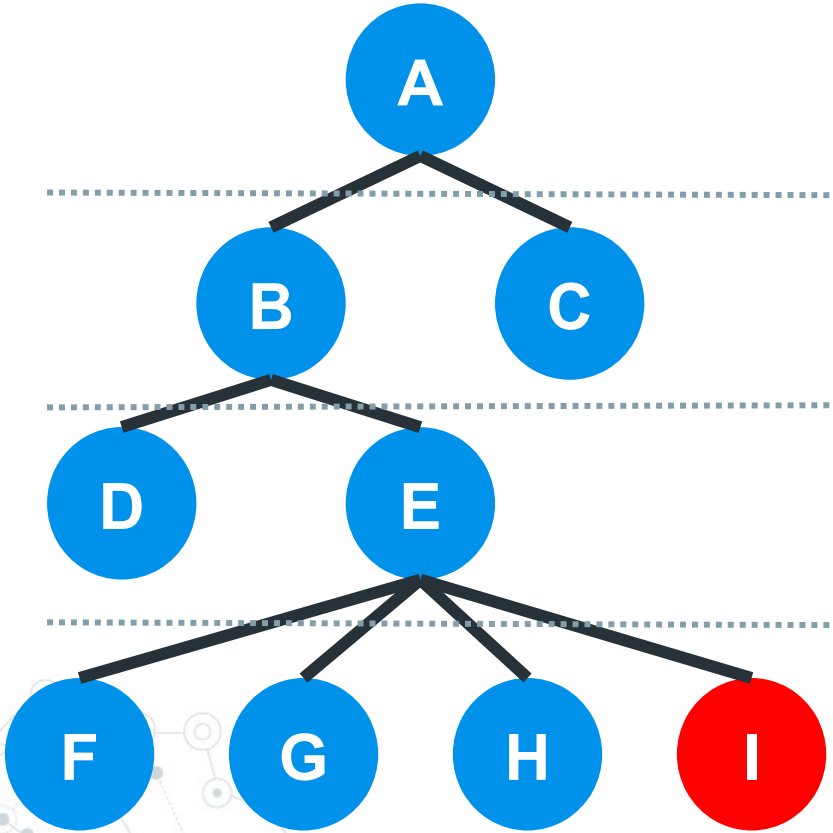
NÓ H:

Tipo: Folha

Ordem: 0

1.4. Propriedades

Árvore: Nível 3 e Ordem 4



NÓ I:

Tipo: Folha

Ordem: 0

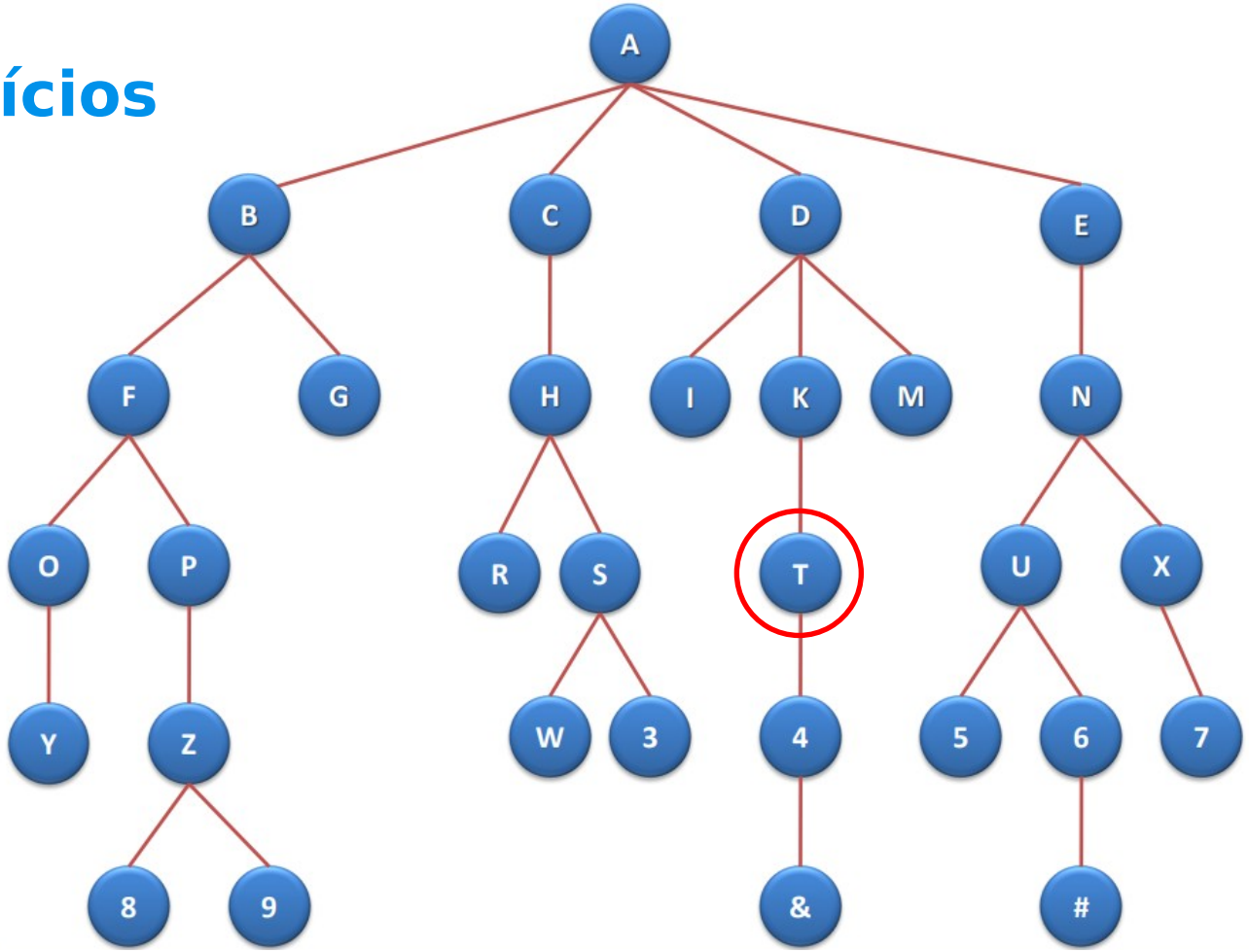
1.5. Exercícios

NÓ T:

Tipo:

Nível:

Ordem:



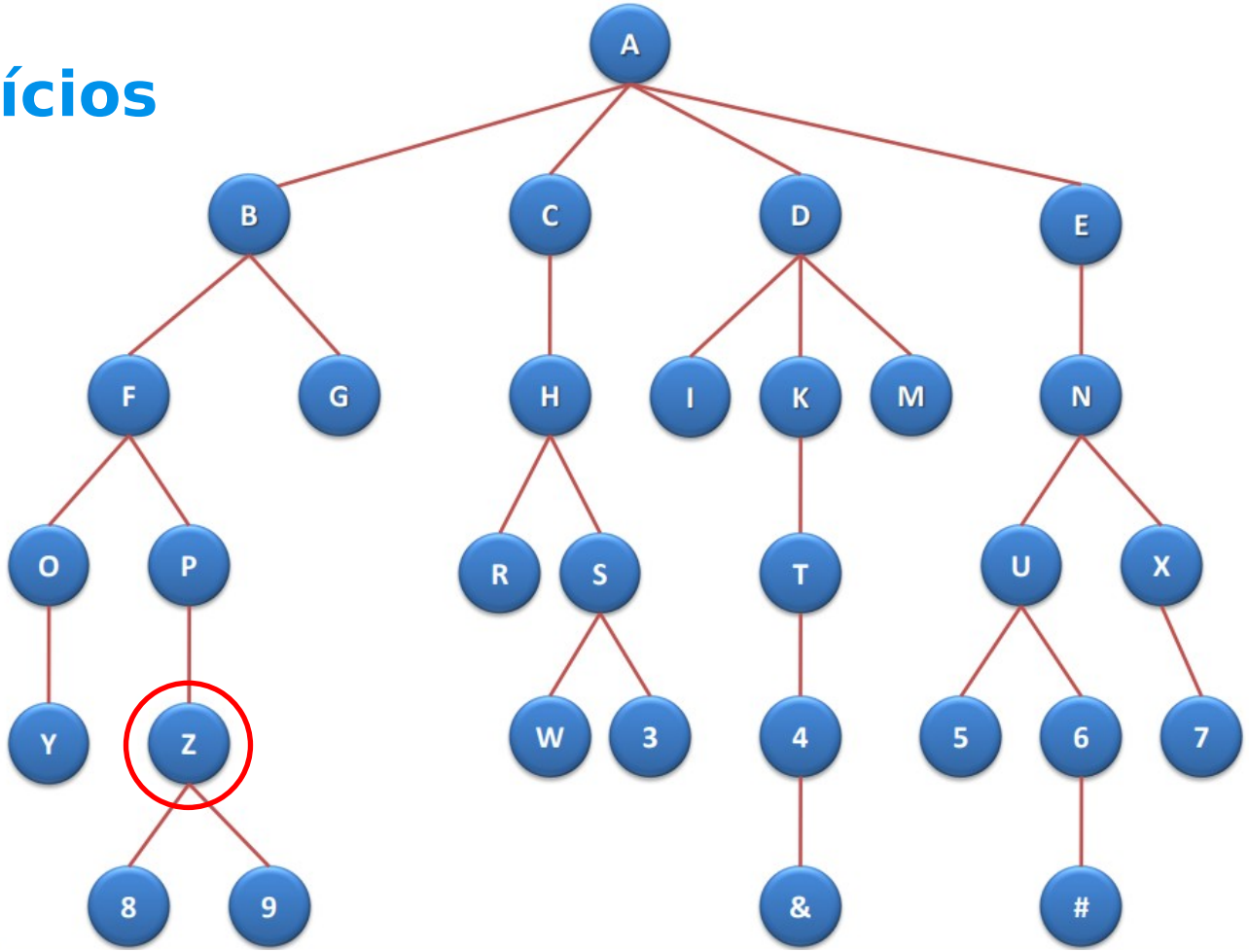
1.5. Exercícios

NÓ Z:

Tipo:

Nível:

Ordem:



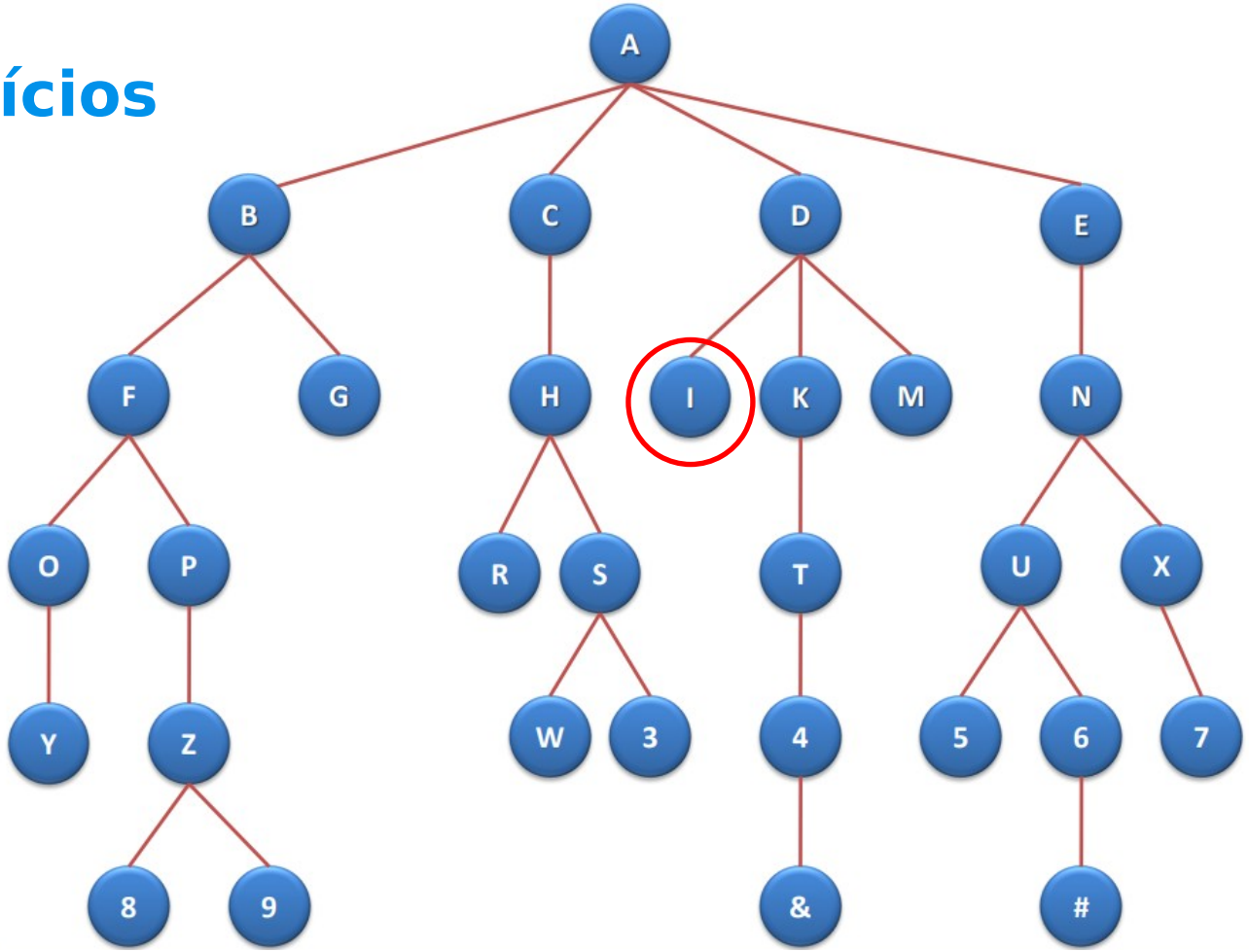
1.5. Exercícios

NÓ I:

Tipo:

Nível:

Ordem:



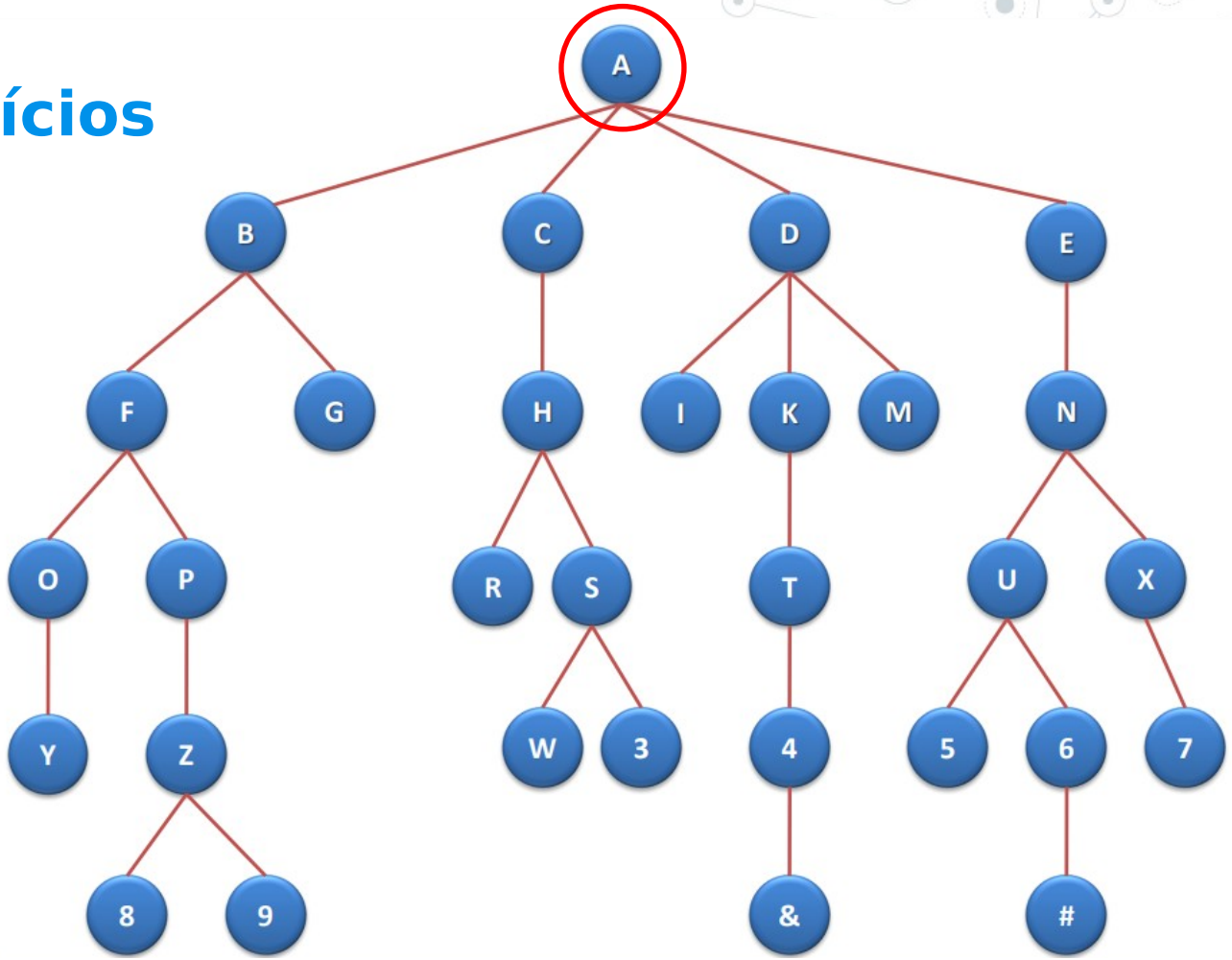
1.5. Exercícios

NÓ A:

Tipo:

Nível:

Ordem:



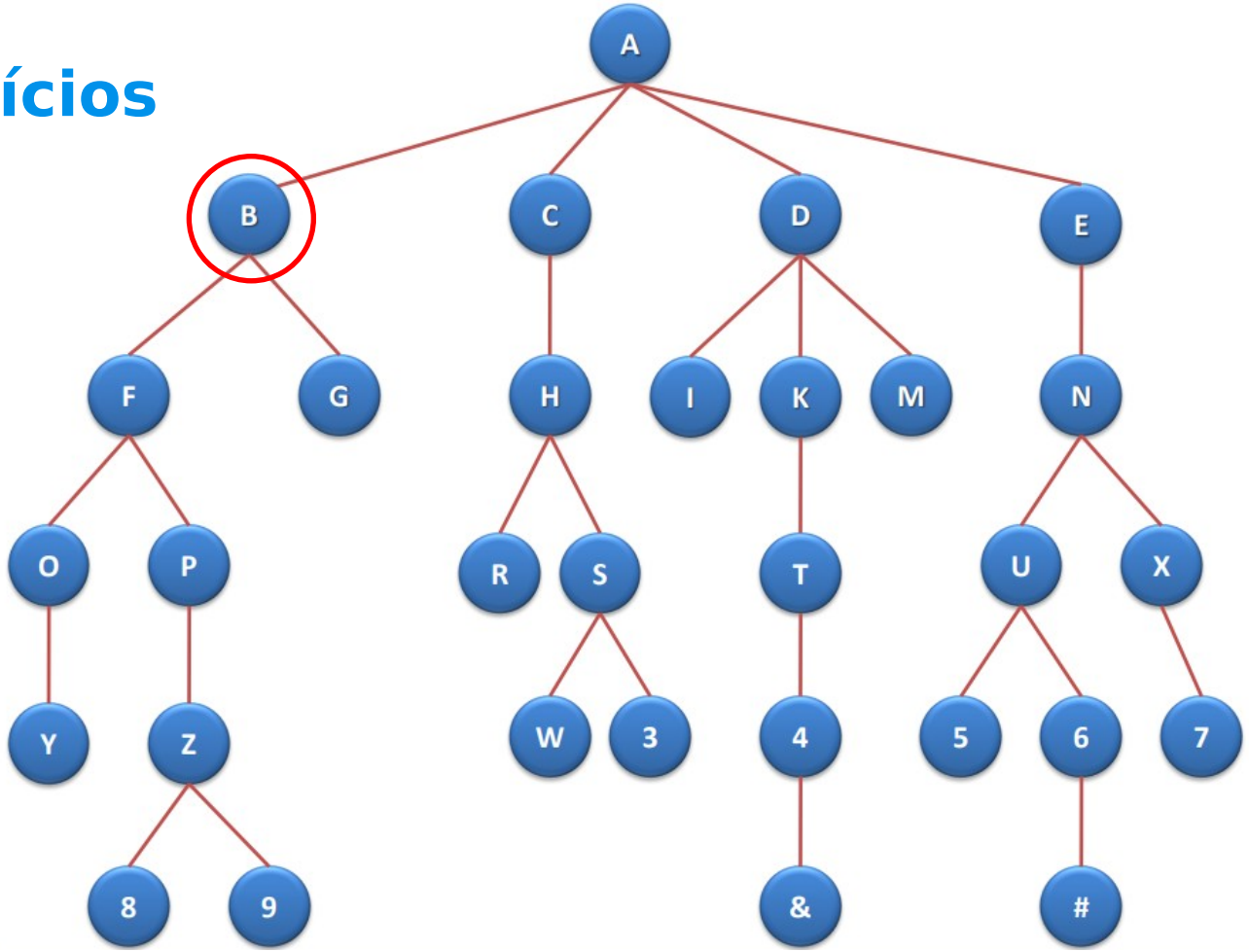
1.5. Exercícios

NÓ B:

Tipo:

Nível:

Ordem:

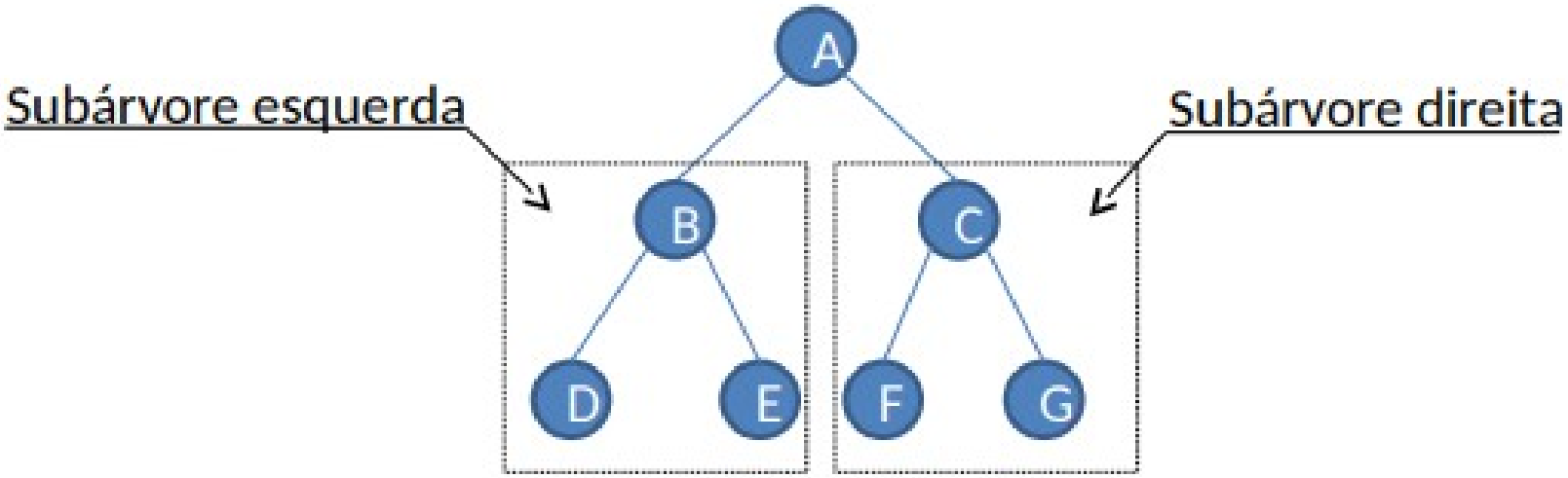


1.6. Árvores Binárias

- É muito útil para modelar situações em que precisam ser tomadas decisões bidirecionais em cada ponto de um processo.

- **Árvores Estritamente Binárias:** Todo nó possui exatamente dois nós filhos. Exceto os nos folha, que devem possuir exatamente 0 filhos

1.6. Árvores Binárias



1.6.1. Árvores Binárias Completas

- **Árvore Binária Completa de nível d**
  - Todas as folhas estejam no nível d

- Se contiver **m** nós no nível **l**, ela conterá no máximo **2m** nós no nível **l+1**
- Uma árvore binária completa de nível **d** contém exatamente **2^l** nós em cada nível **l** entre **0** e **d** (profundidade **d** com exatamente **2^d** nós no nível **d**)

