

Aula anterior

TAD Listas:

- Lista Linear
- Lista Simples
- Lista Dupla

Árvore Binária

Prof. Diego Silva Caldeira Rocha

Objetivos

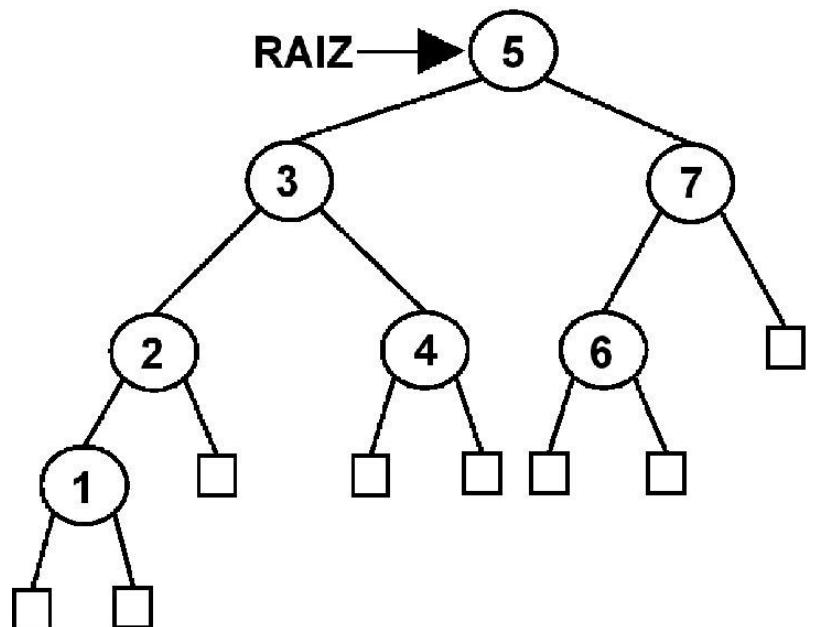
- Introdução a árvore binária
- Inserção
- Pesquisa e caminhamento
- Remoção

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

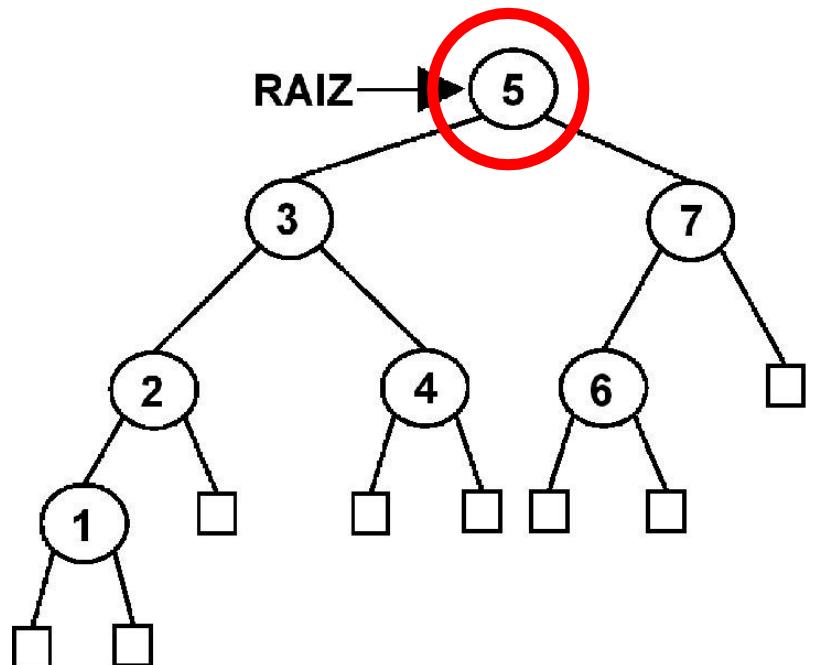
- Custo de inserção, remoção e pesquisa **pode** ser $\Theta(\lg(n))$ comparações
- Formada por um conjunto finito nós (vértices) conectados por arestas



Árvore

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

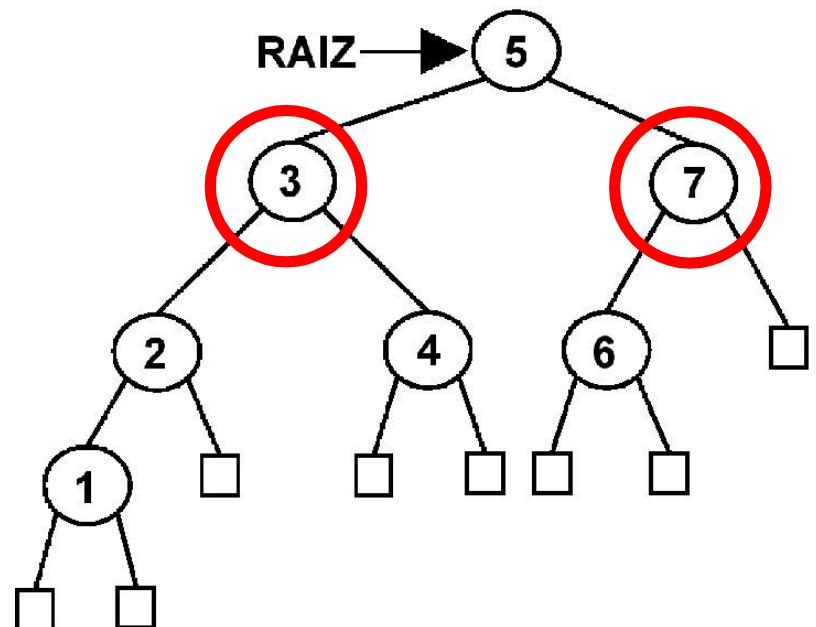
O nó 5 é denominado nó raiz e ele está no nível 0



Árvore

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

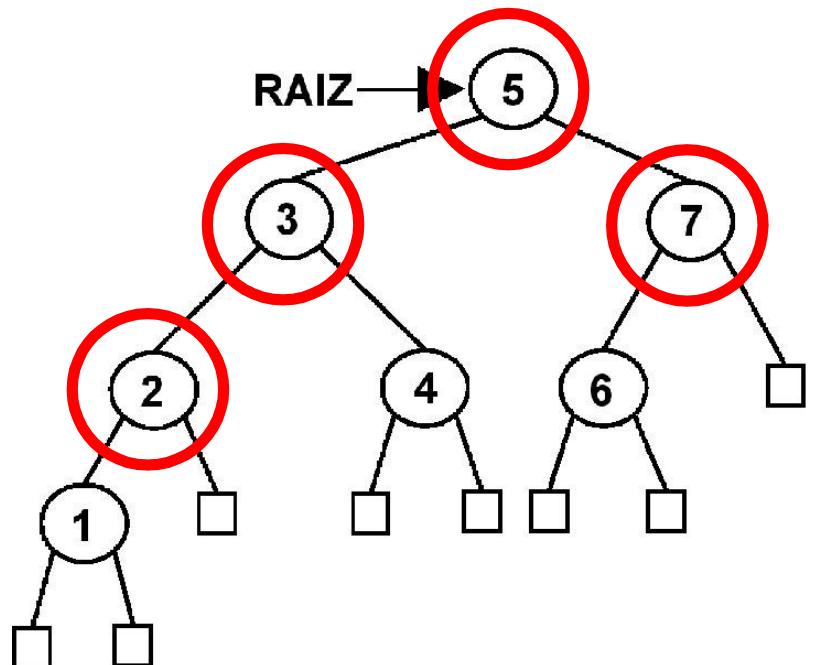
Os nós 3 e 7 são filhos do 5 e
esse é pai dos dois primeiros



Árvore

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

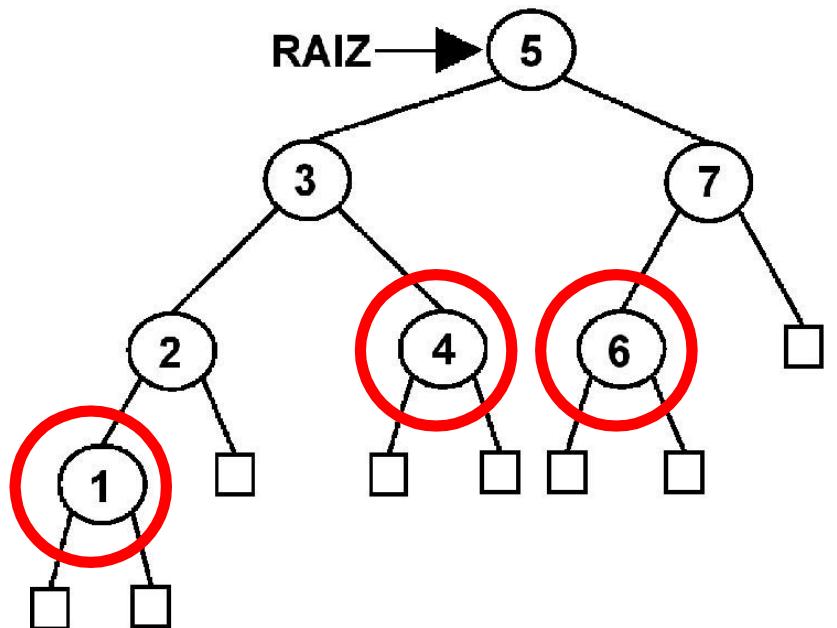
Um nó com filho(s) é chamado de **nó interno** e outro sem, de folha



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



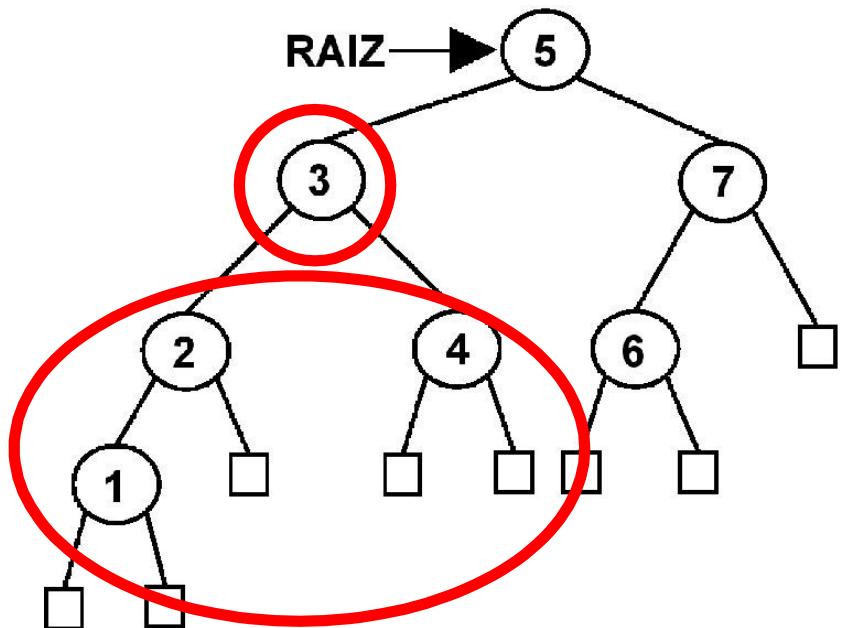
Descobriram Nossa Árvore



Árvore

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

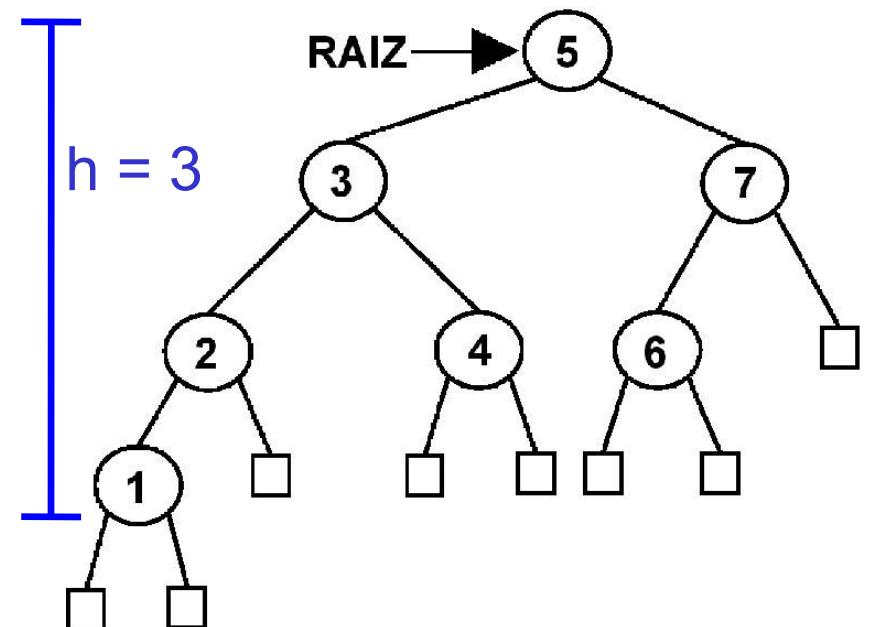
Os nós 1, 2 e 4 formam uma subárvore com raiz no nó 3



