

# Unidade V: Árvore Binária - Introdução



Instituto de Ciências Exatas e Informática  
Departamento de Ciência da Computação

# Agenda

- Definições e conceitos
- Classe Nó em Java
- Classe Árvore Binária em Java

# Agenda

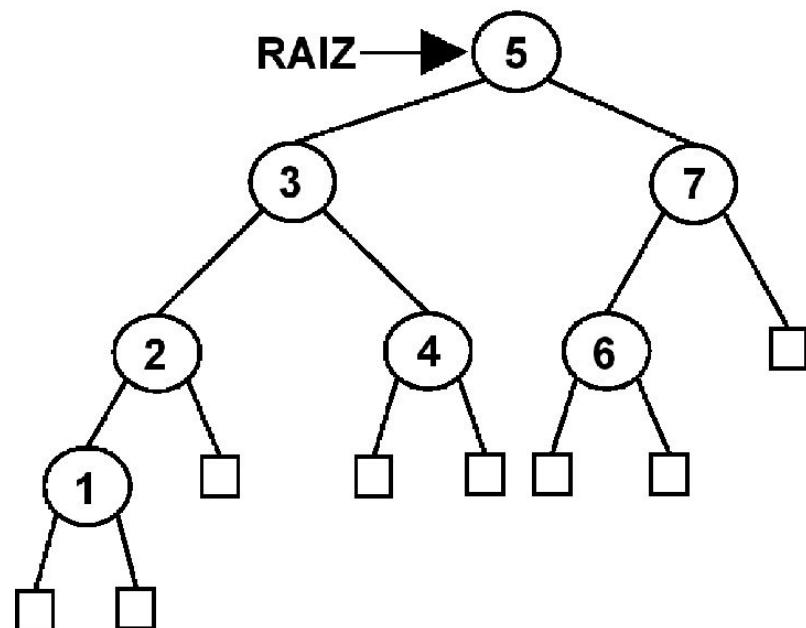
- **Definições e conceitos** 
- Classe Nó em Java
- Classe Árvore Binária em Java

# Introdução

- Custo de inserção, remoção e pesquisa nas listas:  $\Theta(n)$  comparações
- Custo de pesquisa na lista sequencial ordenada:  $\Theta(\lg(n))$  comparações
  - Inserção e remoção nesta lista:  $\Theta(n)$  comparações

# Árvore

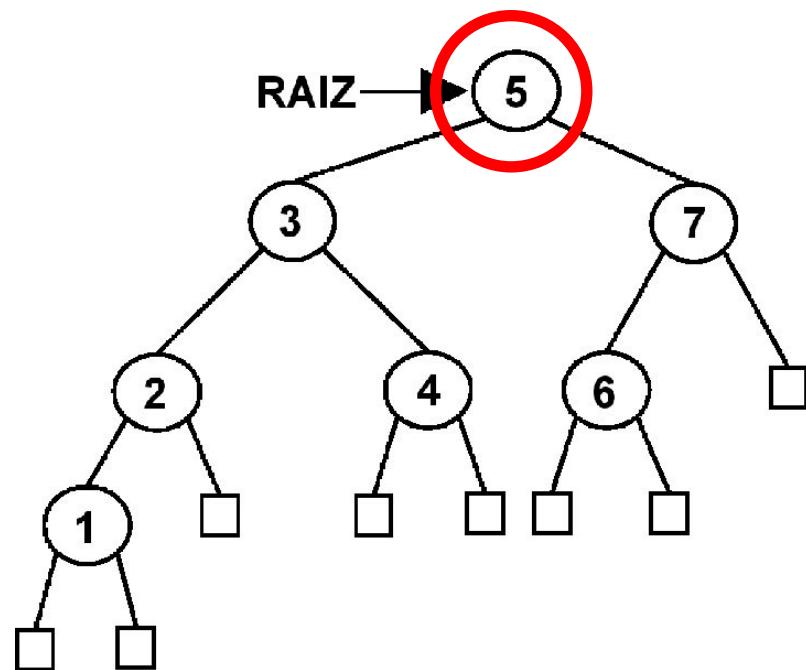
- Custo de inserção, remoção e pesquisa **pode** ser  $\Theta(\lg(n))$  comparações
- Formada por um conjunto finito nós (vértices) conectados por arestas



# Árvore

- Custo de inserção, remoção e pesquisa pode ser  $\Theta(\lg(n))$  comparações
- Formada por um conjunto finito nós (vértices) conectados por arestas

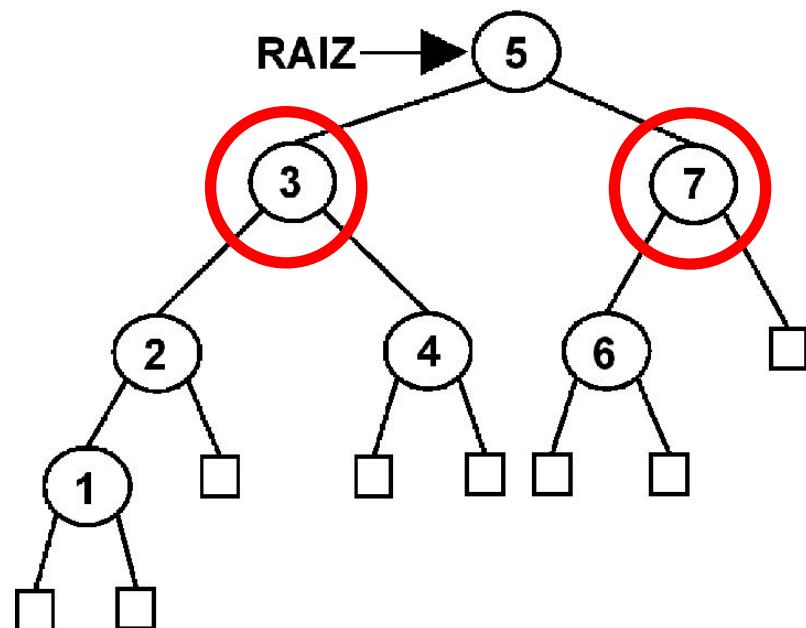
O nó 5 é denominado nó raiz e ele está no nível 0



# Árvore

- Custo de inserção, remoção e pesquisa pode ser  $\Theta(\lg(n))$  comparações
- Formada por um conjunto finito nós (vértices) conectados por arestas

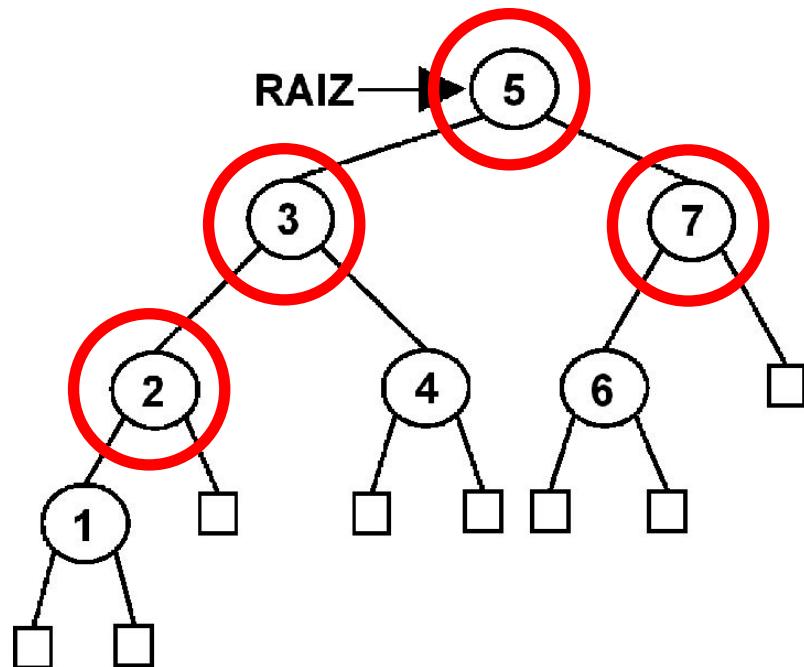
Os nós 3 e 7 são filhos do 5 e esse é pai dos dois primeiros



# Árvore

- Custo de inserção, remoção e pesquisa pode ser  $\Theta(\lg(n))$  comparações
- Formada por um conjunto finito nós (vértices) conectados por arestas