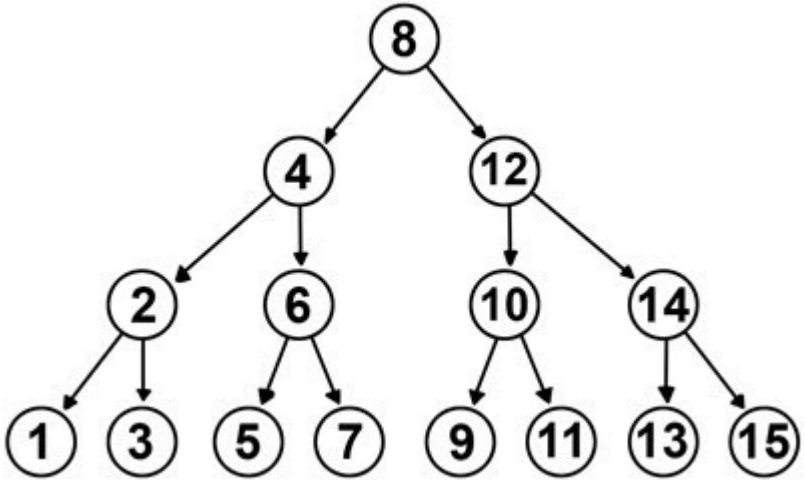
1.6.1. Árvores Binárias Completas

Nível (d)	n. max. nós	Potência
0	1	2^0
1	2	2^1
2	4	2^2
3	8	2^3
4	16	2^4
5	32	2^5

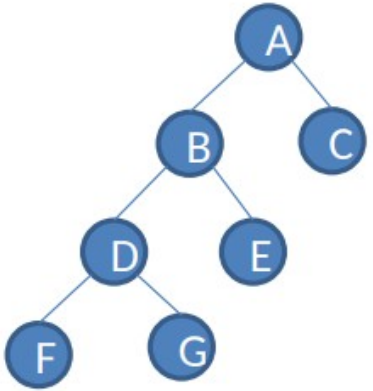
1.6.1. Árvores Binárias Completas

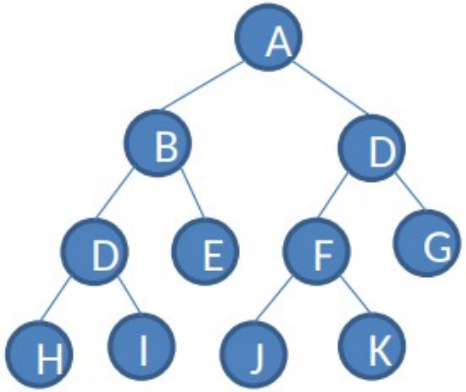
- **Número total de nós:**

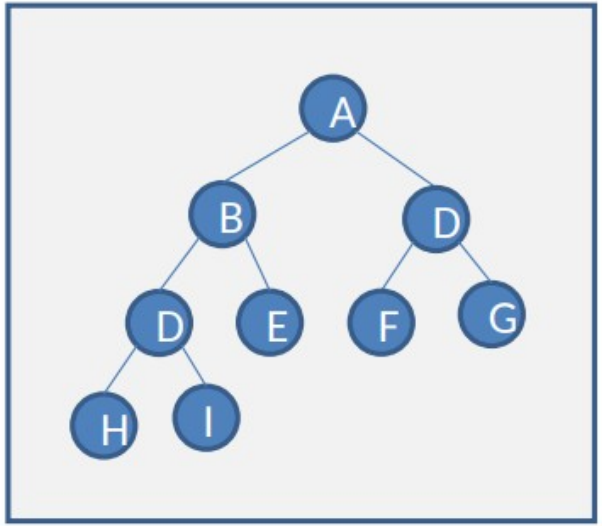
$$t_n = 2^0 + 2^1 + 2^2 + \ldots + 2^n = \sum_{j=0}^{d} 2^j \quad \xrightarrow{\text{Por indução}} \quad t_n = 2^{d+1} - 1$$



1.6.2. Árvore Binárias Quase Completas

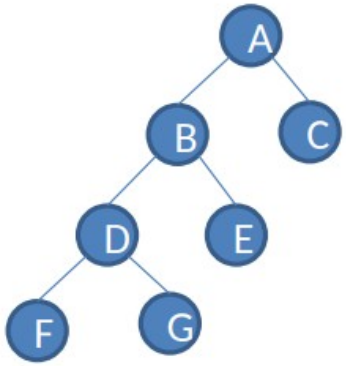


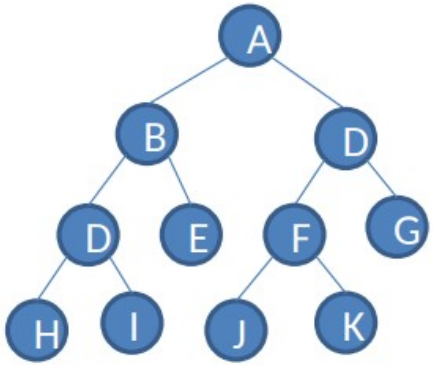


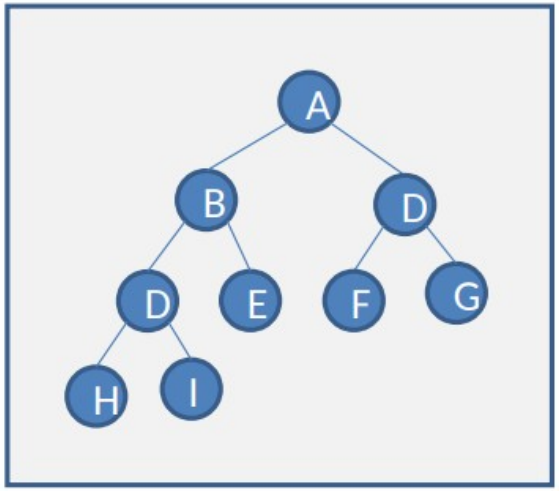


- Todas as folhas da árvore devem estar localizadas no **nível d** ou no **nível d-1**;

1.6.2. Árvore Binárias Quase Completas



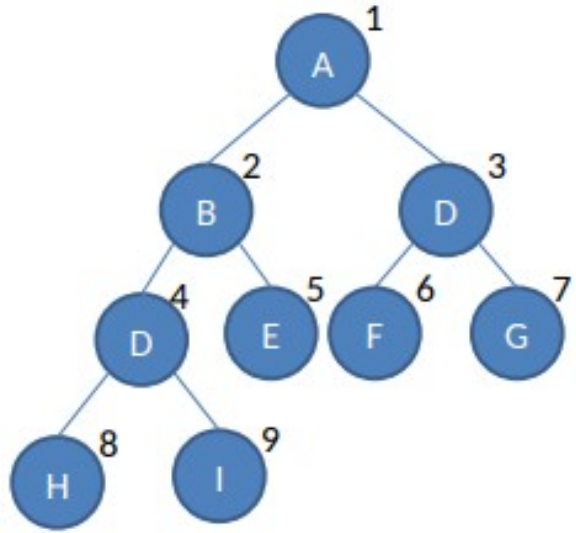




- Para cada nó **nd** na árvore com um descendente direto no nível **d**, todos os descendentes esquerdos de **nd** que forem folhas estiverem também no nível **d**.

1.6.2. Árvore Binárias Quase Completas

- Os nós de uma árvore binária quase completa podem ser numerados
  - 1 para a **raiz**
  - **Filho esquerdo** = dobro do n. do pai
  - **Filho direito** = dobro + 1 do n. do pai
- A numeração ajuda na implementação



A	B	D	D	E	F	G	H	I

1.7. Percorrendo Árvores Binárias

- Todos estes três métodos podem ser definidos recursivamente e se baseiam em três operações básicas:
  - **visitar a raiz,**
  - **percorrer a subárvore da esquerda e**
  - **percorrer a subárvore da direita.**
- A única diferença entre estes métodos é a ordem em que estas operações são executadas.