Árvore

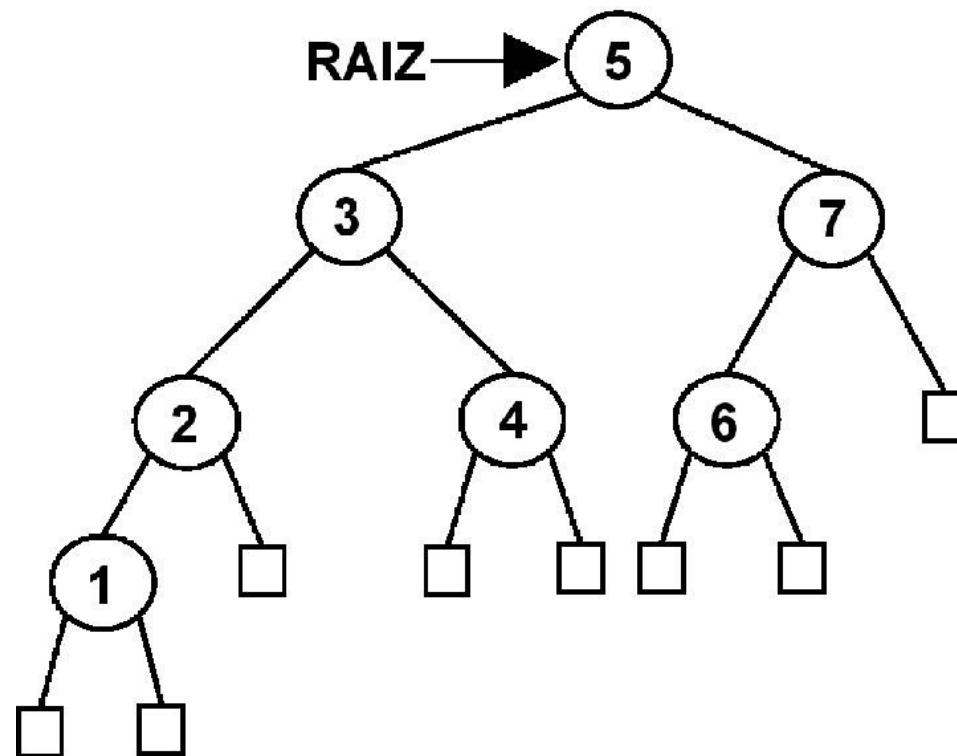
- Custo de inserção, remoção e pesquisa pode ser $\Theta(\lg(n))$ comparações
- Formada por um conjunto finito nós (vértices) conectados por arestas

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



Árvore Binária

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

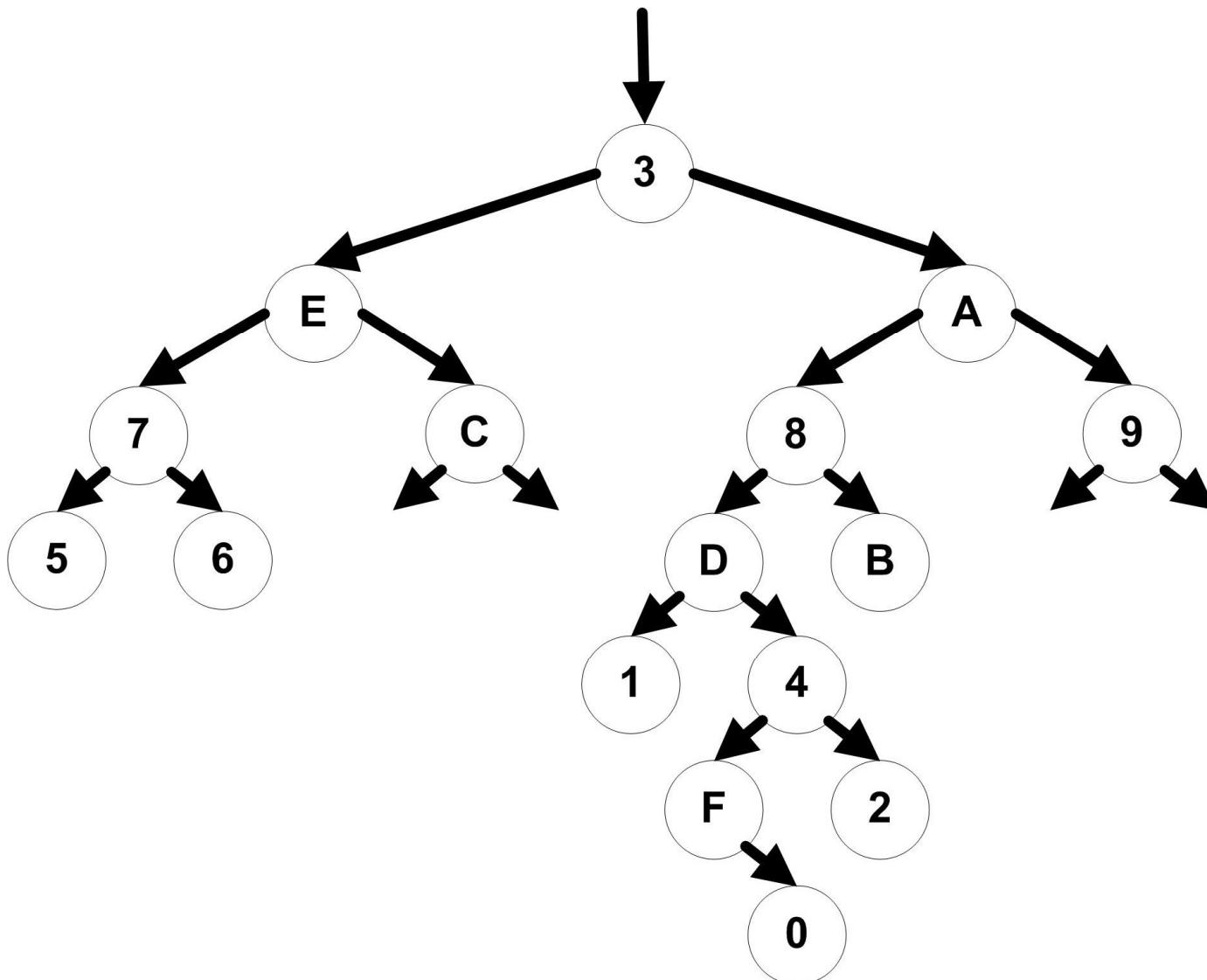


Exercício Resolvido (2)

- Crie uma árvore binária com os dígitos hexadecimais

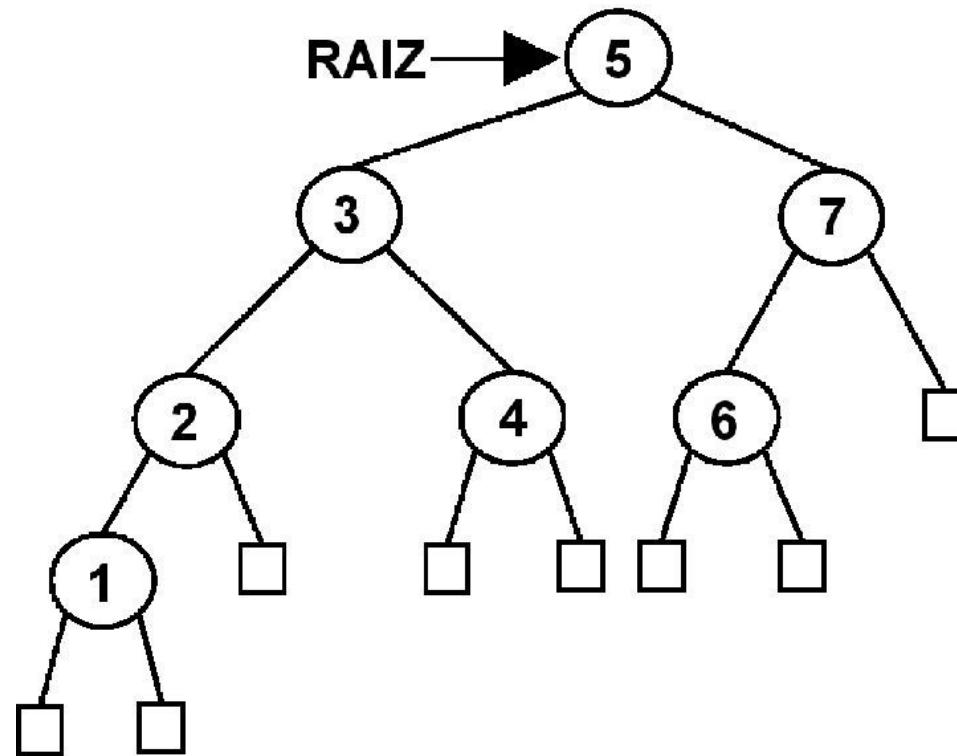
Exercício Resolvido (2)

- 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:



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

- 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)



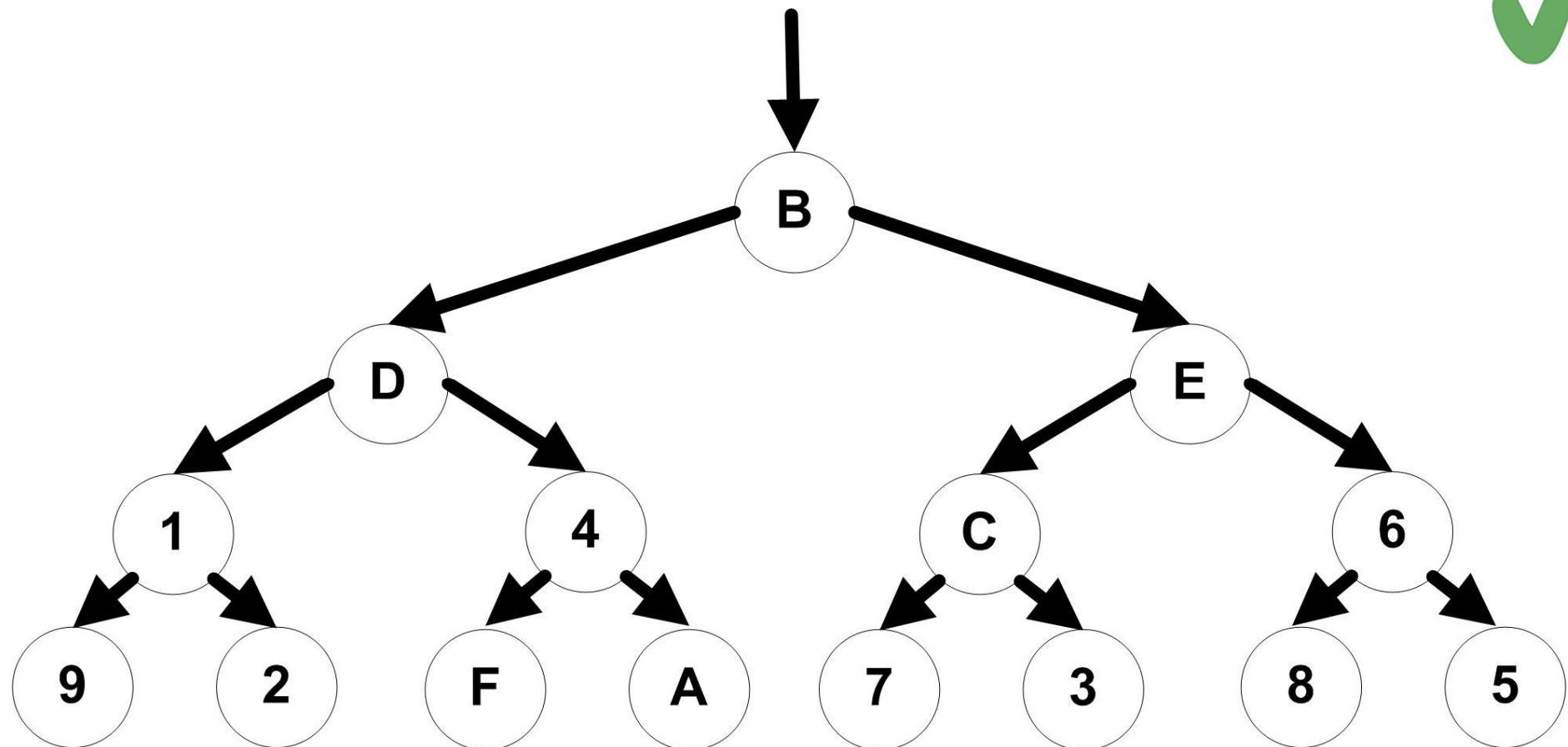
Impossível, pois o número de nós
é uma potência de dois
(está sobrando um nó)