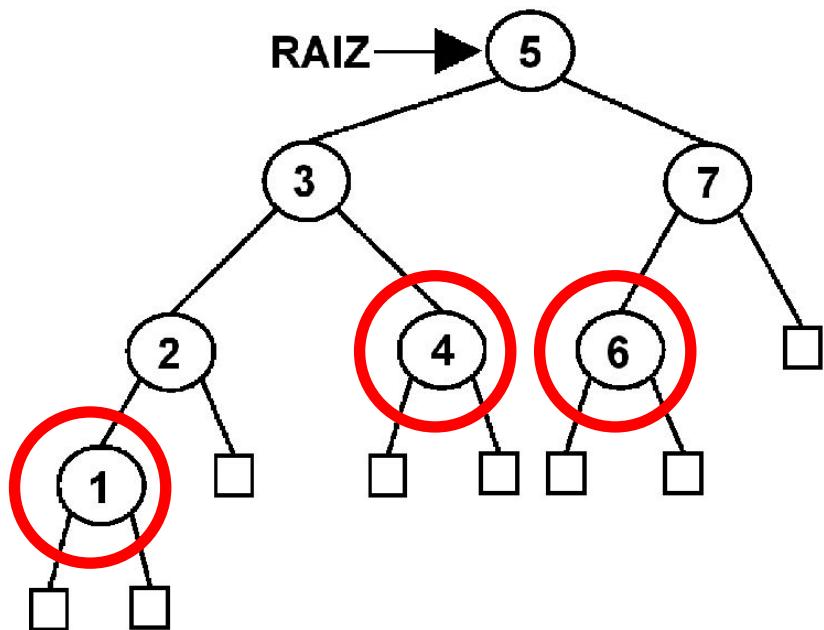
Um nó com filho(s) é chamado de **nó interno** e outro sem, de folha



# Árvore

- Custo de inserção, remoção e pesquisa pode ser  $\Theta(\lg(n))$  comparações
- Formada por um conjunto finito nós (vértices) conectados por arestas

Um nó com filho(s) é chamado de nó interno e outro sem, de **folha**



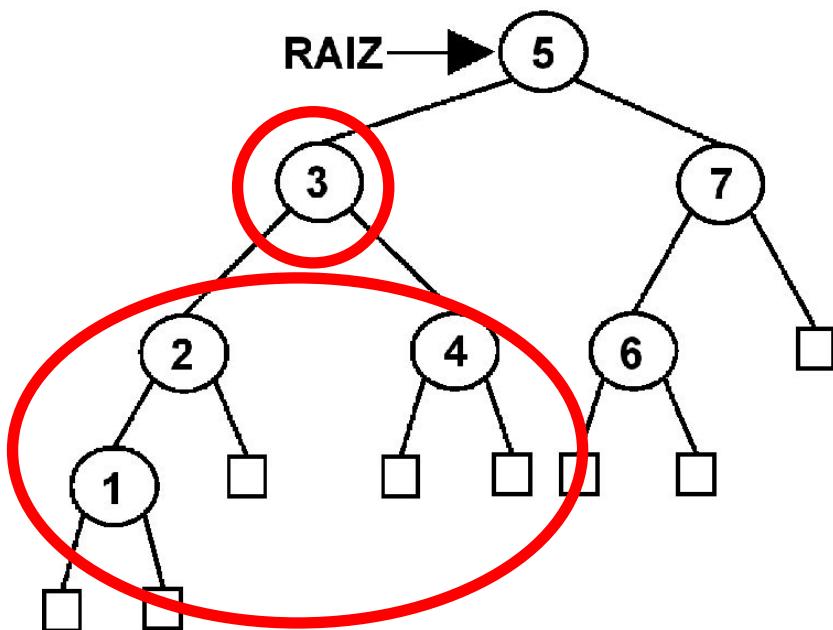
# Descobriram Nossa Árvore



# Árvore

- Custo de inserção, remoção e pesquisa pode ser  $\Theta(\lg(n))$  comparações
- Formada por um conjunto finito nós (vértices) conectados por arestas

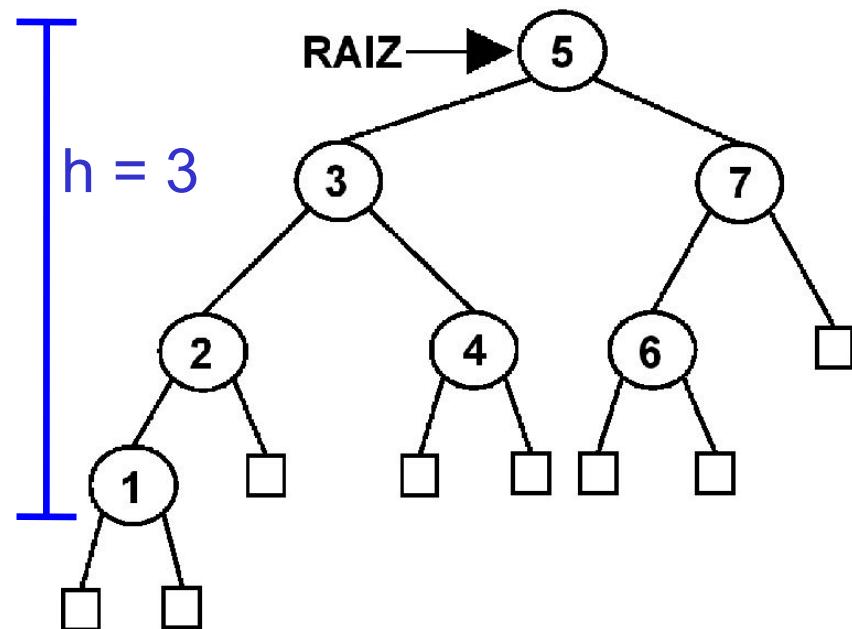
Os nós 1, 2 e 4 formam uma subárvore com raiz no nó 3



# Árvore

- Custo de inserção, remoção e pesquisa pode ser  $\Theta(\lg(n))$  comparações
- Formada por um conjunto finito nós (vértices) conectados por arestas