1.7. Percorrendo Árvores Binárias

- As 3 formas de percorrer uma árvore binária são:

  - **Caminhamento Pré-Ordem ou Pré-Fixado:**
    1. Visite a raiz;
    2. Visite a subárvore esquerda;
    3. Visite a subárvore direita.

1.7. Percorrendo Árvores Binárias

- As 3 formas de percorrer uma árvore binária são:

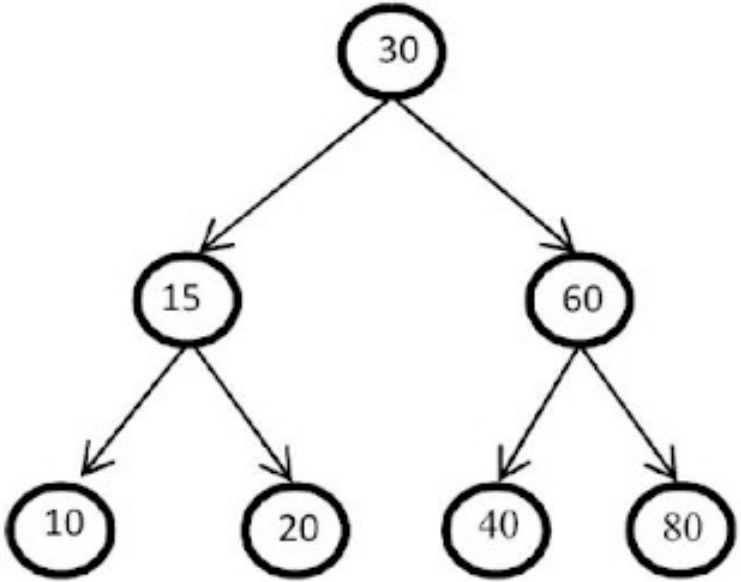
  - **Caminhamento Em ordem ou Central:**
    1. Visite a subárvore esquerda;
    2. Visite a raiz;
    3. Visite a subárvore direita.

1.7. Percorrendo Árvores Binárias

- As 3 formas de percorrer uma árvore binária são:

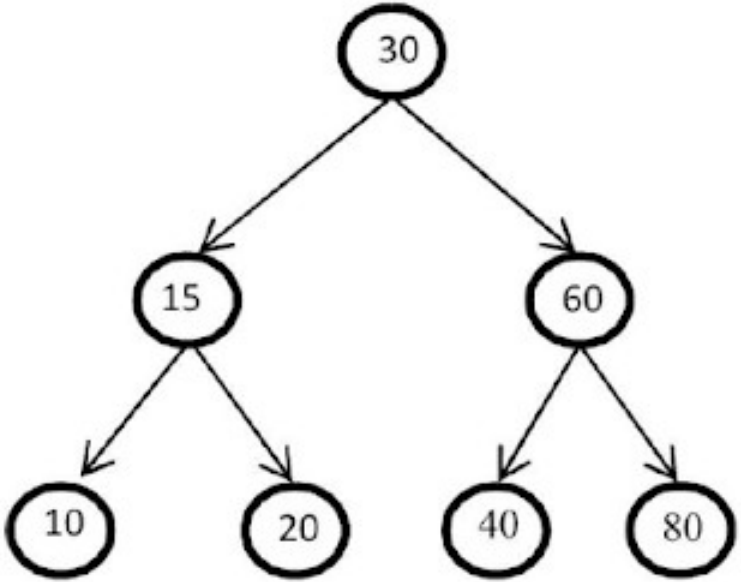
  - **Caminhamento Pós-Ordem ou Pós-Fixado:**
    1. Visite a subárvore esquerda;
    2. Visite a subárvore direita;
    3. Visite a raiz.

1.7. Percorrendo Árvores Binárias



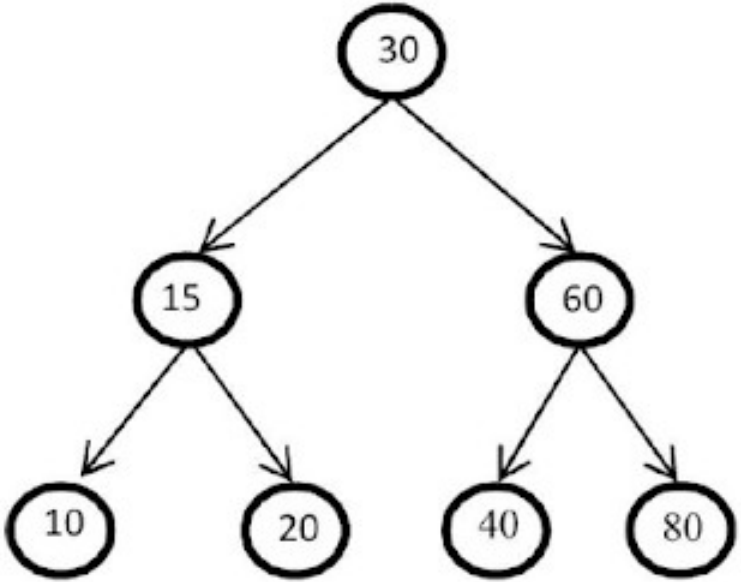
PRÉ-ORDEM:

1.7. Percorrendo Árvores Binárias



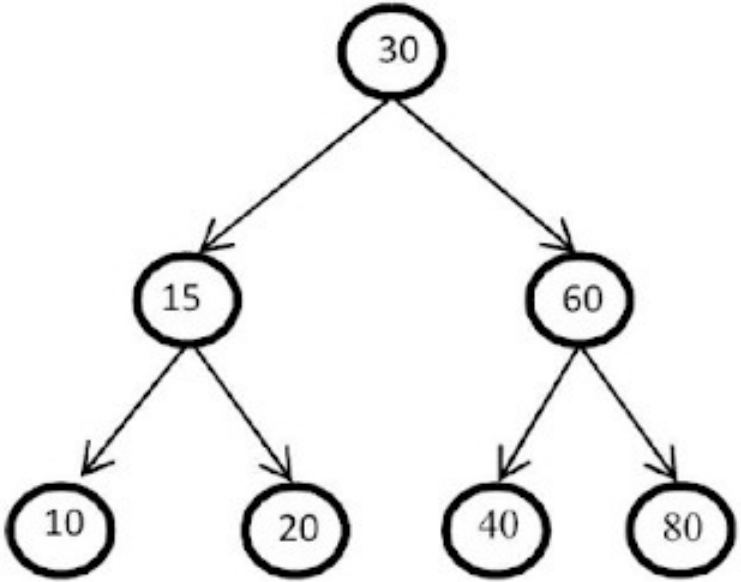
PRÉ-ORDEM: 30, 15, 10, 20, 60, 40, 80.