Exercício Resolvido

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

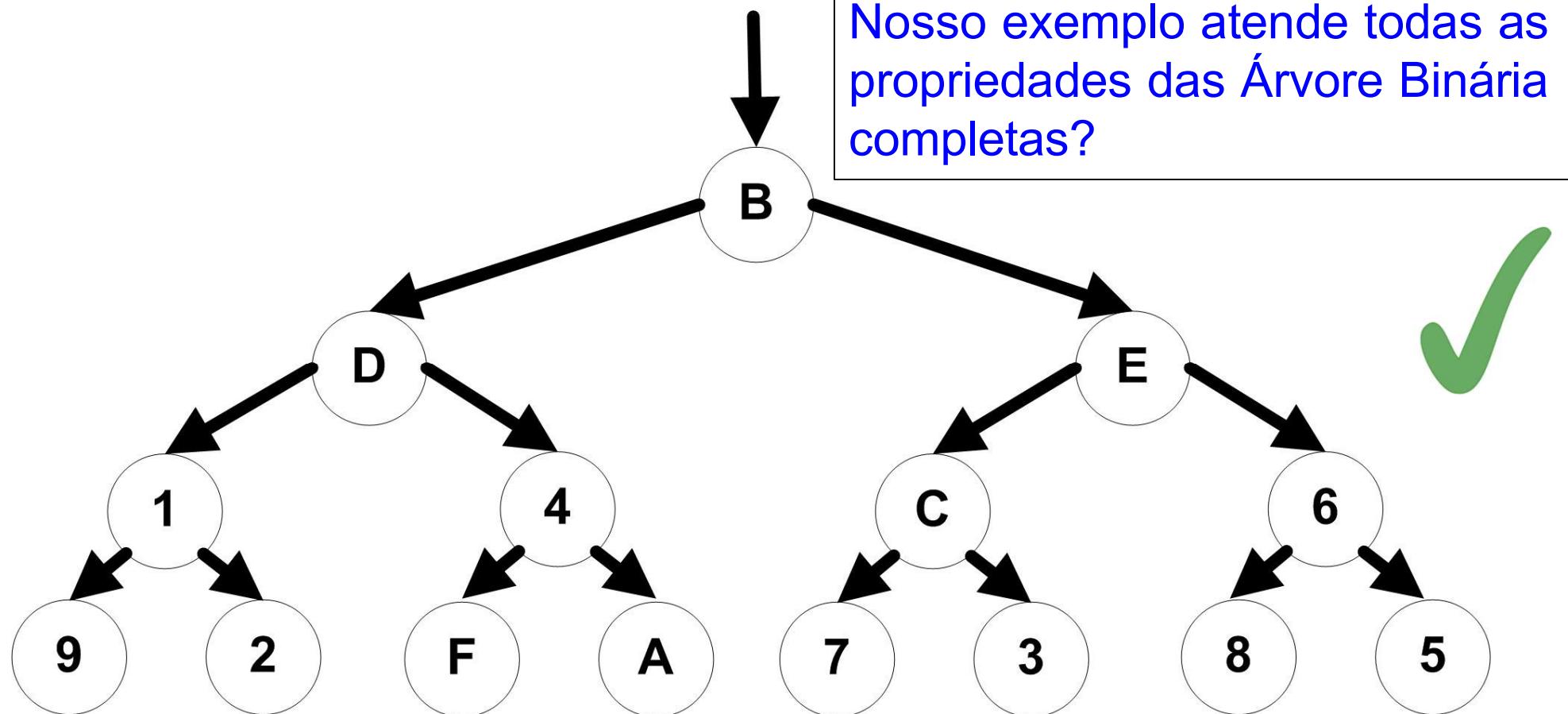
Exercício Resolvido

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



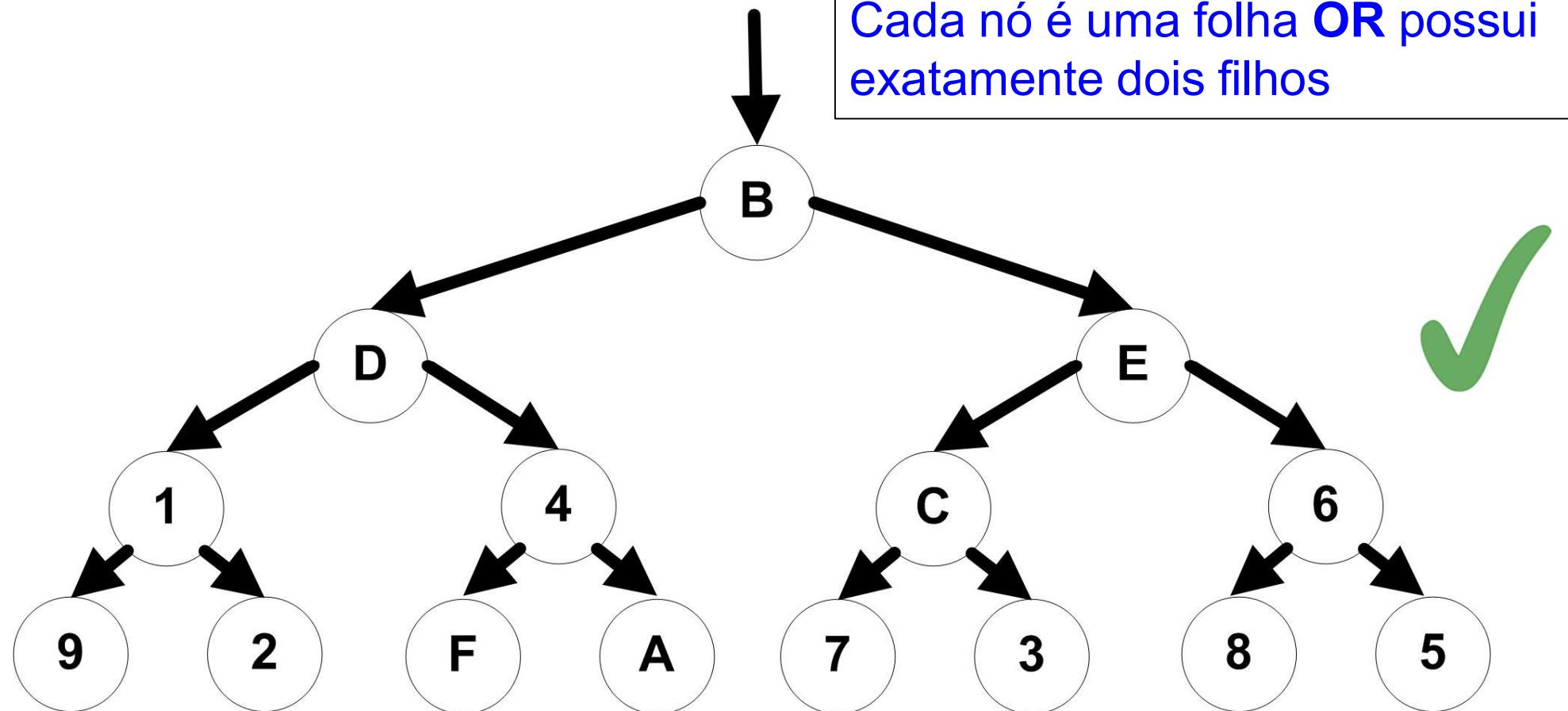
Exercício Resolvido

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



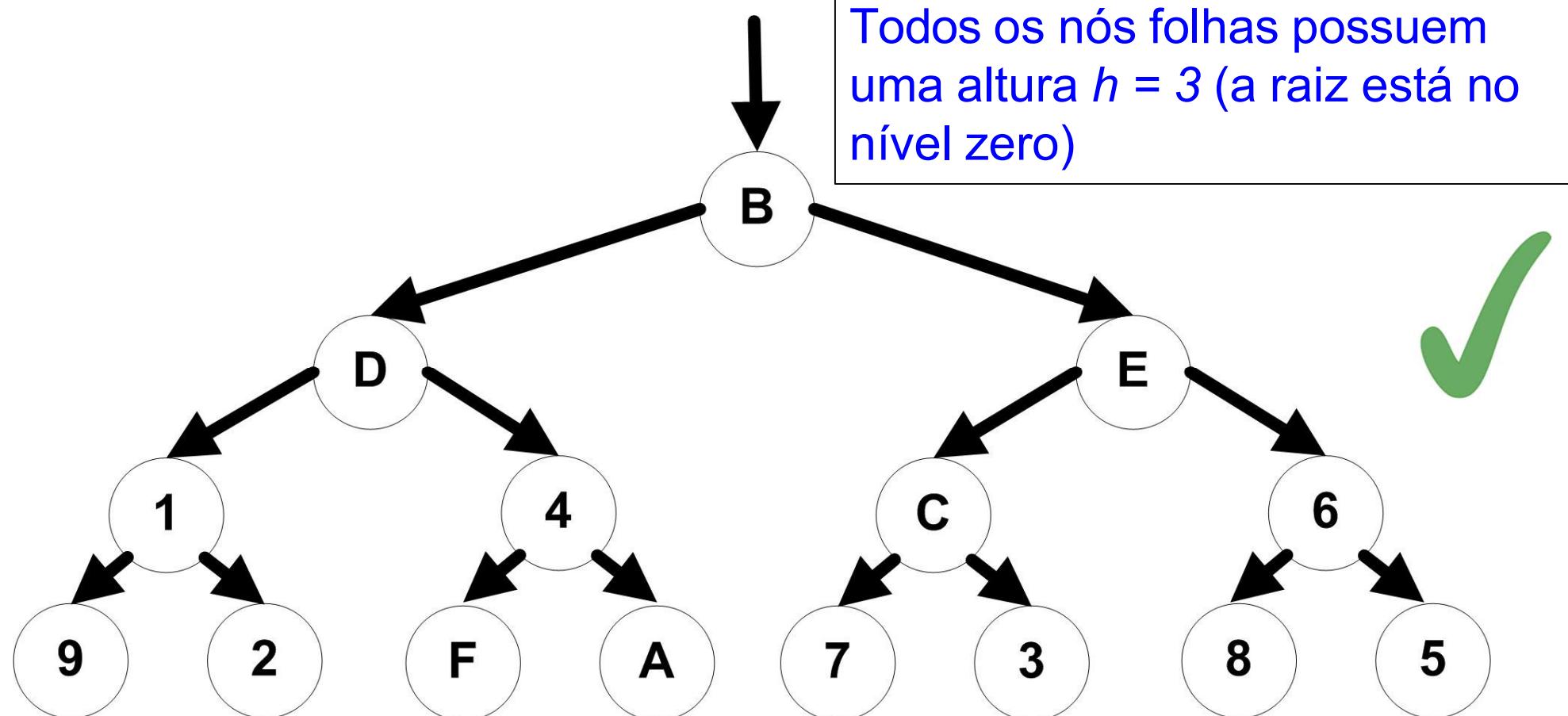
Exercício Resolvido (5)

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



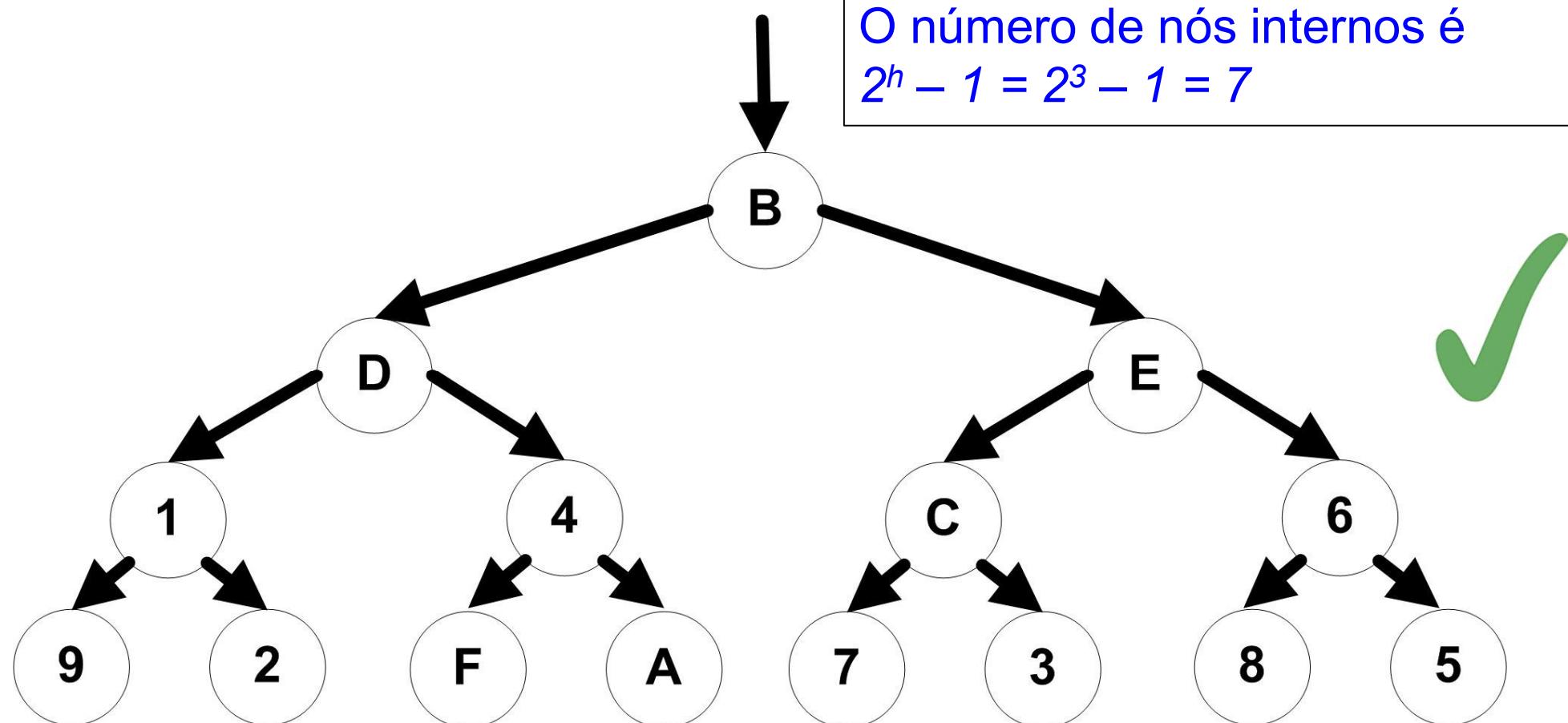
Exercício Resolvido (5)

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



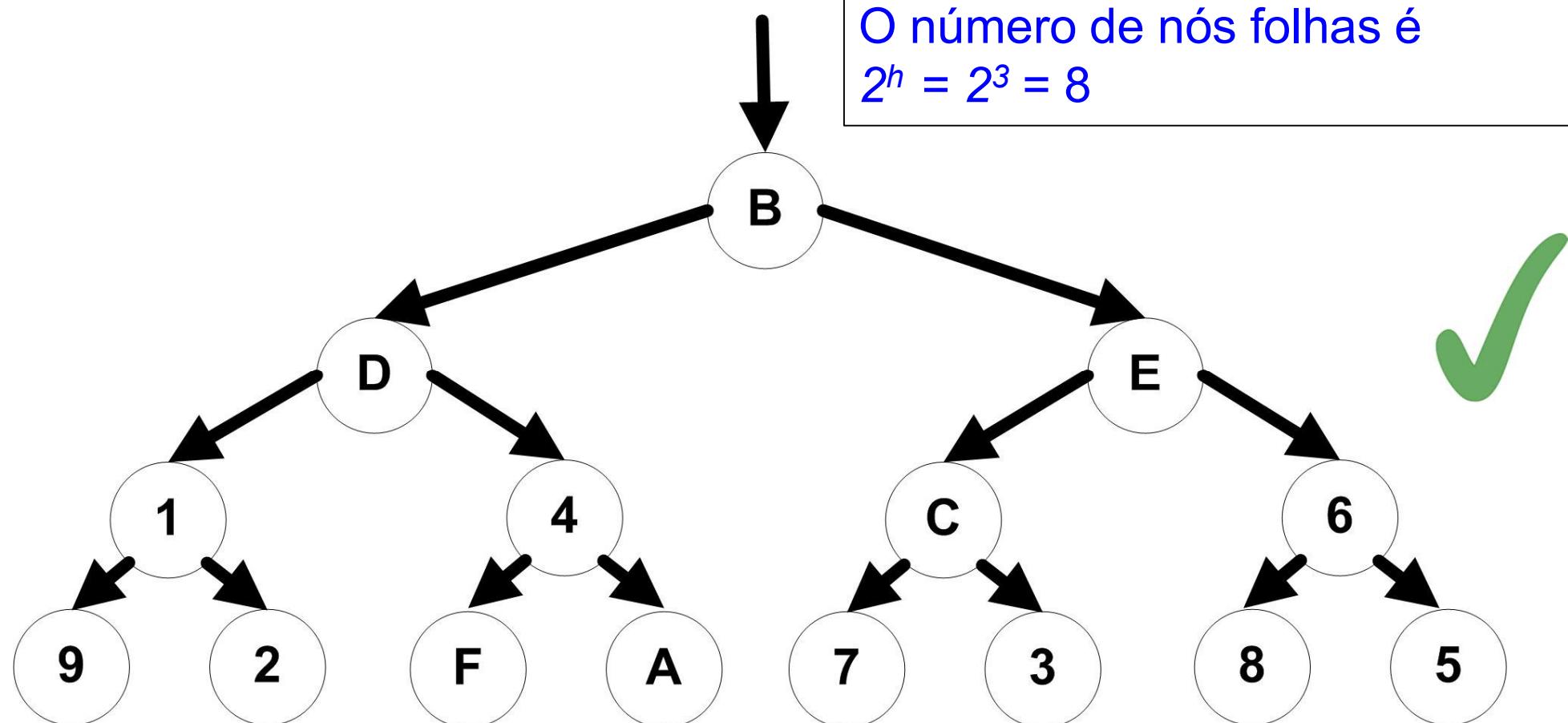
Exercício Resolvido (5)