**Altura (h):** maior distância entre um nó e a raiz

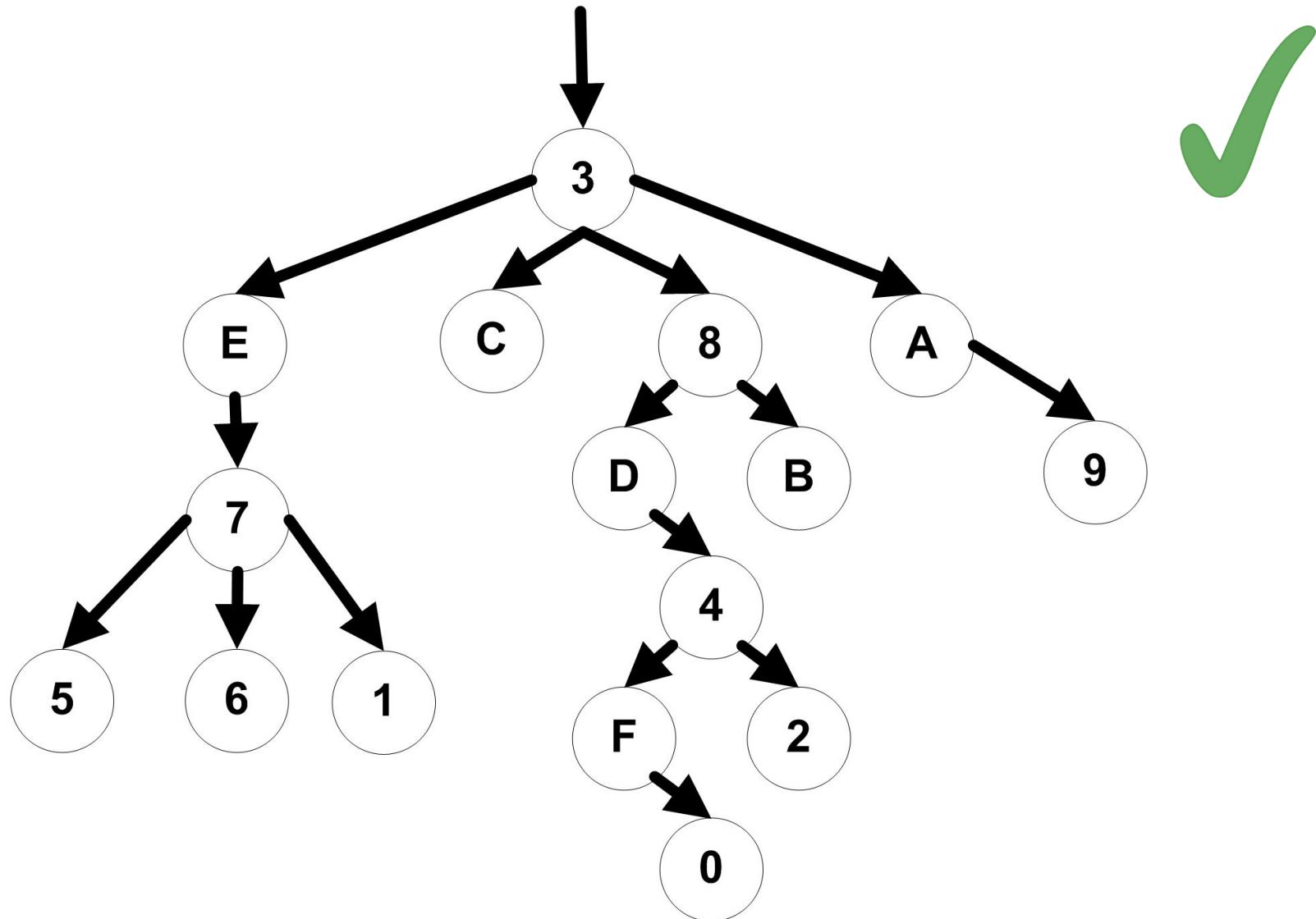


# Exercício Resolvido (1)

- Crie uma árvore com os dígitos hexadecimais (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F)

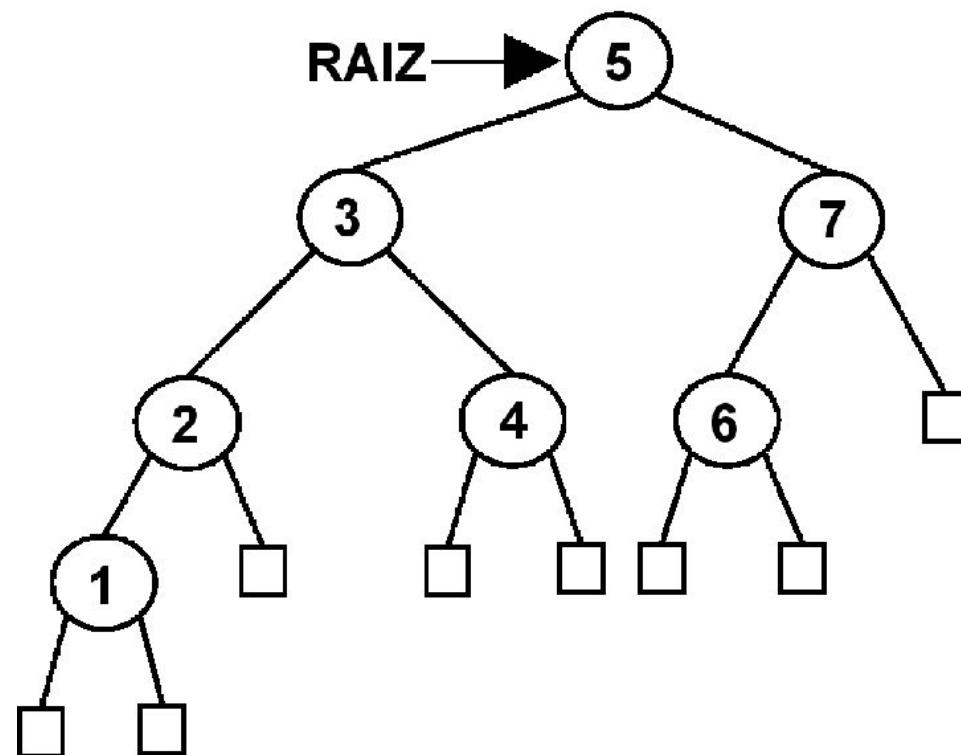
# Exercício Resolvido (1)

- Crie uma árvore com os dígitos hexadecimais (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F)



# Árvore Binária

- Árvore em que cada nó possui **no máximo dois filhos**, por exemplo:

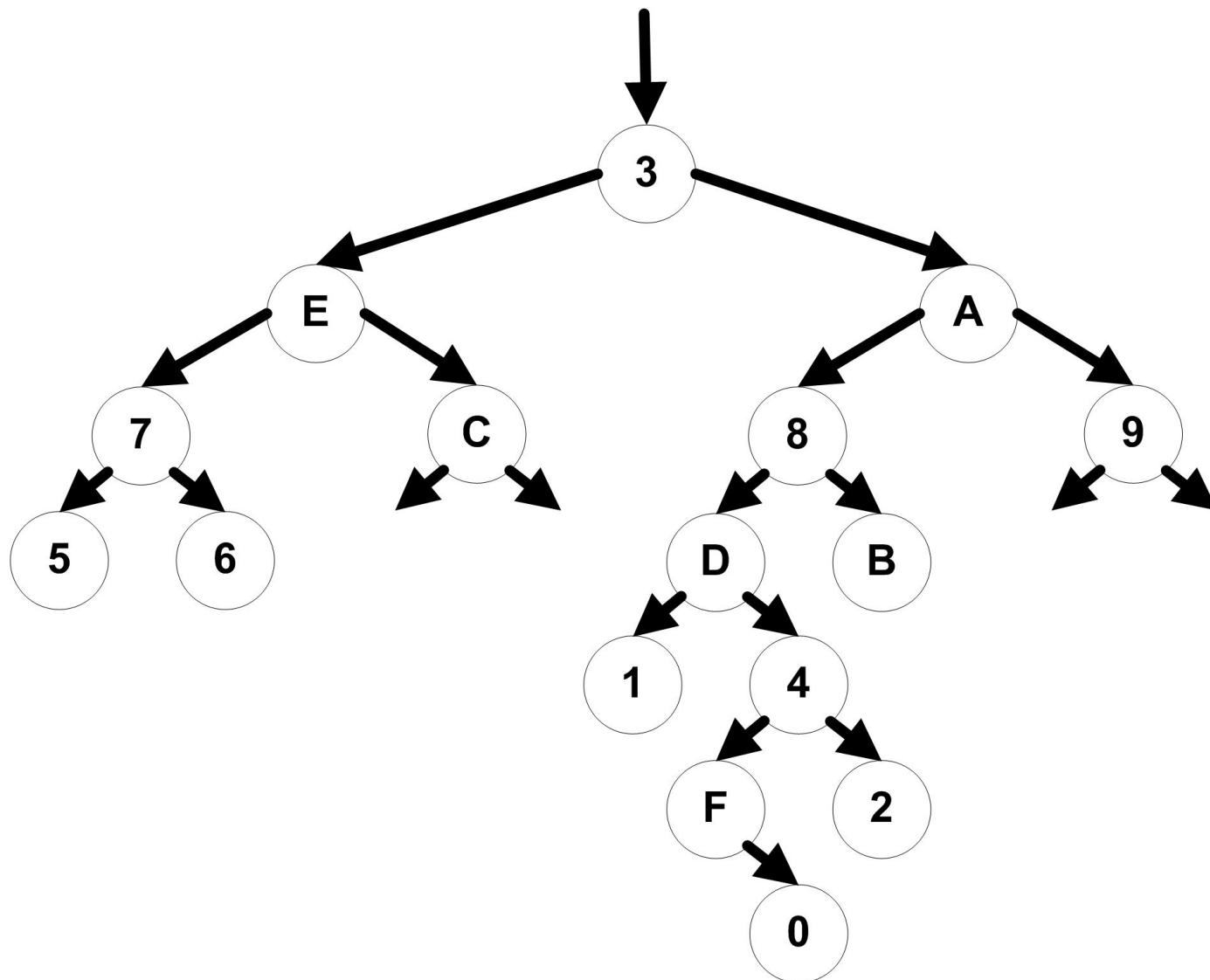


# Exercício Resolvido (2)

- Crie uma árvore binária com os dígitos hexadecimais

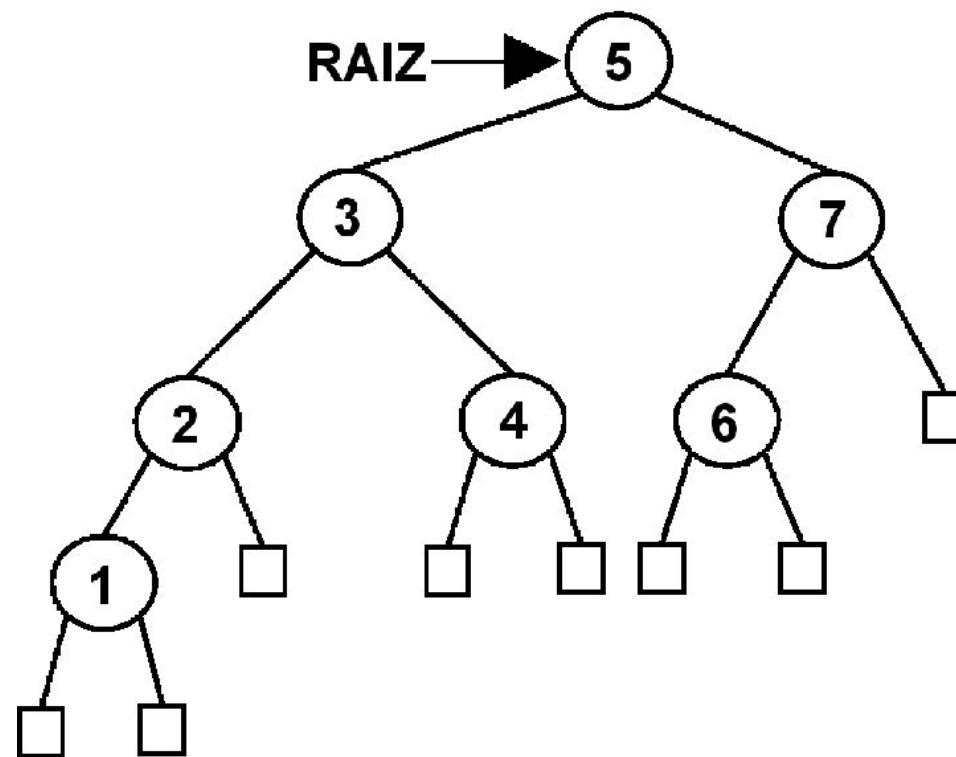
# Exercício Resolvido (2)

- Crie uma árvore binária com os dígitos hexadecimais



# Árvore Binária de Pesquisa

- Árvore binária em que cada nó é maior que todos seus vizinhos à esquerda e menor que todos à direita. Por exemplo:

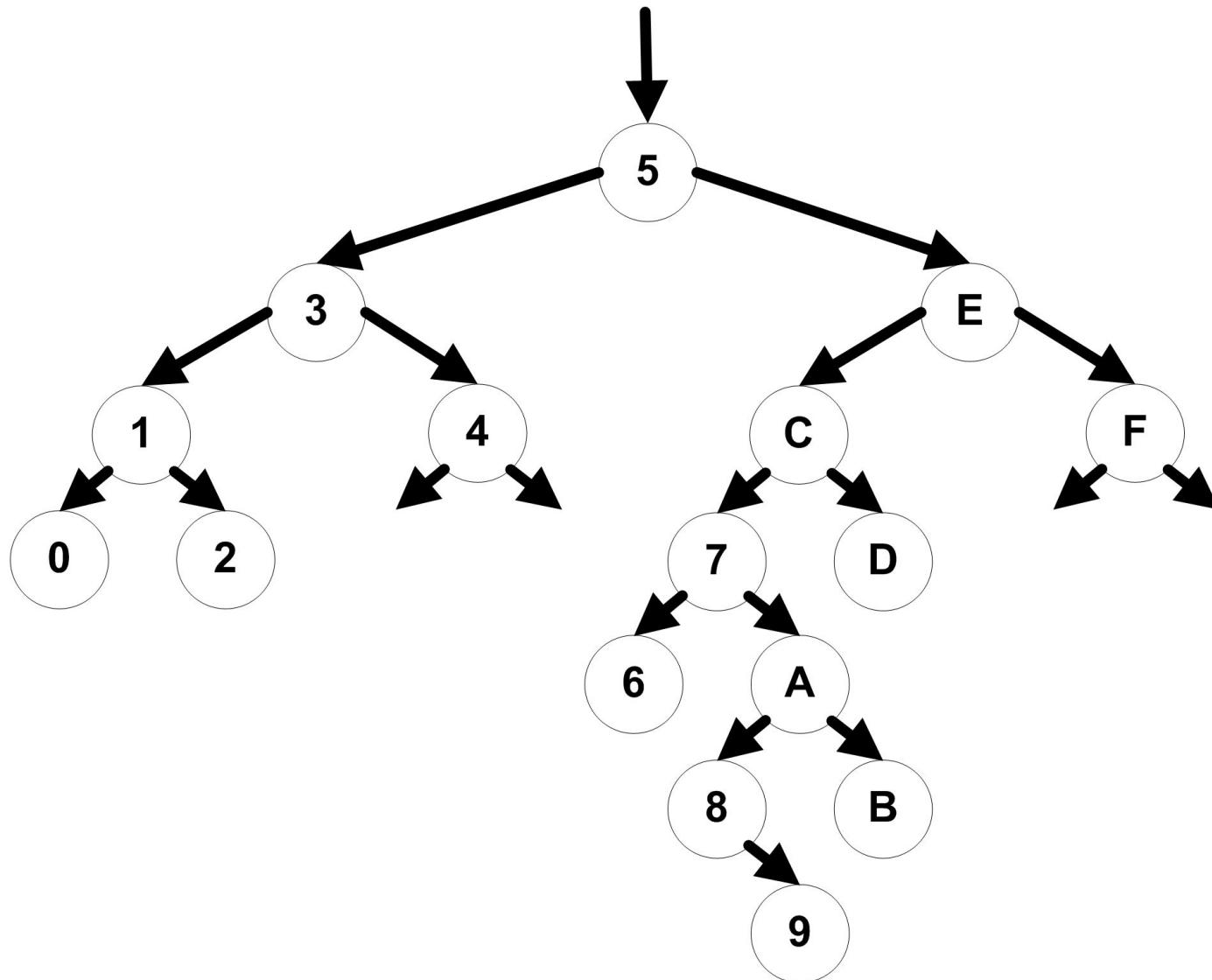


# Exercício Resolvido (3)

- Crie uma árvore binária de pesquisa com os dígitos hexadecimais

# Exercício Resolvido (3)

- Crie uma árvore binária de pesquisa com os dígitos hexadecimais



# Árvore Binária Completa

- Árvore binária em que:
  - Cada nó é uma folha **OR** possui exatamente dois filhos
  - Todos os nós folhas possuem uma altura  $h$
  - O número de nós internos é  $2^h - 1$
  - O número de nós folhas é  $2^h$
  - O número total de nós é  $(2^h - 1) + (2^h) = 2^{(h+1)} - 1$

# Exercício Resolvido (4)

- Crie uma árvore binária completa com os dígitos hexadecimais (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F)

## Exercício Resolvido (4)

- Crie uma árvore binária completa com os dígitos hexadecimais (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F)