1.7. Percorrendo Árvores Binárias



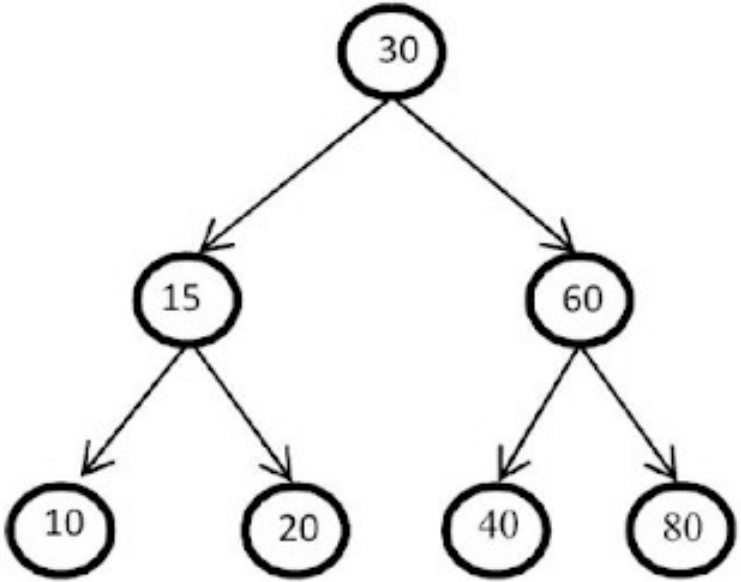
EM ORDEM:

1.7. Percorrendo Árvores Binárias



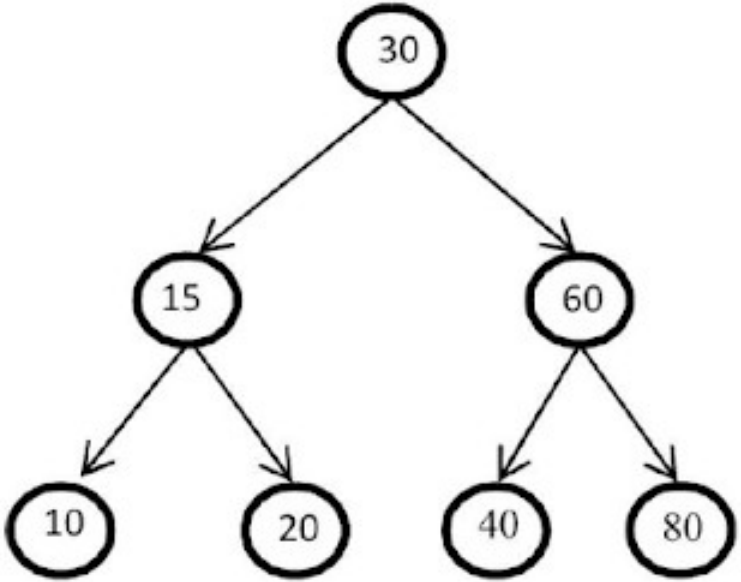
EM ORDEM : 10, 15, 20, 30, 40, 60, 80.

1.7. Percorrendo Árvores Binárias



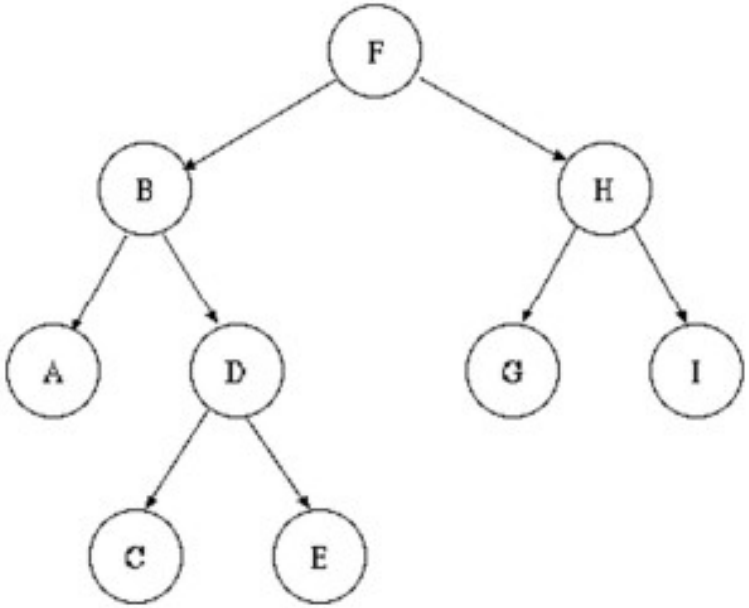
PÓS-ORDEM:

1.7. Percorrendo Árvores Binárias



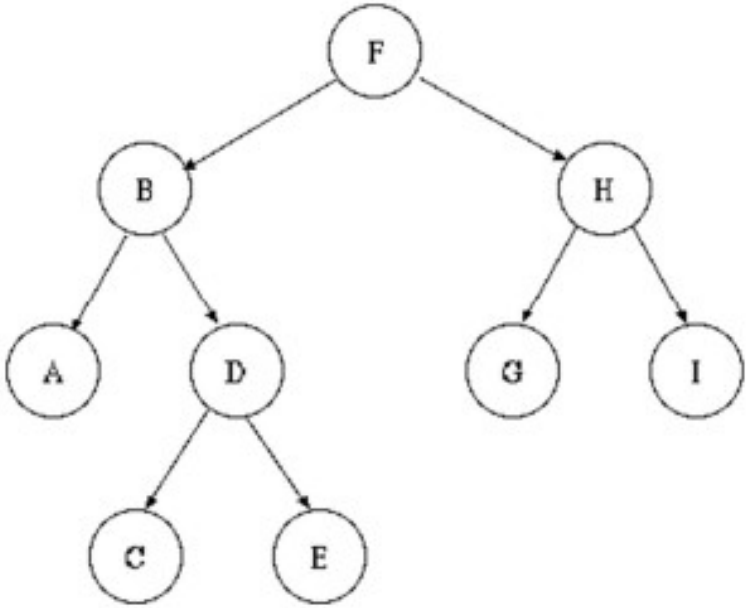
PÓS-ORDEM: 10, 20, 15, 40, 80, 60, 30.