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



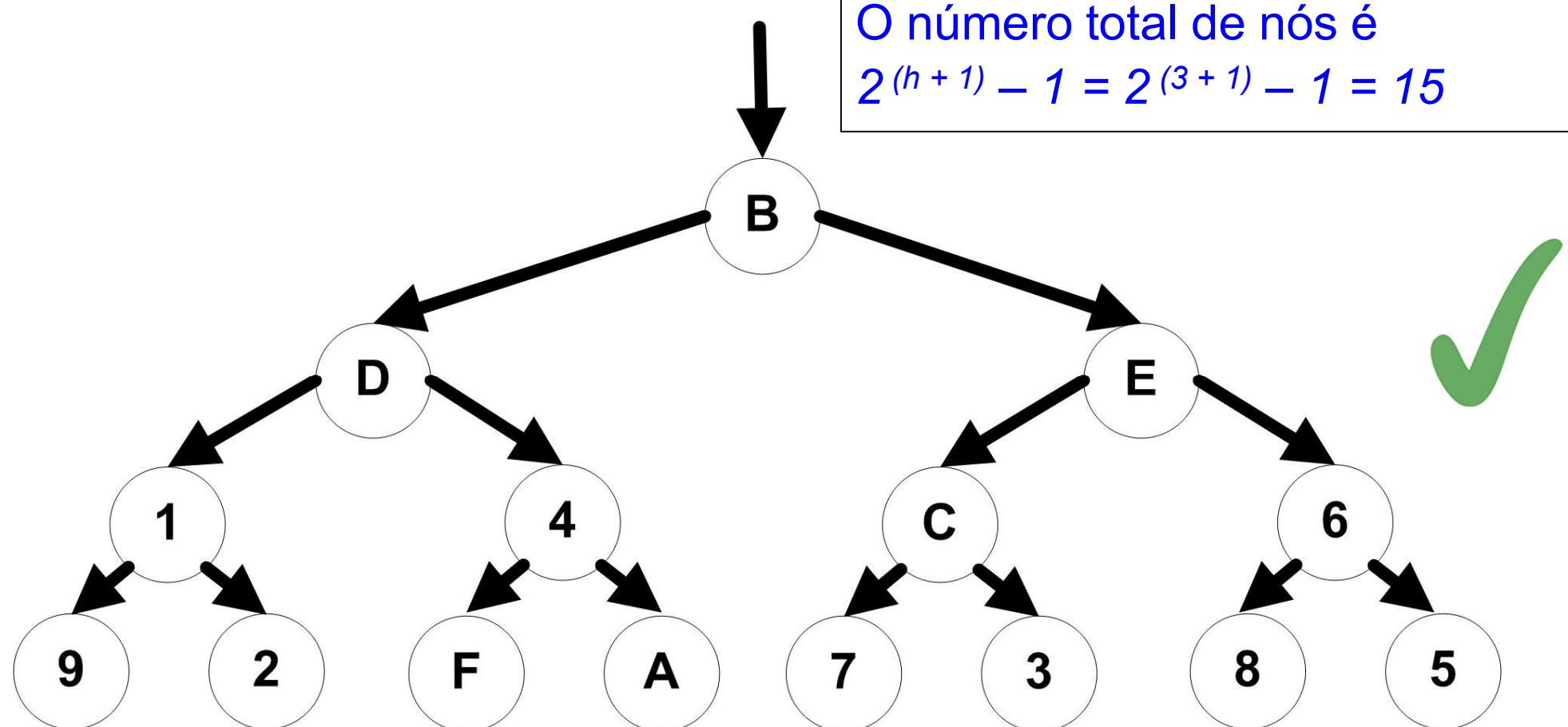
Exercício Resolvido (5)