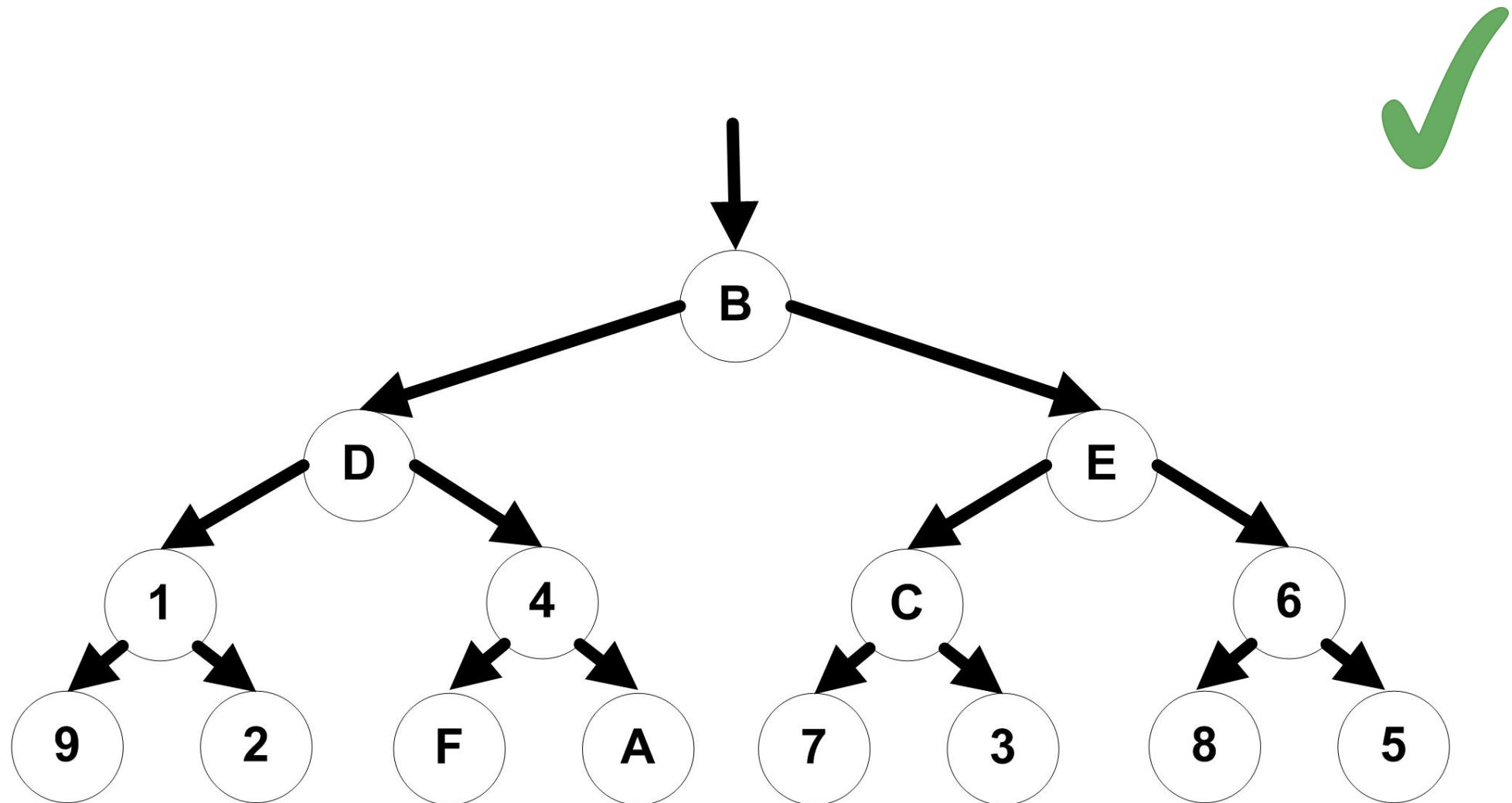
Impossível, pois o número de nós  
**é uma potência de dois**  
(está sobrando um nó)

# Exercício Resolvido (5)

- Crie uma árvore binária completa com os dígitos hexadecimais não nulos

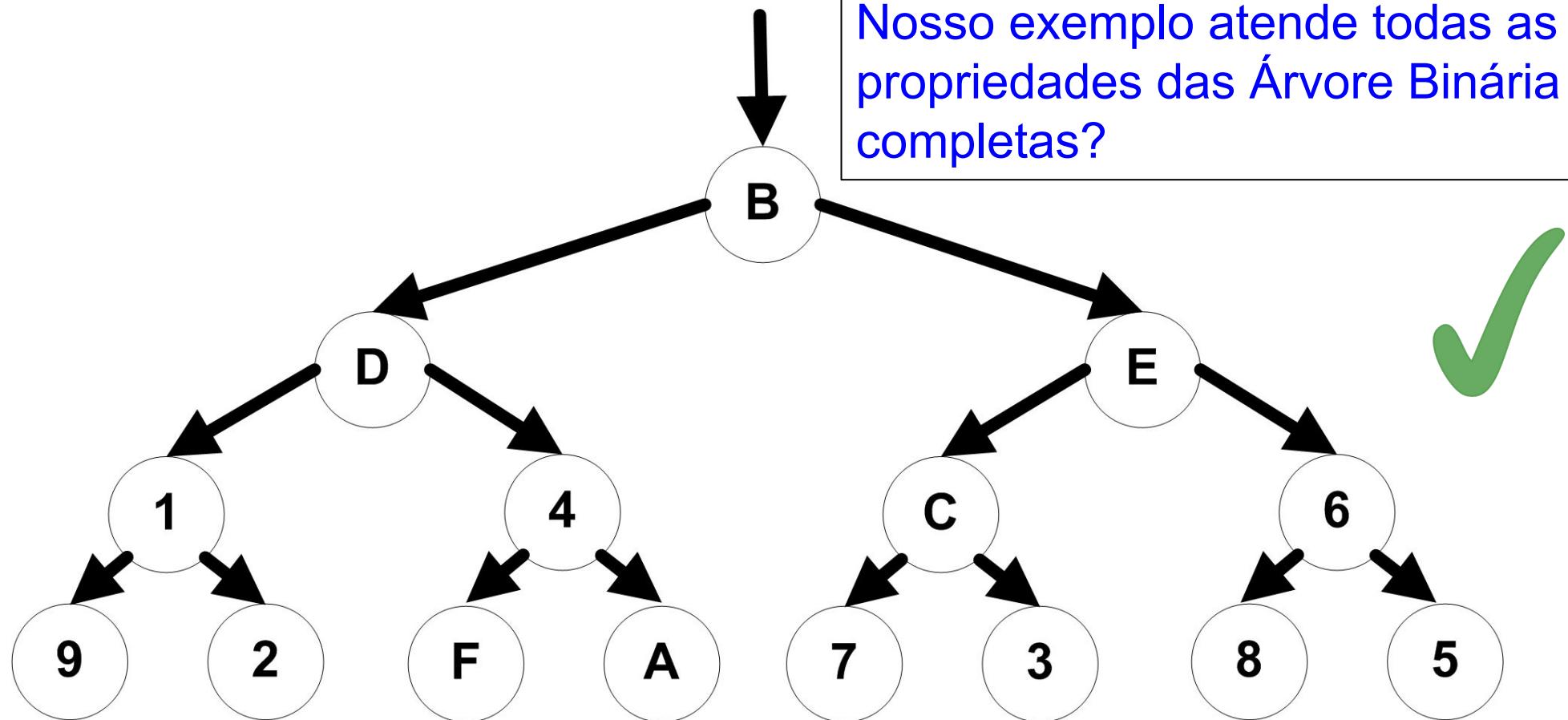
## Exercício Resolvido (5)

- Crie uma árvore binária completa com os dígitos hexadecimais não nulos



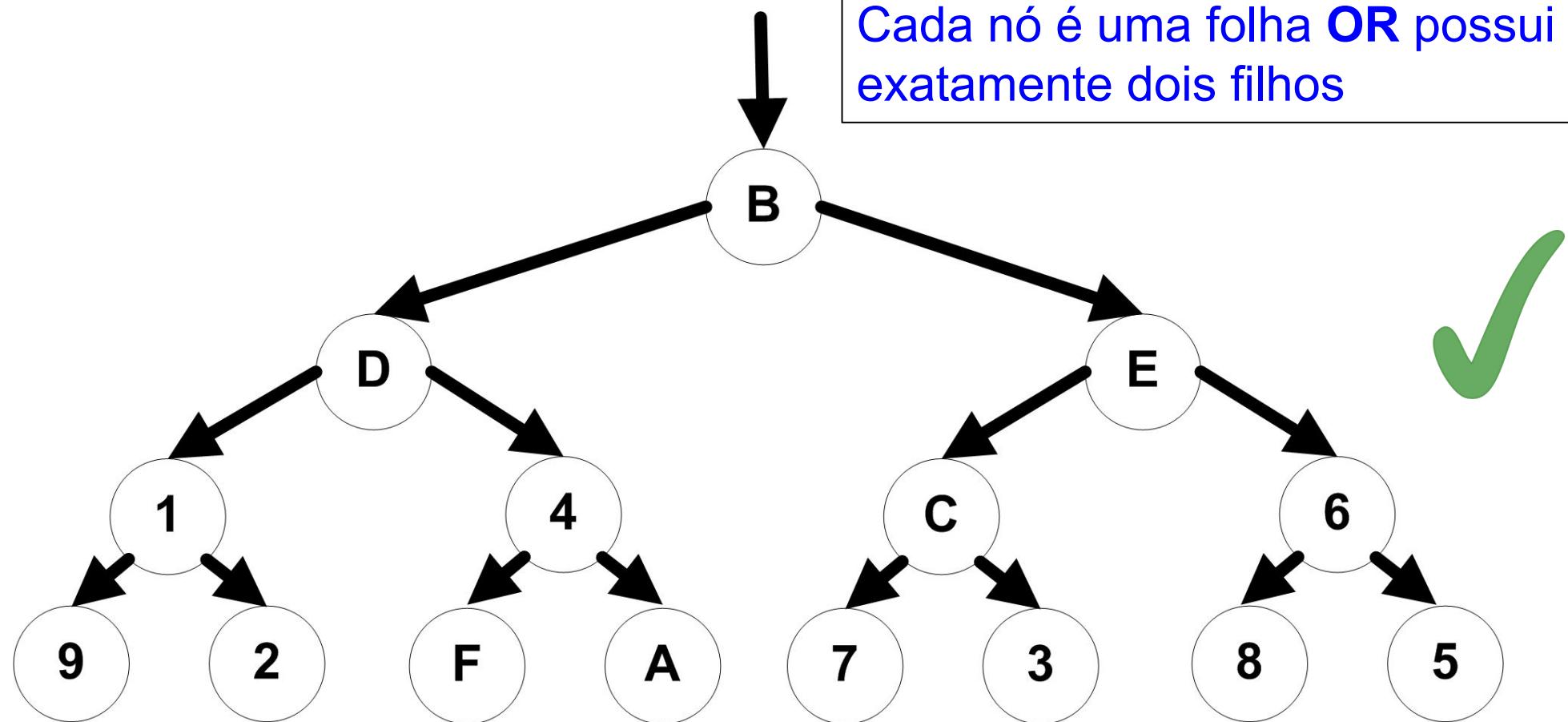
## Exercício Resolvido (5)