1.7. Percorrendo Árvores Binárias



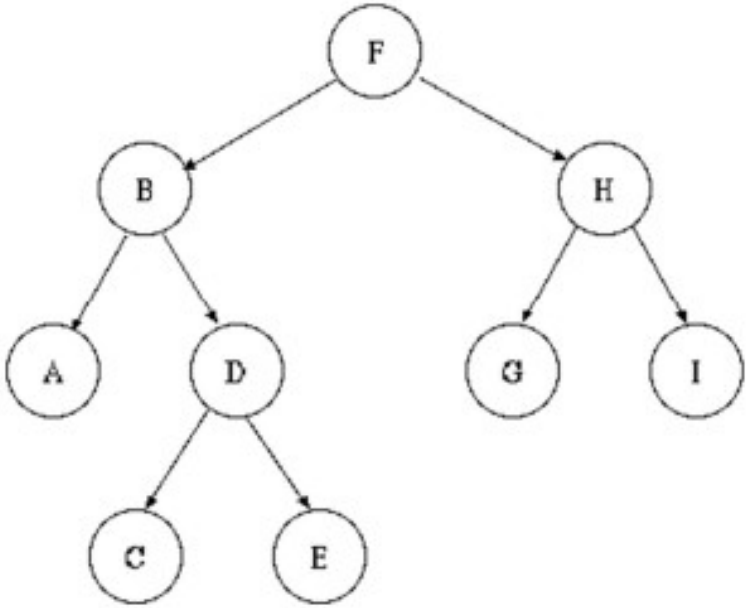
PRÉ-ORDEM:

1.7. Percorrendo Árvores Binárias



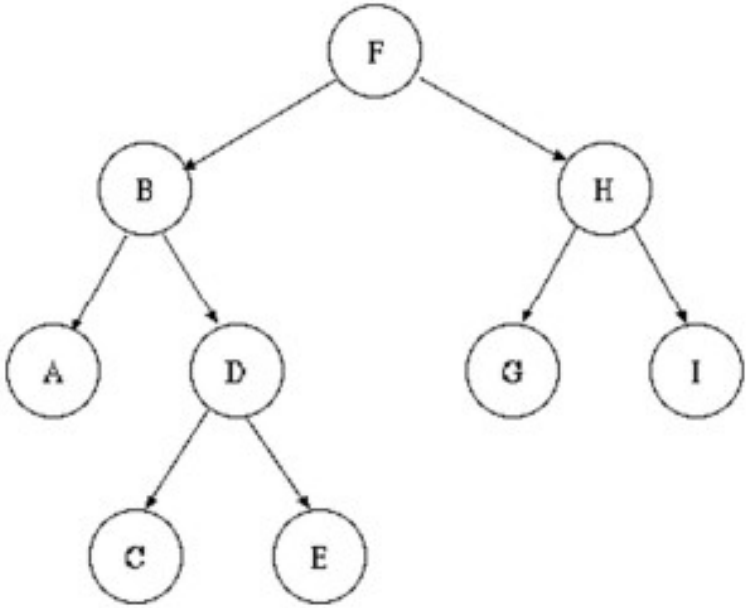
PRÉ-ORDEM: F, B, A, D, C, E, H, G, I.

1.7. Percorrendo Árvores Binárias



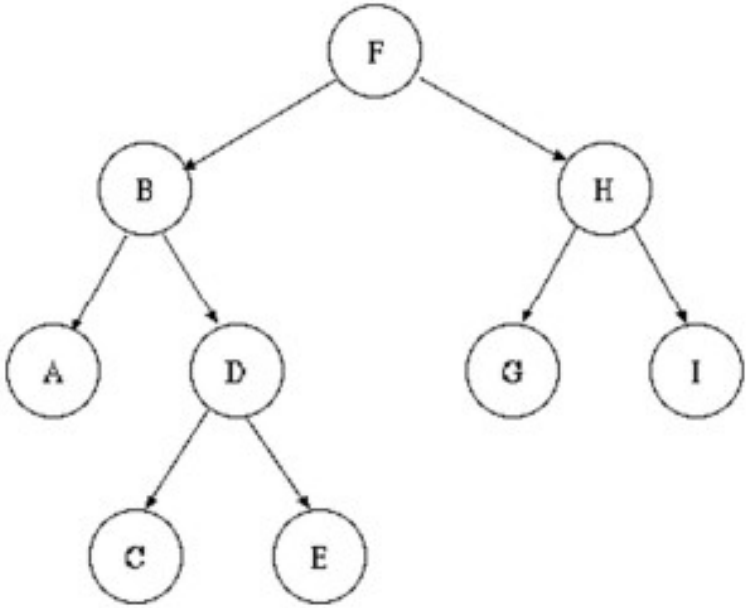
EM ORDEM:

1.7. Percorrendo Árvores Binárias



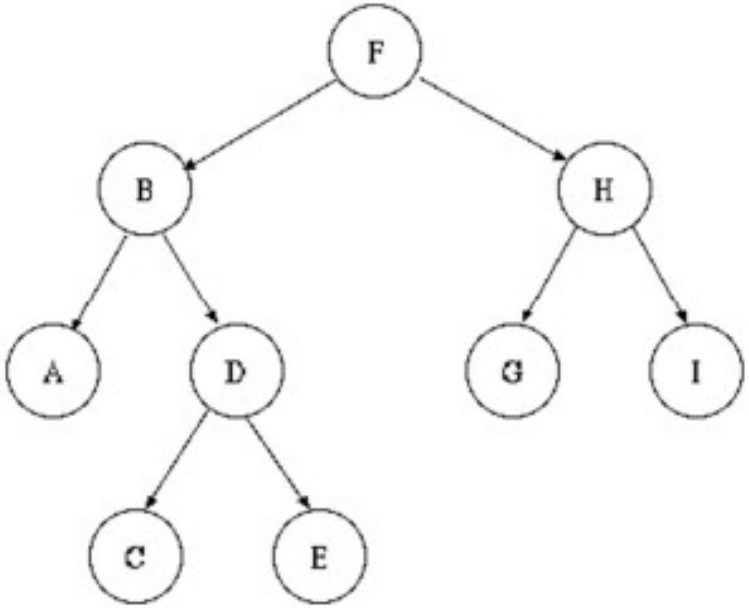
EM ORDEM: A, B, C, D, E, F, G, H, I.

1.7. Percorrendo Árvores Binárias



PÓS-ORDEM:

1.7. Percorrendo Árvores Binárias



PÓS-ORDEM: A, C, E, D, B, G, I, H, F.

1.8. Árvore Binárias - Implementação

- Assim como vimos em nosso estudo sobre listas, árvores também podem ser armazenadas de forma estática ou dinâmica.

- Começaremos nosso estudo com o armazenamento estático.

Na próxima aula.

RES PIRA !

Obrigado!

Perguntas?

 heraldo.junior@ifsertao-pe.edu.br