- 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

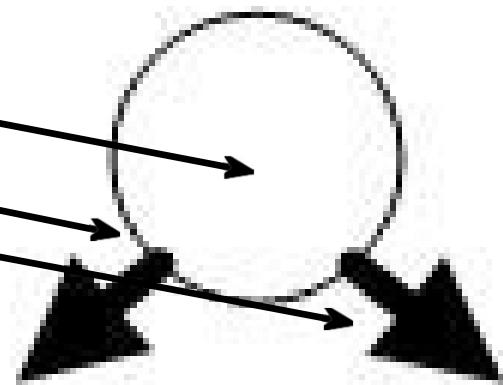


Consideração

- 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

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;  
    }  
}
```



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



Funcionamento Básico da Inserção

- (1) Se a raiz estiver vazia, insere-se o elemento nela
- (2) Senão, se o novo elemento for **menor** que o da raiz, chama-se recursivamente a inserção para a subárvore da **esquerda**
- (3) Senão, se o novo elemento for **maior** que o da raiz, chama-se recursivamente a inserção para a subárvore da **direita**
- (4) Senão, se o novo elemento for **igual** ao da raiz, não inserir um elemento repetido

Exemplo

- Inserir, na ordem, os elementos 3, 5, 1, 8, 2, 4, 7 e 6

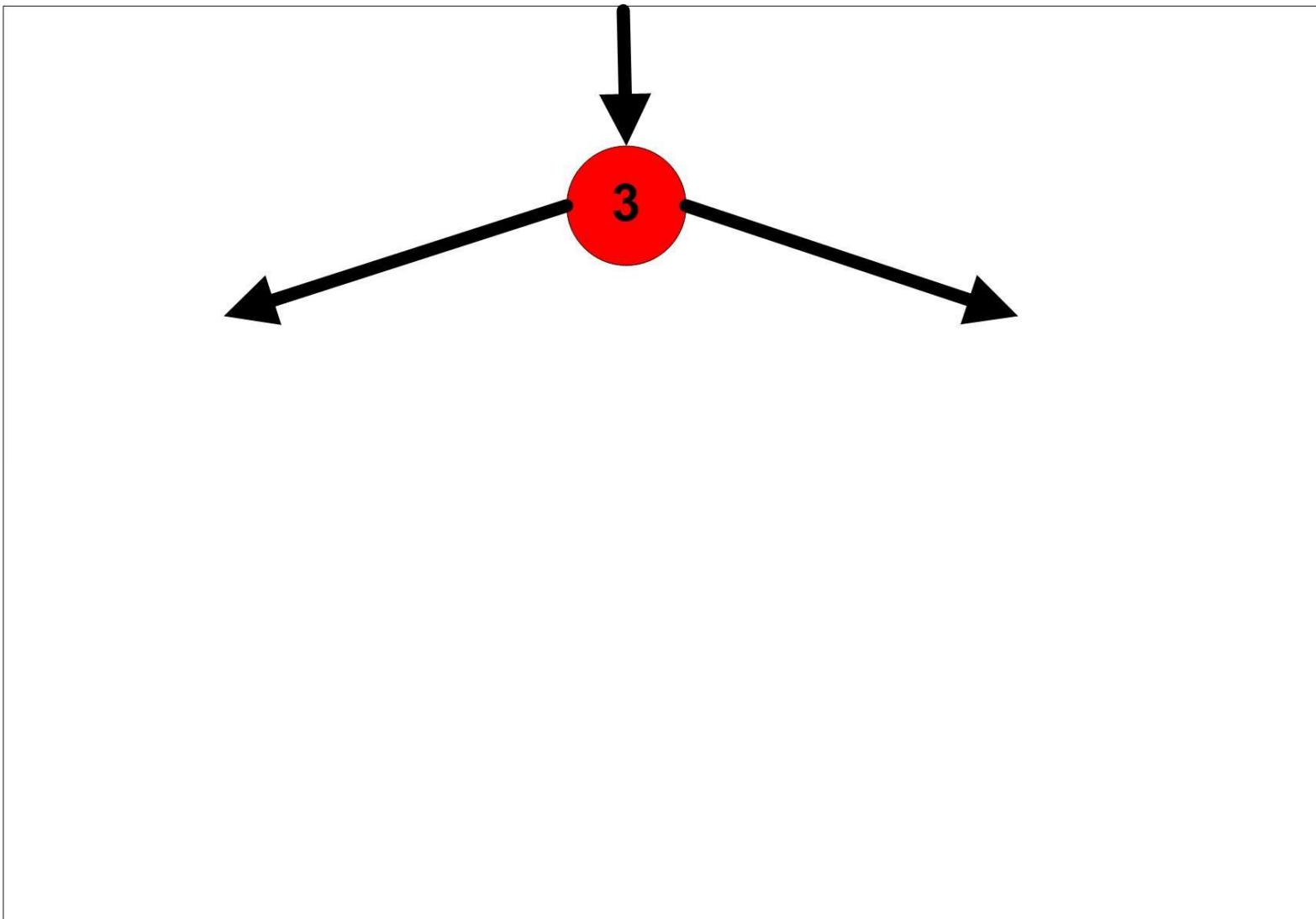
Exemplo

- Inserir, na ordem, os elementos 3, 5, 1, 8, 2, 4, 7 e 6



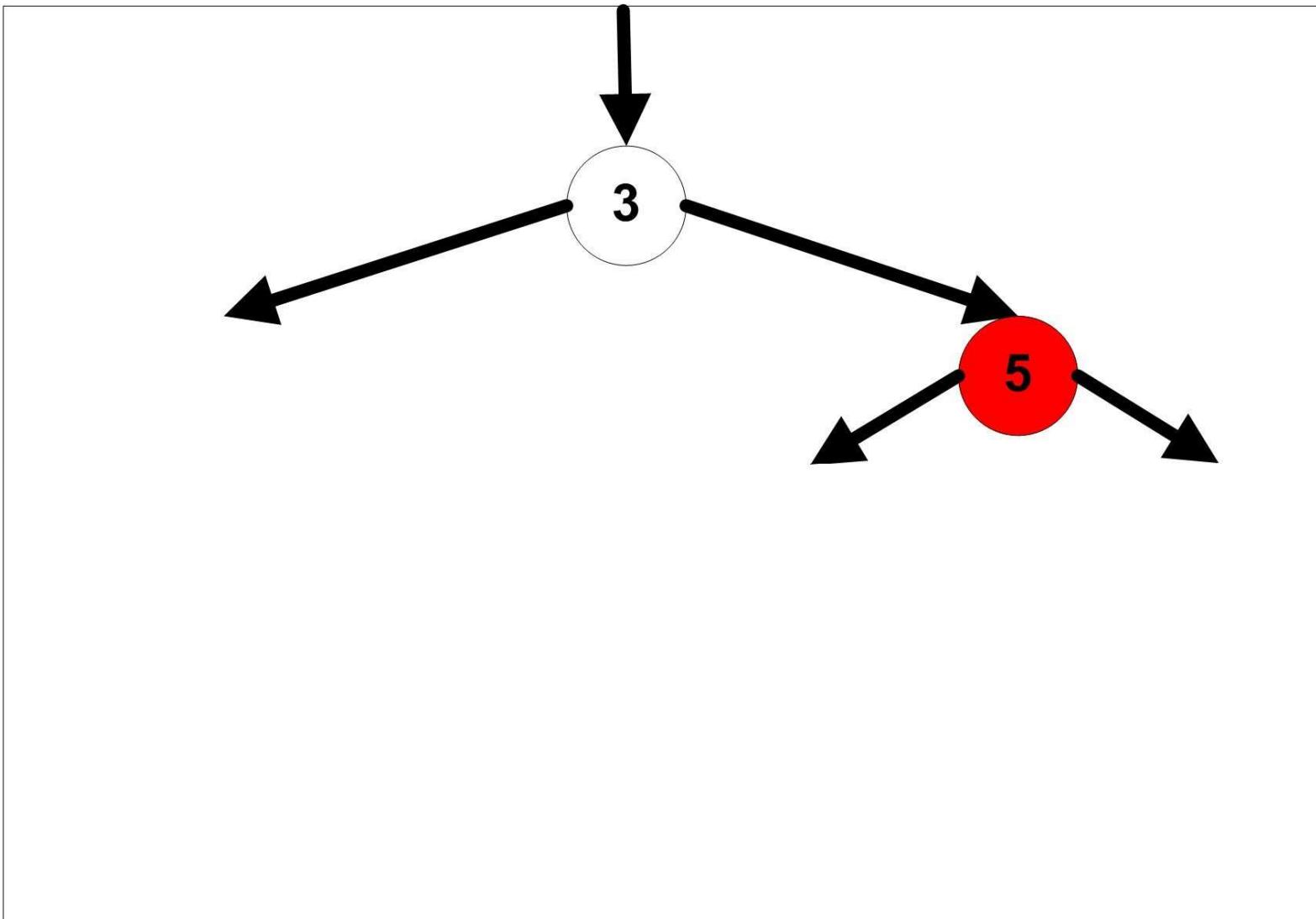
Exemplo

- Inserir, na ordem, os elementos 3, 5, 1, 8, 2, 4, 7 e 6



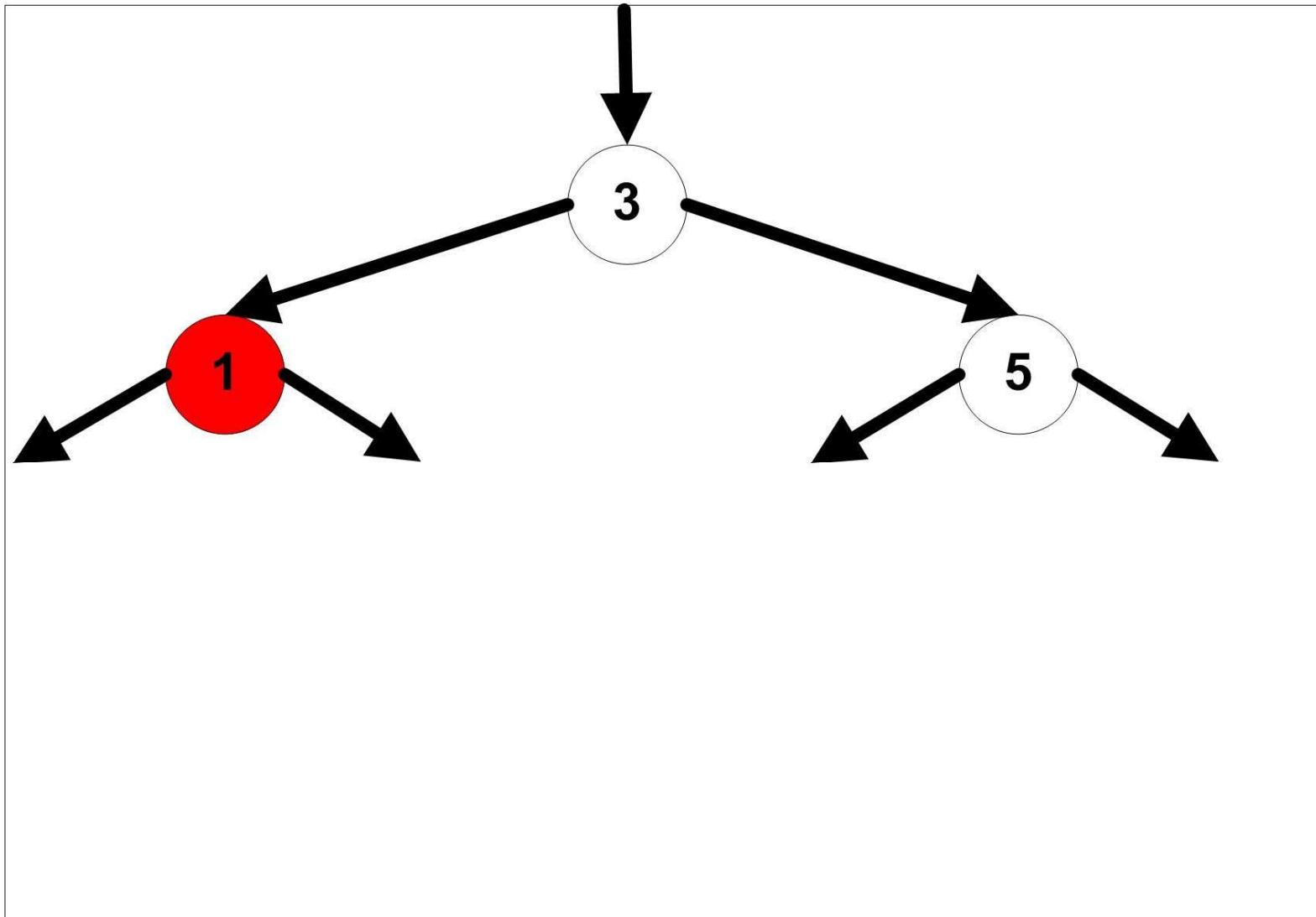
Exemplo

- Inserir, na ordem, os elementos 3, **5**, 1, 8, 2, 4, 7 e 6



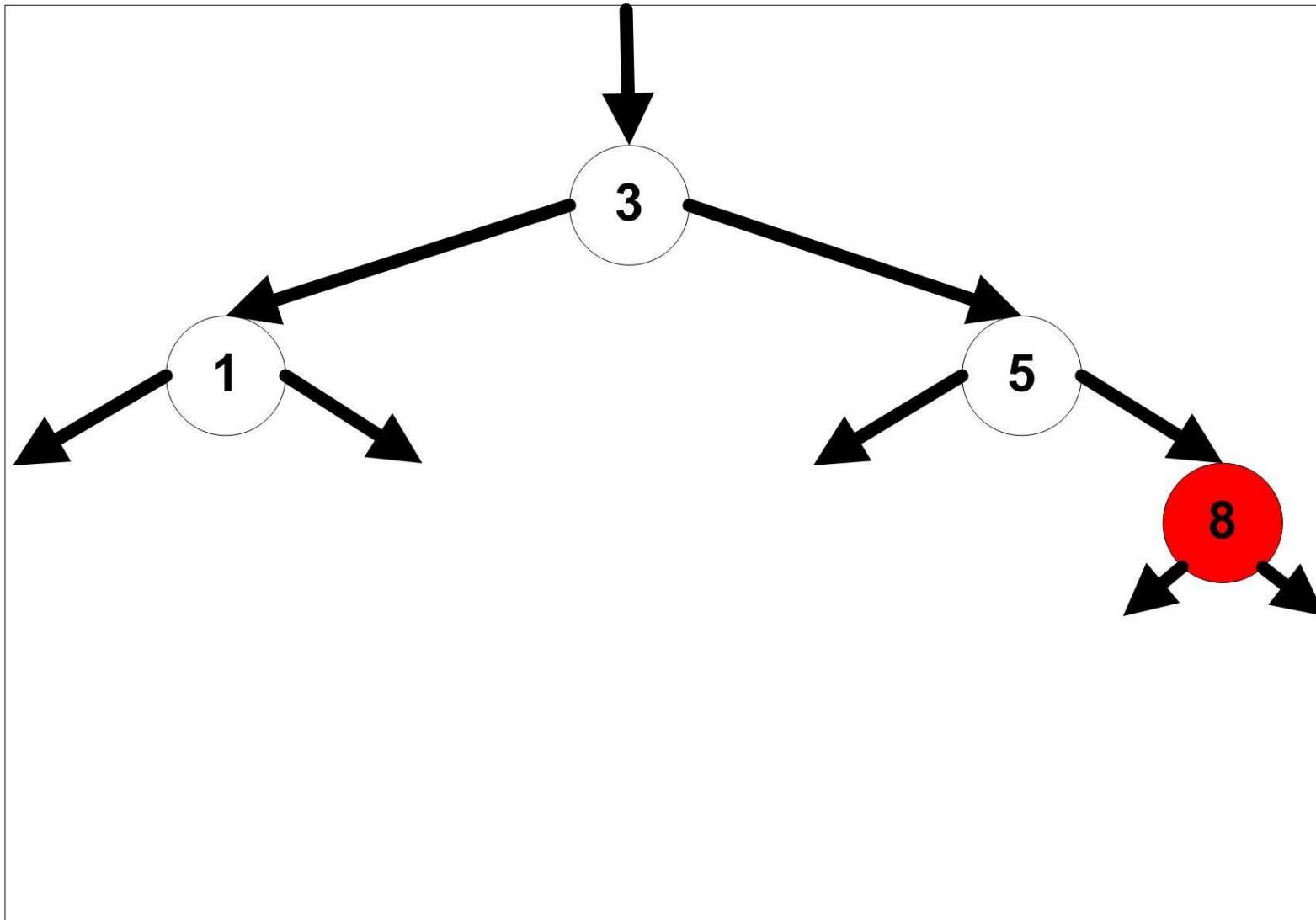
Exemplo

- Inserir, na ordem, os elementos 3, 5, **1**, 8, 2, 4, 7 e 6



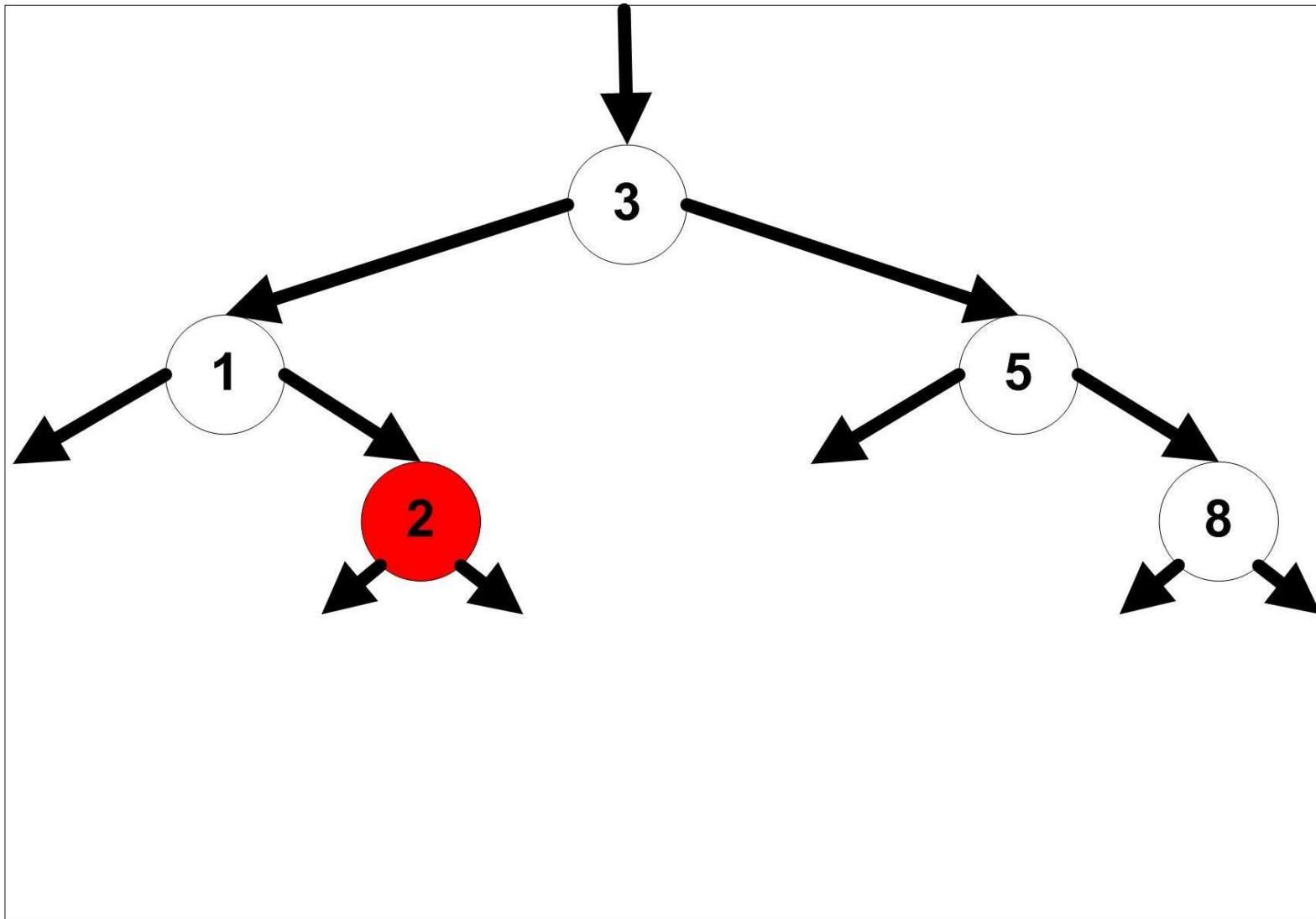
Exemplo

- Inserir, na ordem, os elementos 3, 5, 1, **8**, 2, 4, 7 e 6



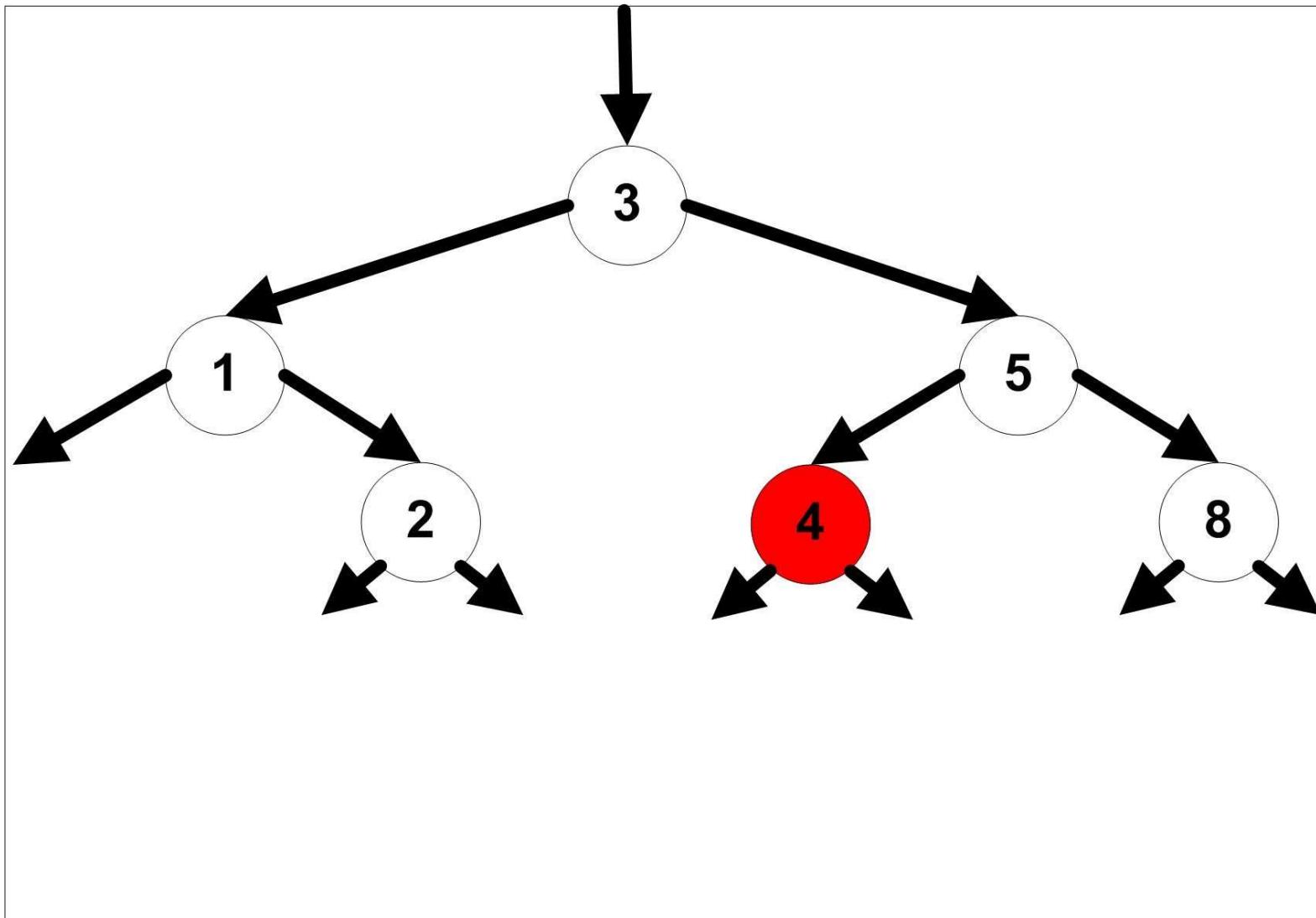
Exemplo

- Inserir, na ordem, os elementos 3, 5, 1, 8, **2**, 4, 7 e 6



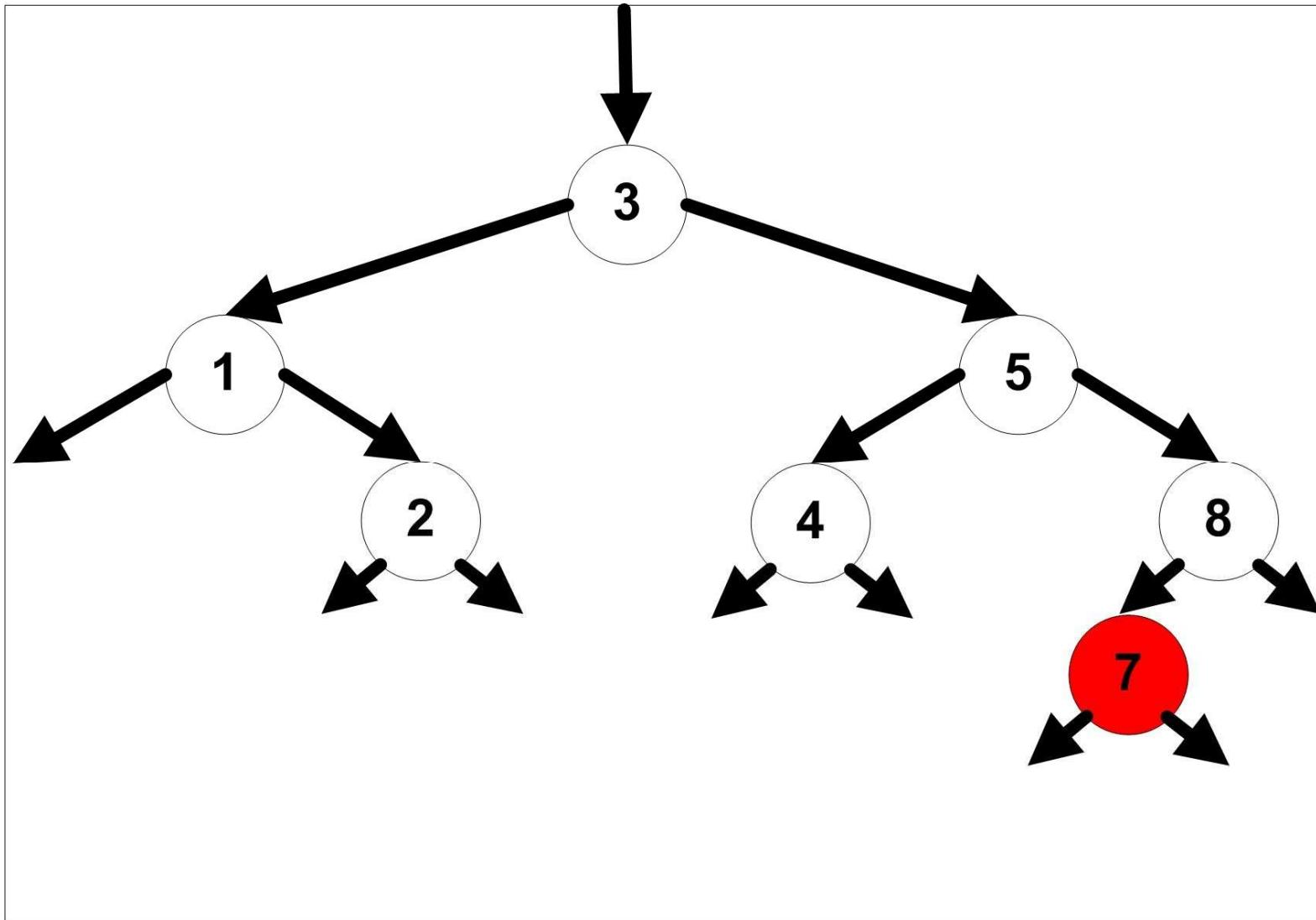
Exemplo

- Inserir, na ordem, os elementos 3, 5, 1, 8, 2, **4**, 7 e 6



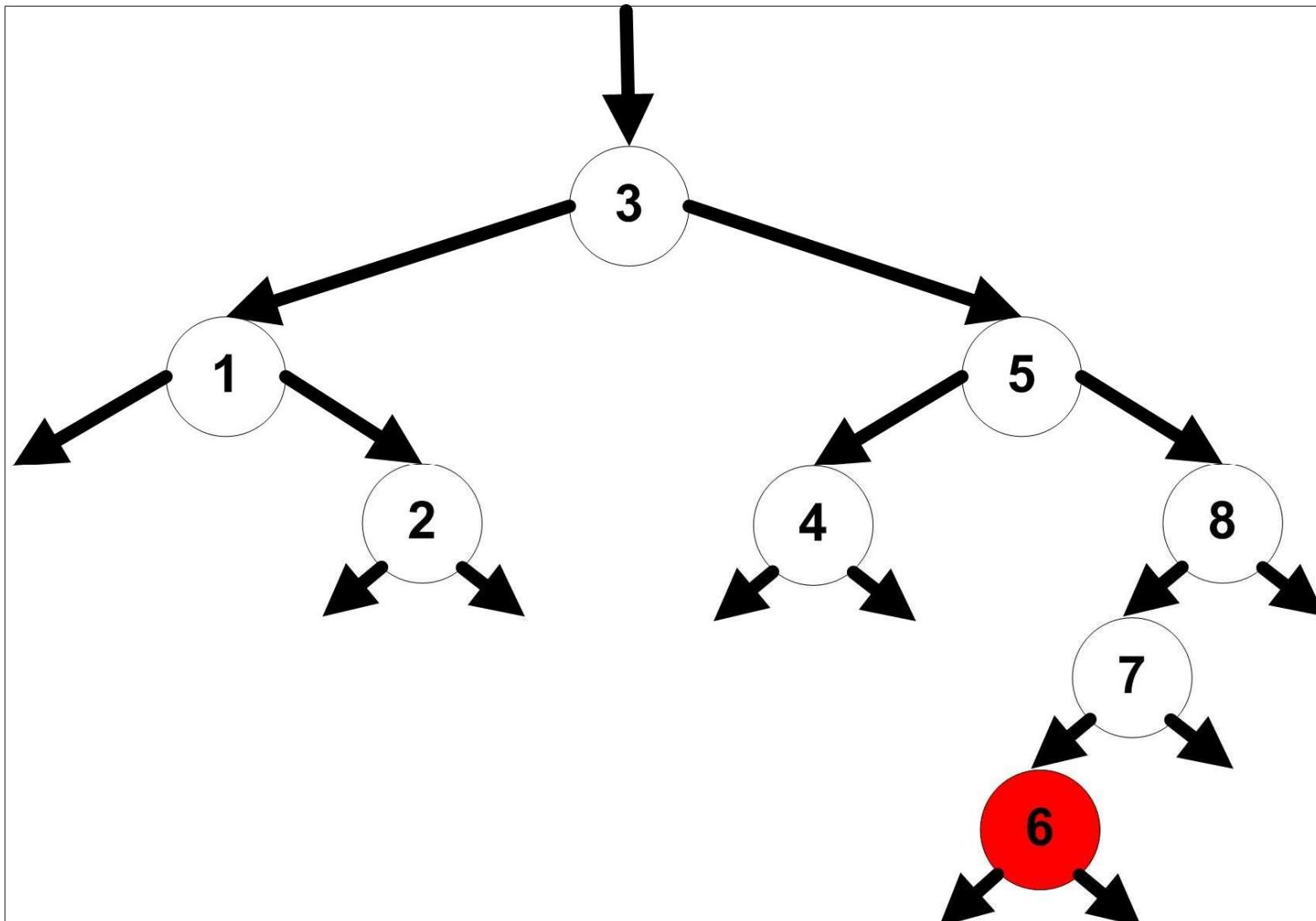
Exemplo

- Inserir, na ordem, os elementos 3, 5, 1, 8, 2, 4, **7** e 6



Exemplo

- Inserir, na ordem, os elementos 3, 5, 1, 8, 2, 4, 7 e **6**



Exercício

- Inserir, na ordem, os elementos 1, 2, 3, 4, 5, 6, 7 e 8

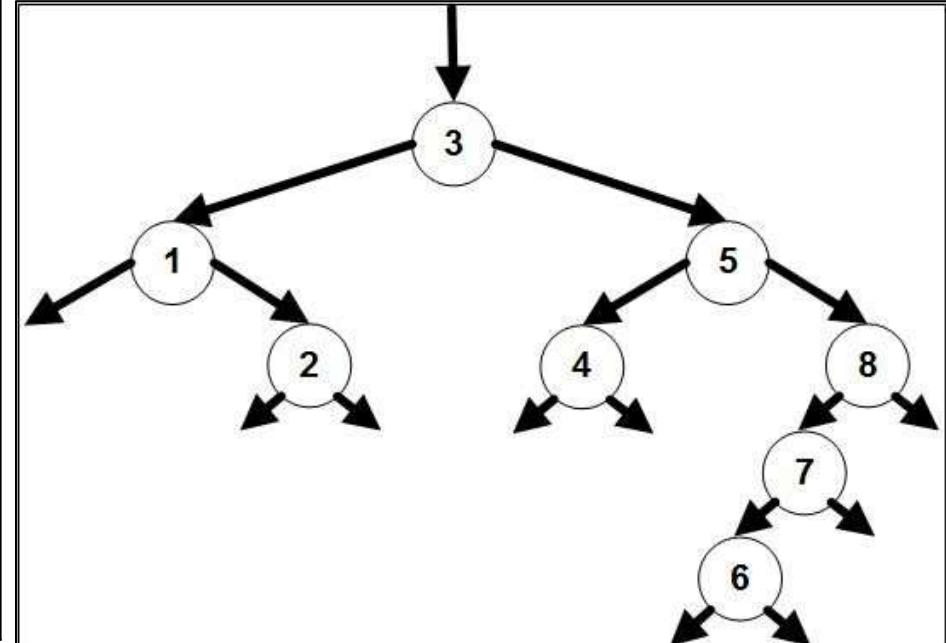
Classe Árvore Binária em Java

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

Inserção em Java Interativa

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