- Crie uma árvore binária completa com os dígitos hexadecimais não nulos



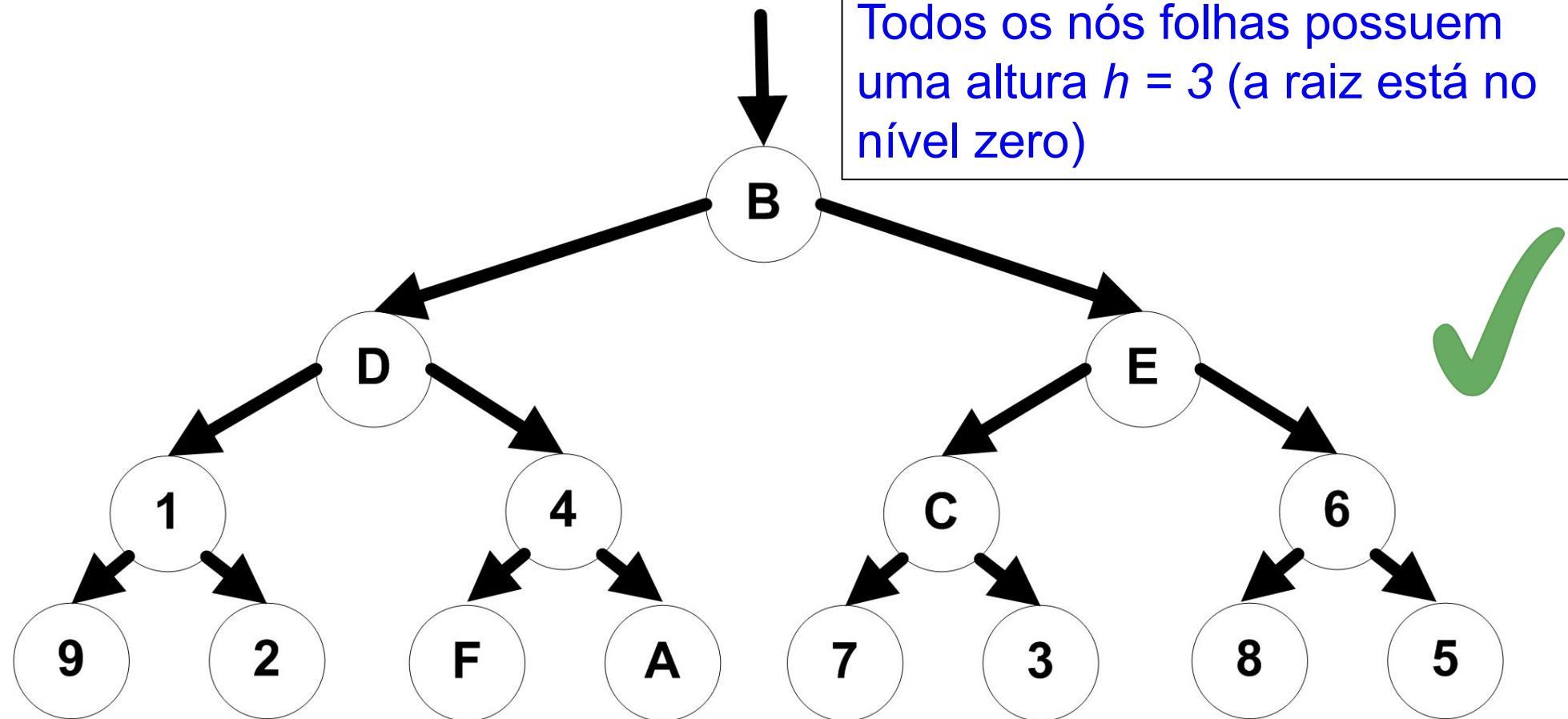
## Exercício Resolvido (5)

- Crie uma árvore binária completa com os dígitos hexadecimais não nulos



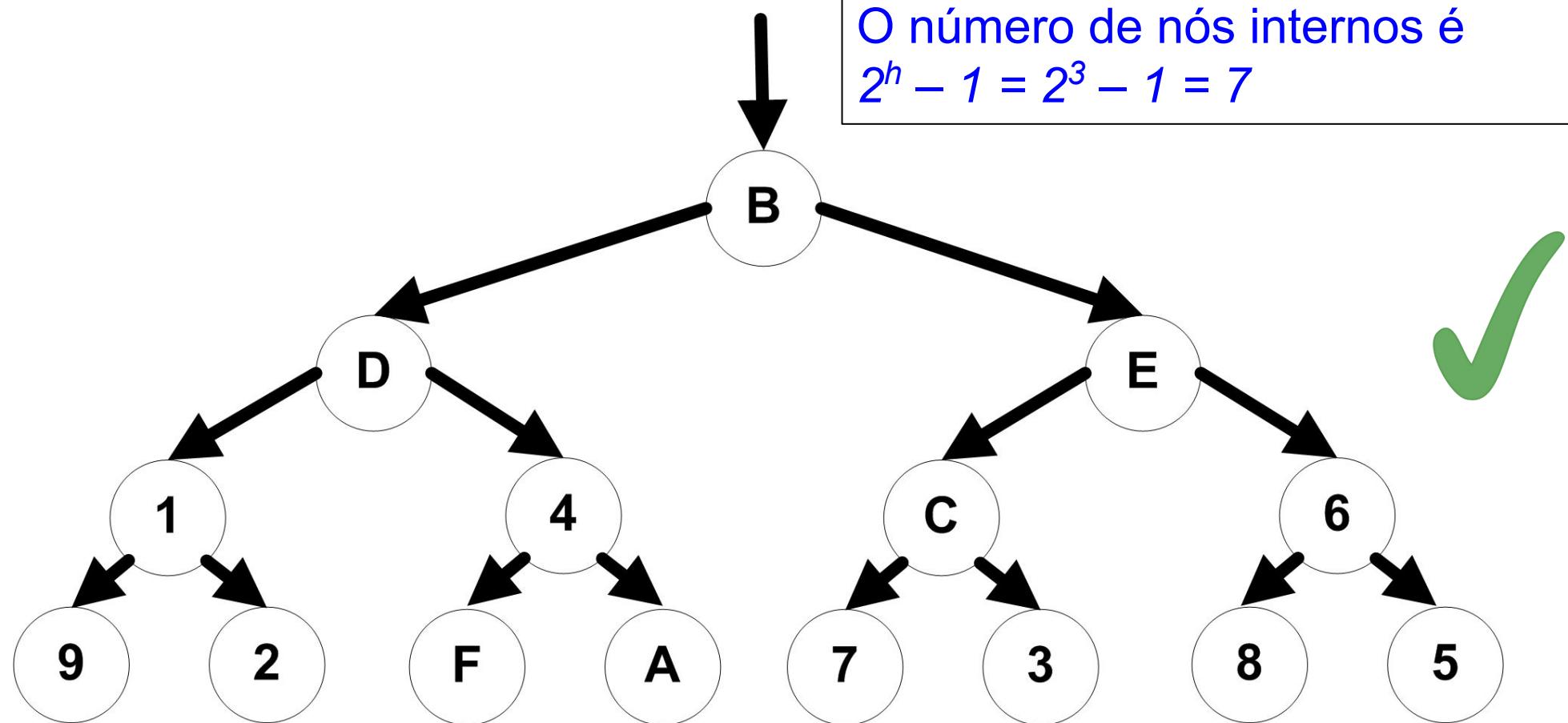
## Exercício Resolvido (5)

- Crie uma árvore binária completa com os dígitos hexadecimais não nulos



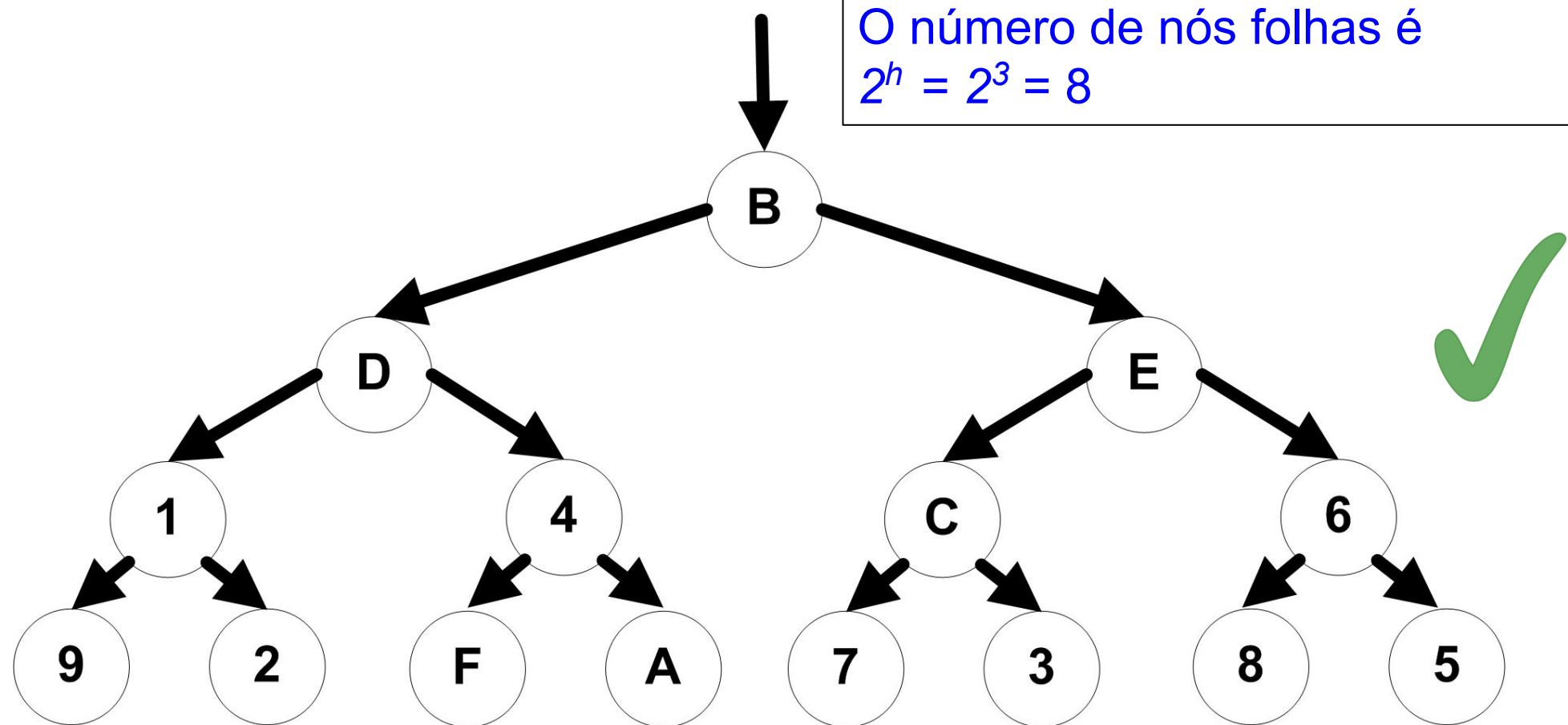
## Exercício Resolvido (5)

- Crie uma árvore binária completa com os dígitos hexadecimais não nulos



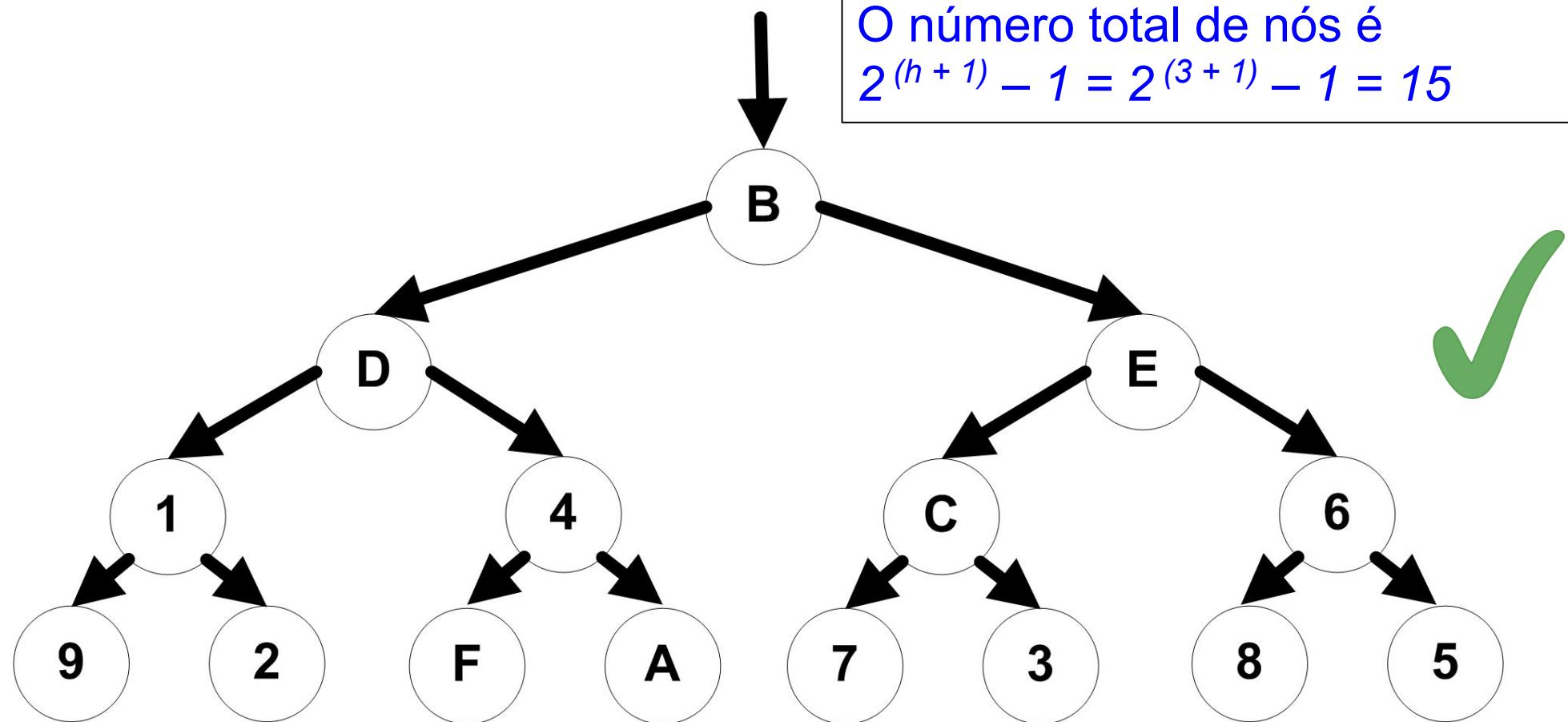
## Exercício Resolvido (5)