```
void inserirIterativo(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
        return;  
    }  
  
    No atual = raiz;  
    No anterior = null;  
    while (atual != null) {  
        anterior = atual;  
        if (x < atual.elemento) {  
            atual = atual.esq;  
        } else if (x > atual.elemento) {  
            atual = atual.dir;  
        }  
        // Faz a ligação com o nó pai.  
        if (x < anterior.elemento) {  
            anterior.esq = new No(x);  
        } else {  
            anterior.dir = new No(x);  
        }  
    }  
}
```



Inserção em Java com Retorno de Referência

```
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



Inserção em Java com Retorno de Referência

```
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



Vamos inserir os elementos
3, 5, 1, 8, 2, 4, 7 e 6
(várias chamadas do inserir)

Inserção em Java com Retorno de Referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

```
void inserir(int x) throws Exception {  
    raiz = inserir(x, raiz);  
}
```

```
No inserir(int x, No i) throws Exception {  
    if (i == null) {  
        i = new No(x);  
    } else if (x < i.elemento) {  
        i.esq = inserir(x, i.esq);  
    } else if (x > i.elemento) {  
        i.dir = inserir(x, i.dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz null x



Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {  
    raiz = inserir(x, raiz);  
}
```

```
No inserir(int x, No i) throws Exception {  
    if (i == null) {  
        i = new No(x);  
    } else if (x < i.elemento) {  
        i.esq = inserir(x, i.esq);  
    } else if (x > i.elemento) {  
        i.dir = inserir(x, i.dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz  x

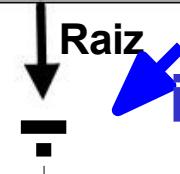
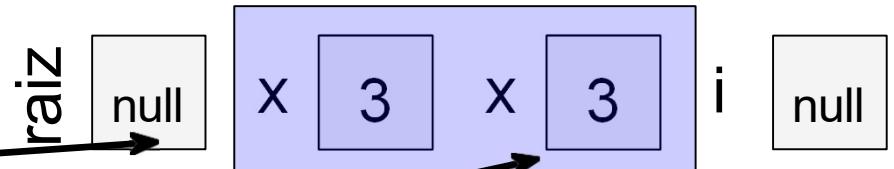


Inserção em Java com Retorno de Referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else
        throw ...
    }
    return i;
}
```



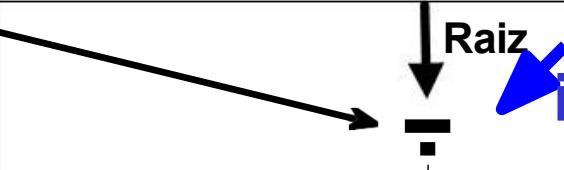
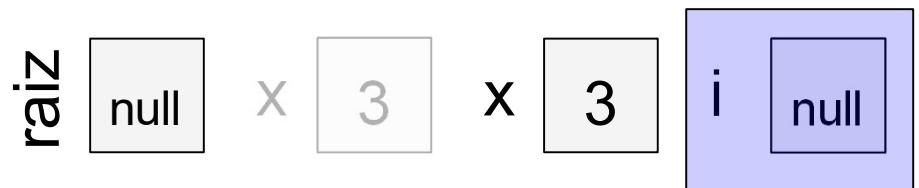
Cada chamada do inserir cria novas variáveis
e, por isso, temos duas variáveis com nome x

Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else
        throw ...
    }
    return i;
}
```



O valor inicial do ponteiro *i* é o mesmo
do ponteiro *raiz*, ou seja, *null*

Inserção em Java com Retorno de Referência

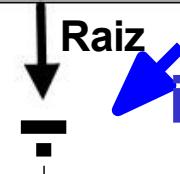
```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
```

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
}
```

true: null == null



Inserção em Java com Retorno de Referência

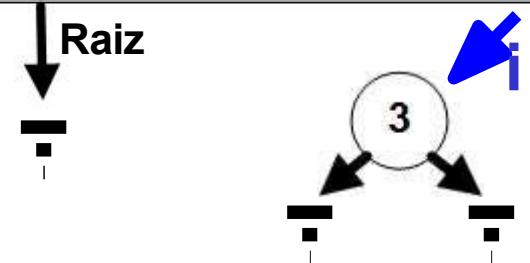
```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
```

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
}
```

raiz	null	x	3	x	3	i	n(3)
------	------	---	---	---	---	---	------



Inserção em Java com Retorno de Referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

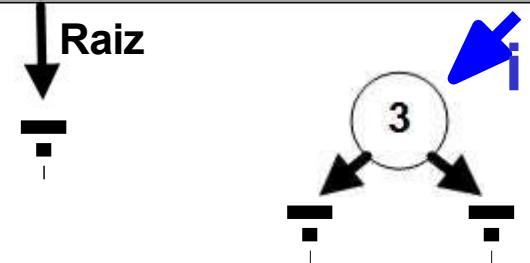
```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

No inserir(int x, No i) throws Exception {

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
```

} retorna o endereço de n(3)

raiz null x 3 x 3 i n(3)



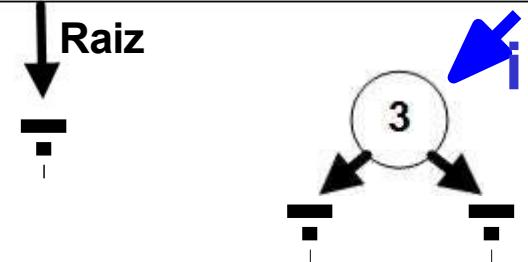
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz null x 3 x 3 i n(3)



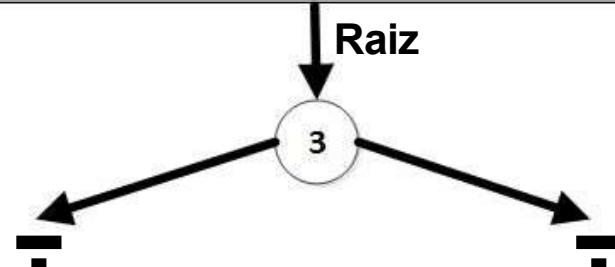
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {  
    raiz = inserir(x, raiz);  
}
```

```
No inserir(int x, No i) throws Exception {  
    if (i == null) {  
        i = new No(x);  
    } else if (x < i.elemento) {  
        i.esq = inserir(x, i.esq);  
    } else if (x > i.elemento) {  
        i.dir = inserir(x, i.dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz n(3) x



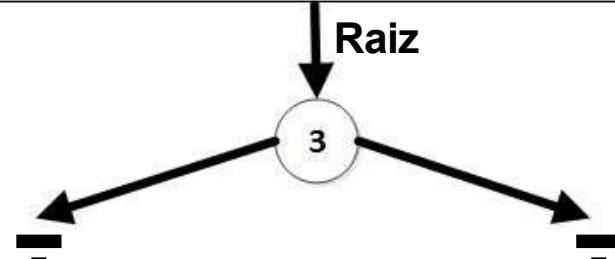
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {  
    raiz = inserir(x, raiz);  
}
```

```
No inserir(int x, No i) throws Exception {  
    if (i == null) {  
        i = new No(x);  
    } else if (x < i.elemento) {  
        i.esq = inserir(x, i.esq);  
    } else if (x > i.elemento) {  
        i.dir = inserir(x, i.dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz



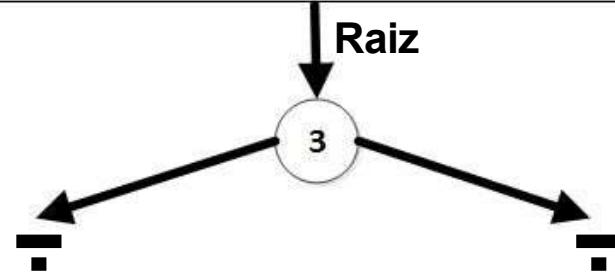
Inserção em Java com Retorno de Referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

```
void inserir(int x) throws Exception {  
    raiz = inserir(x, raiz);  
}
```

```
No inserir(int x, No i) throws Exception {  
    if (i == null) {  
        i = new No(x);  
    } else if (x < i.elemento) {  
        i.esq = inserir(x, i.esq);  
    } else if (x > i.elemento) {  
        i.dir = inserir(x, i.dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz n(3) x



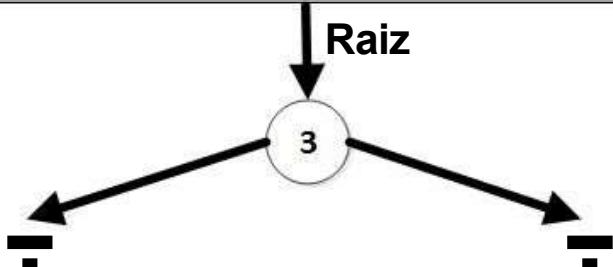
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {  
    raiz = inserir(x, raiz);  
}
```

```
No inserir(int x, No i) throws Exception {  
    if (i == null) {  
        i = new No(x);  
    } else if (x < i.elemento) {  
        i.esq = inserir(x, i.esq);  
    } else if (x > i.elemento) {  
        i.dir = inserir(x, i.dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz n(3) x



Inserção em Java com Retorno de Referência

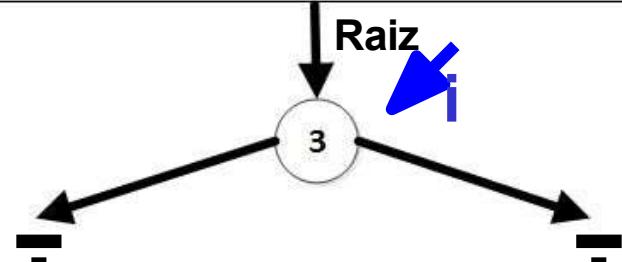
```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
```

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
}
```

raiz n(3) x 5 x 5 i n(3)



Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

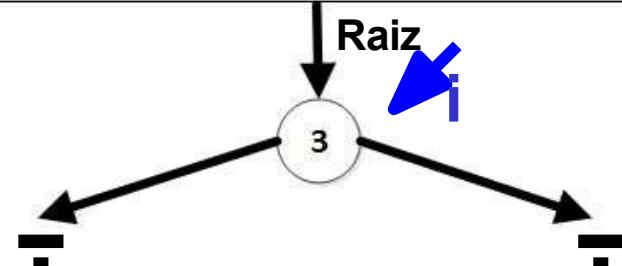
```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
```

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
}
```

false: n(3) == null

raiz	n(3)	x	5	x	5	i	n(3)
------	------	---	---	---	---	---	------



Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
```

```
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
```

```
        i.esq = inserir(x, i.esq);
```

```
    } else if (x > i.elemento) {
```

```
        i.dir = inserir(x, i.dir);
```

```
    } else {
```

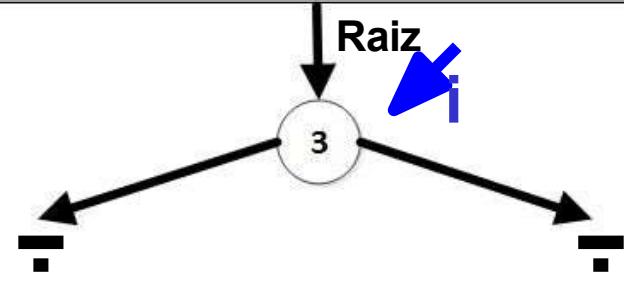
```
        throw new Exception("Erro!");
```

```
}
```

```
return i;    false: 5 < 3
```

```
}
```

raiz n(3) x 5 x 5 i n(3)



Inserção em Java com Retorno de Referência

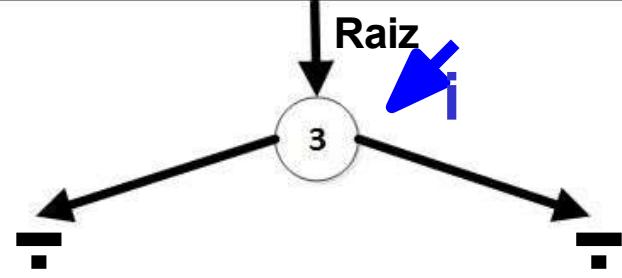
```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

true: 5 > 3

raiz	n(3)	x	5	x	5	i	n(3)
------	------	---	---	---	---	---	------



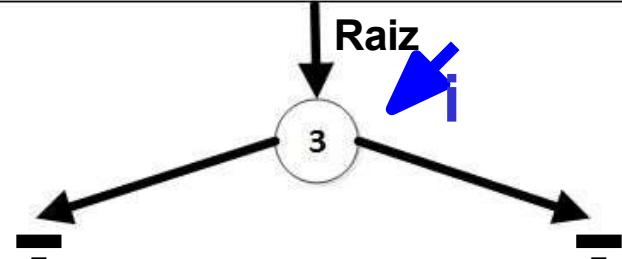
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 5 x 5 i n(3)



Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

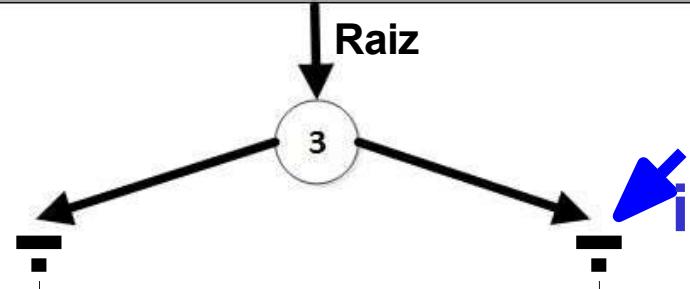
```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
```

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
}
```

raiz n(3) x 5 x 5 i n(3)

x 5 i null



Inserção em Java com Retorno de Referência

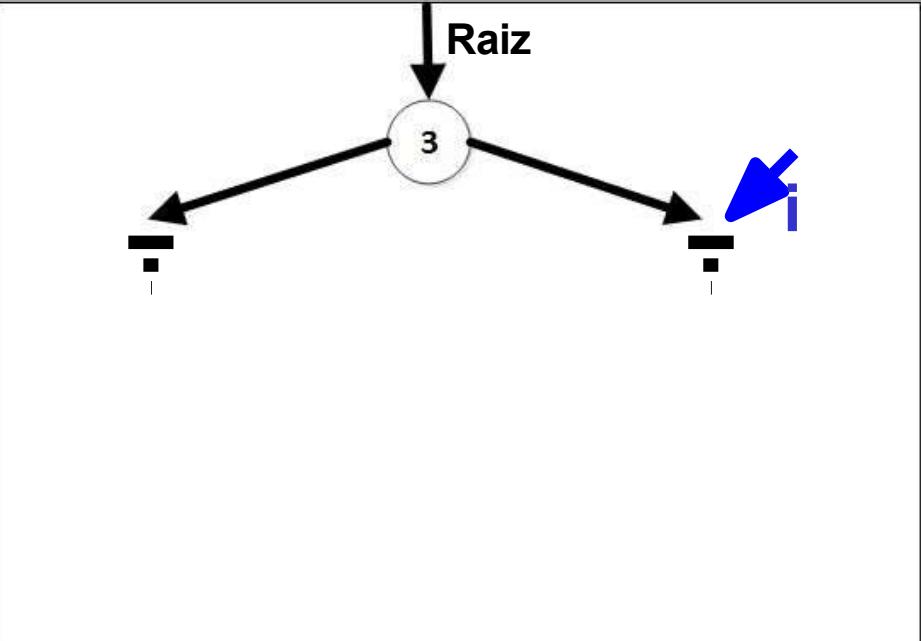
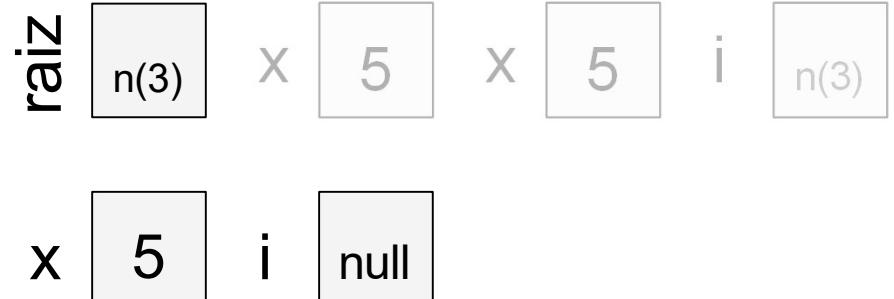
```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
```