- Crie uma árvore binária completa com os dígitos hexadecimais não nulos



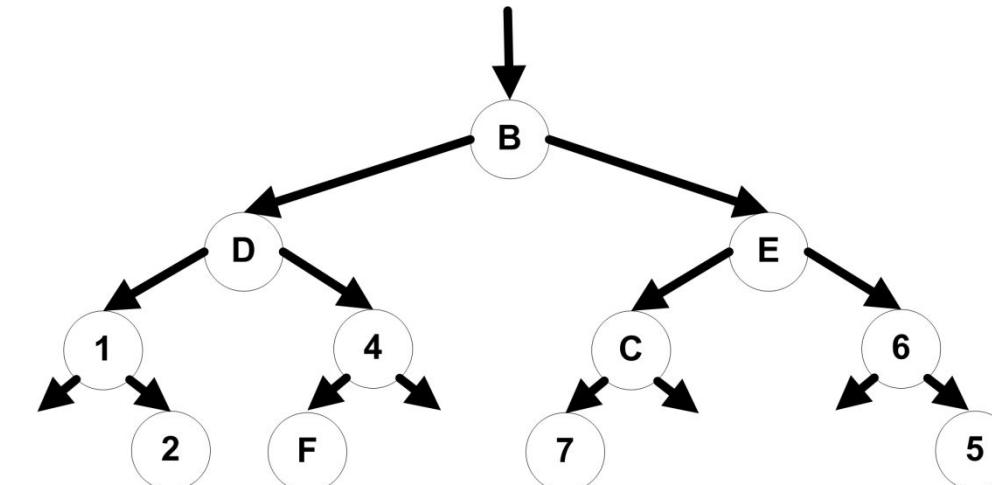
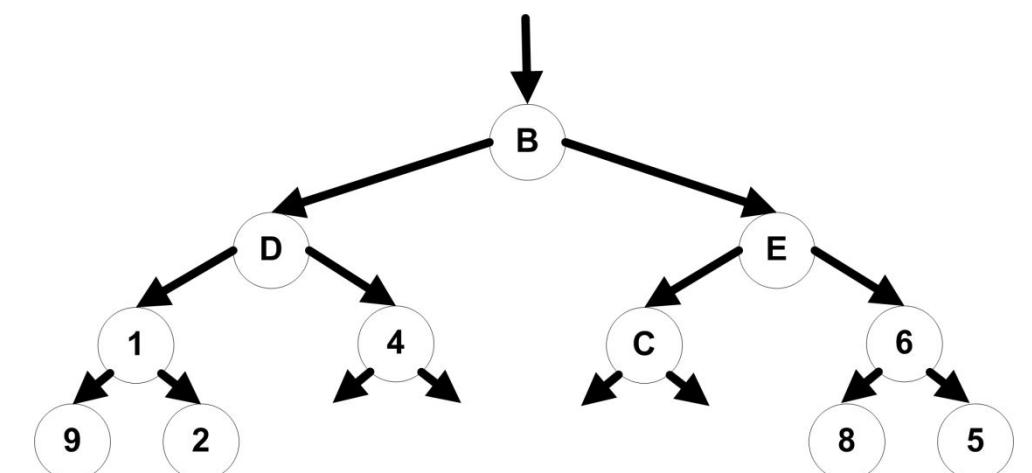
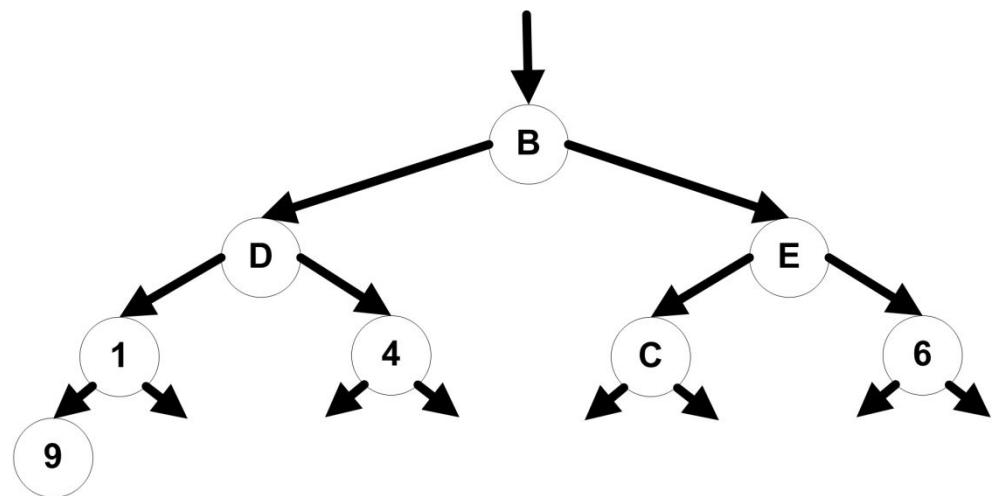
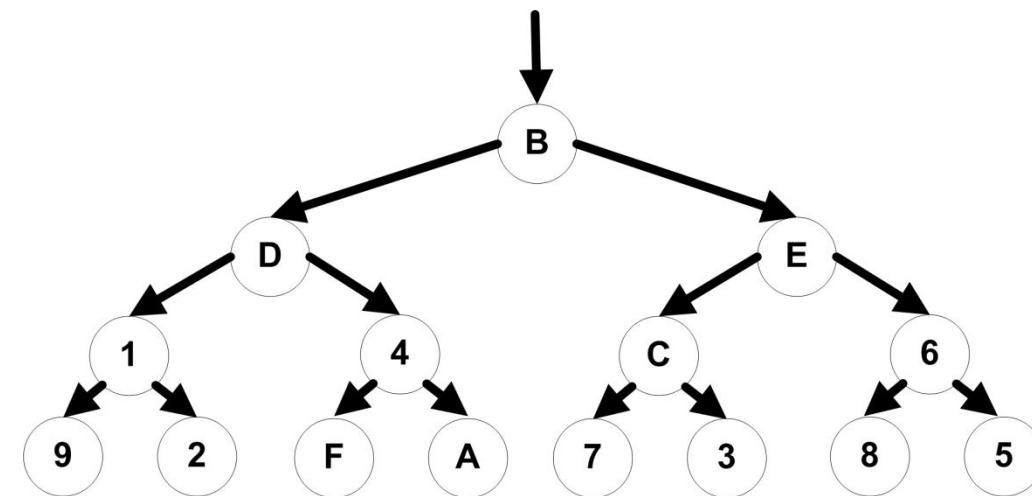
## Exercício Resolvido (5)

- Crie uma árvore binária completa com os dígitos hexadecimais não nulos



# Árvore Balanceada

- Árvore em que para **TODOS** os nós, a diferença entre a altura de suas árvores da esquerda e da direita sempre será **0** ou  **$\pm 1$**  como, por exemplo:



# Consideração

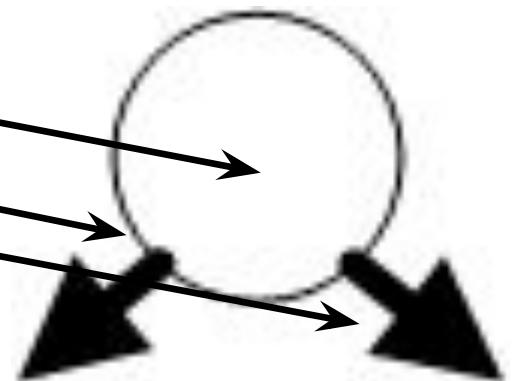
- Árvore Binária de Pesquisa (ABP) também é chamada de Árvore Binária de Busca (ABB) ou *Binary Search Tree* (BST)
- A partir deste ponto, neste material, considera-se que todas as árvores binárias serão de pesquisa

# Agenda

- Definições e conceitos
- **Classe Nó em Java** 
- Classe Árvore Binária em Java

# Algoritmo em Java - Classe Nó

```
class No {  
    int elemento;  
    No esq;  
    No dir;  
    No(int elemento) {  
        this(elemento, null, null);  
    }  
    No(int elemento, No esq, No dir) {  
        this.elemento = elemento;  
        this.esq = esq;  
        this.dir = dir;  
    }  
}
```



# Agenda

- Definições e conceitos
- Classe Nó em Java
- **Classe Árvore Binária em Java** ←

# Classe Árvore Binária em Java

```
class ArvoreBinaria {  
    No raiz;  
    ArvoreBinaria() { raiz = null; }  
    void inserir(int x) { }  
    void inserirPai(int x) { }  
    boolean pesquisar(int x) { }  
    void caminharCentral() { }  
    void caminharPre() { }  
    void caminharPos() { }  
    void remover(int x) { }  
}
```

# Classe Árvore Binária em Java

```
class ArvoreBinaria {  
    No raiz;  
    ArvoreBinaria() { raiz = null; }  
    void inserir(int x) {}  
    void inserirPai(int x) {}  
    boolean pesquisar(int x) {}  
    void caminharCentral() {}  
    void caminharPre() {}  
    void caminharPos() {}  
    void remover(int x) {}  
}
```

raiz  
null