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
}
```

true: null == null



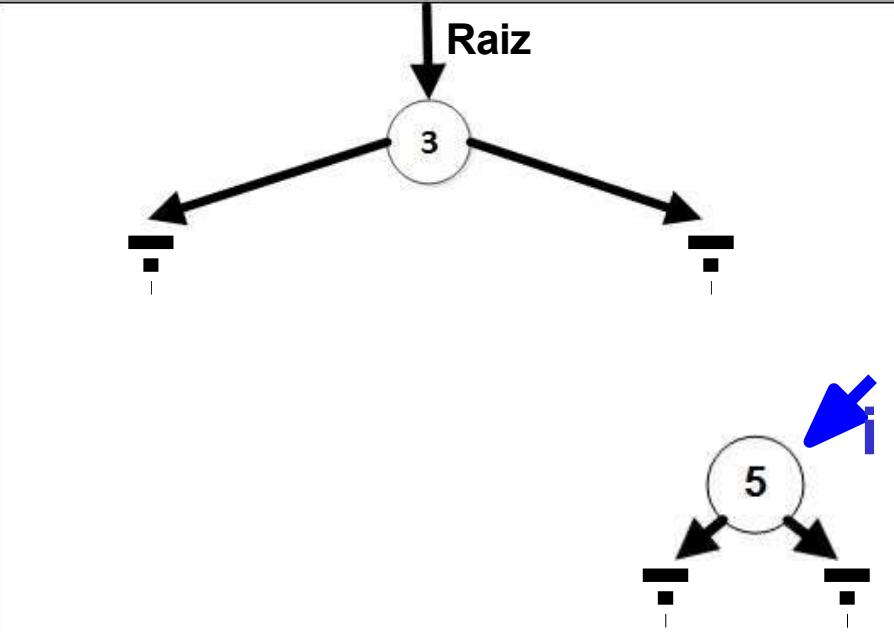
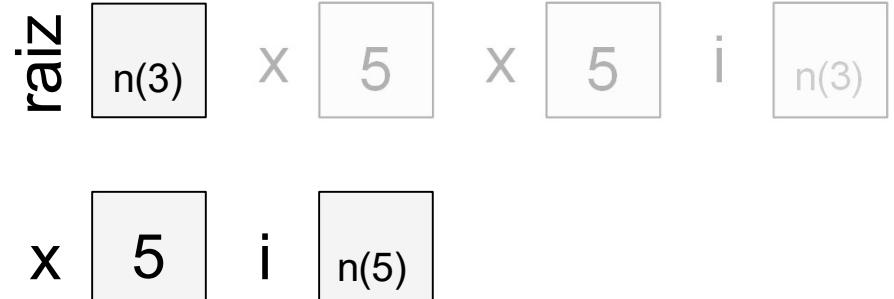
Inserção em Java com Retorno de Referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

No inserir(int x, No i) throws Exception {

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
}
```



Inserção em Java com Retorno de Referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

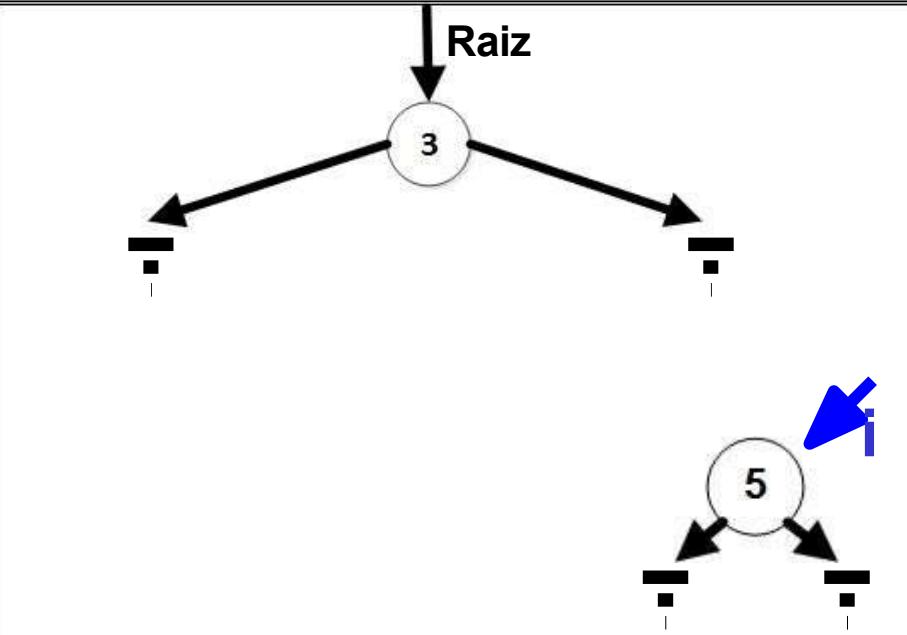
```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

No inserir(int x, No i) throws Exception {

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
```

return i;

retorna o endereço de n(5)



Inserção em Java com Retorno de Referência

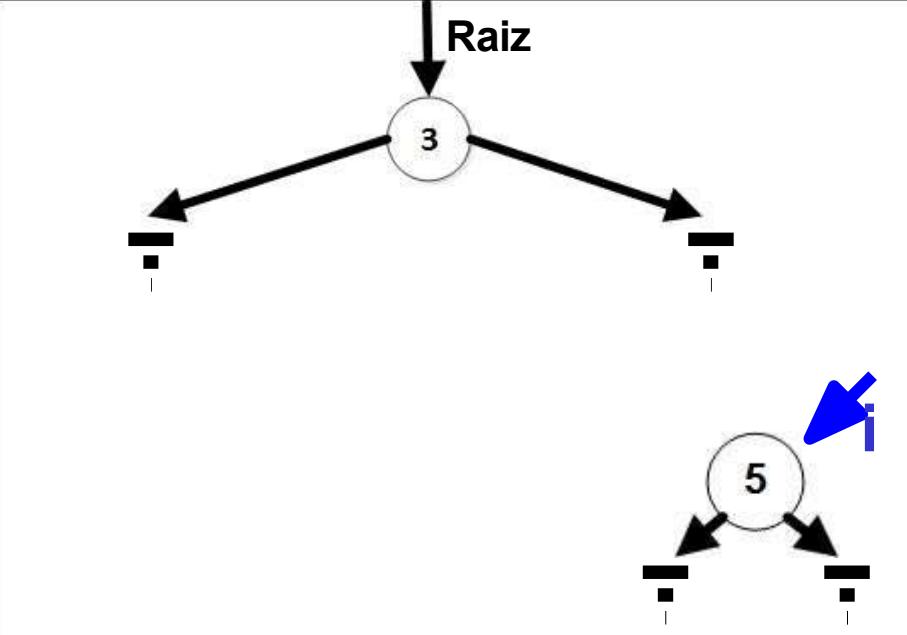
```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 5 x 5 i n(3)

x 5 i n(5)



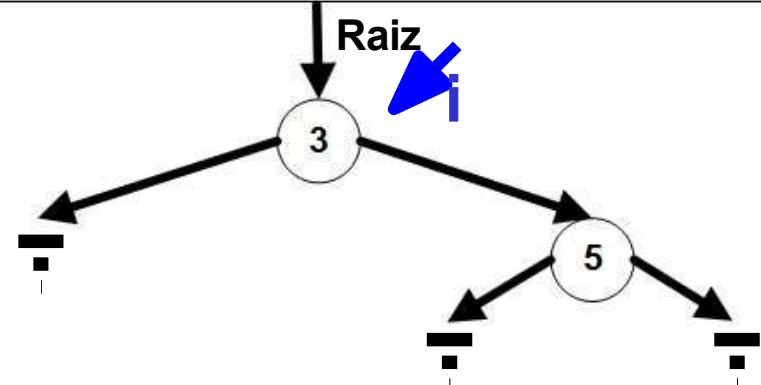
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 5 x 5 i n(3)



Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

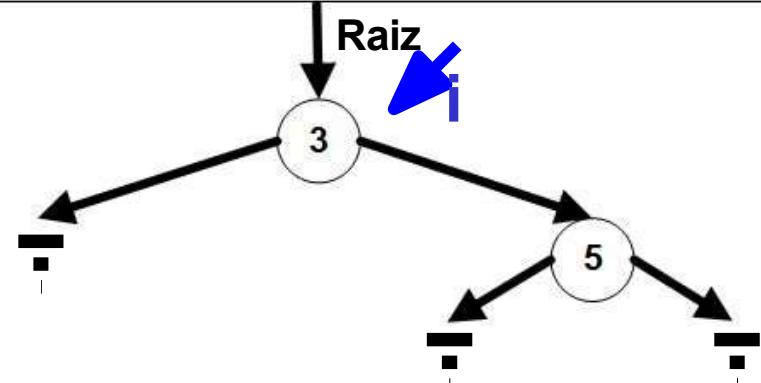
```
No inserir(int x, No i) throws Exception {
```

```
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
}
```

```
return i;
```

retorna o endereço de n(3)

raiz	n(3)	x	5	x	5	i	n(3)
------	------	---	---	---	---	---	------



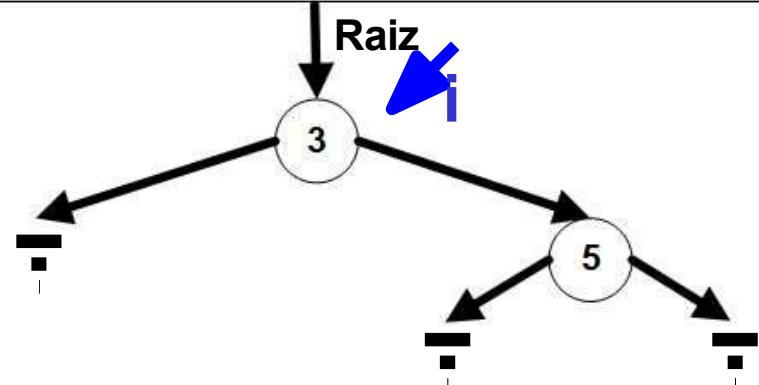
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 5 x 5 i n(3)



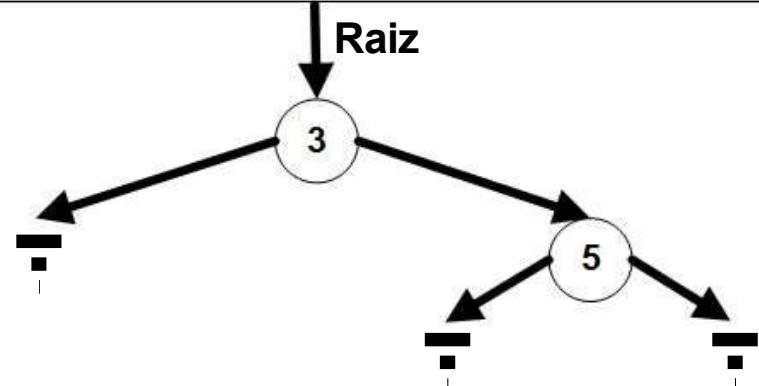
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {  
    raiz = inserir(x, raiz);  
}
```

```
No inserir(int x, No i) throws Exception {  
    if (i == null) {  
        i = new No(x);  
    } else if (x < i.elemento) {  
        i.esq = inserir(x, i.esq);  
    } else if (x > i.elemento) {  
        i.dir = inserir(x, i.dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz n(3) x



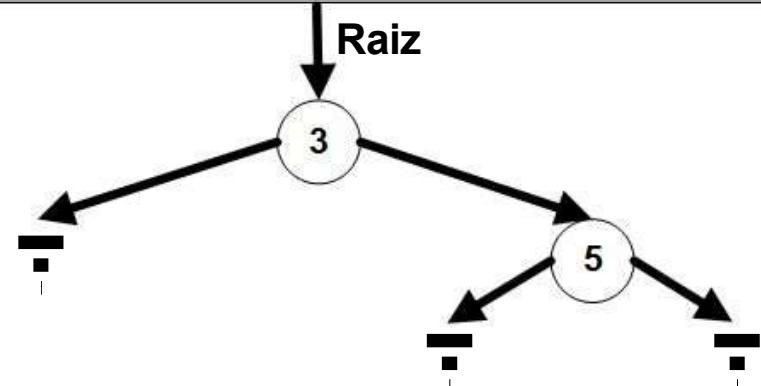
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {  
    raiz = inserir(x, raiz);  
}
```

```
No inserir(int x, No i) throws Exception {  
    if (i == null) {  
        i = new No(x);  
    } else if (x < i.elemento) {  
        i.esq = inserir(x, i.esq);  
    } else if (x > i.elemento) {  
        i.dir = inserir(x, i.dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz
n(3)



Inserção em Java com Retorno de Referência

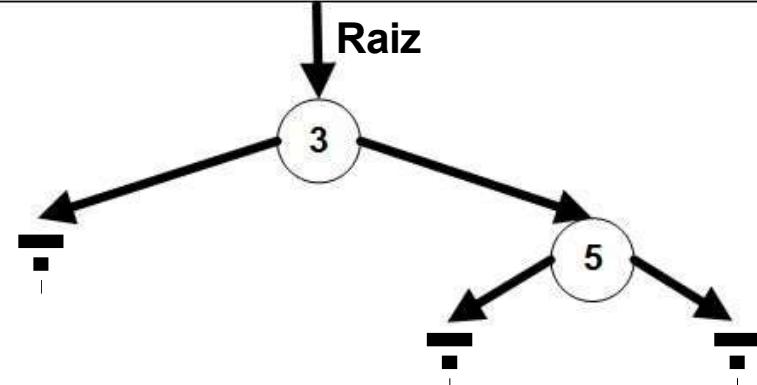
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

```
void inserir(int x) throws Exception {  
    raiz = inserir(x, raiz);  
}
```

```
No inserir(int x, No i) throws Exception {  
    if (i == null) {  
        i = new No(x);  
    } else if (x < i.elemento) {  
        i.esq = inserir(x, i.esq);  
    } else if (x > i.elemento) {  
        i.dir = inserir(x, i.dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

n(3)

x



Inserção em Java com Retorno de Referência

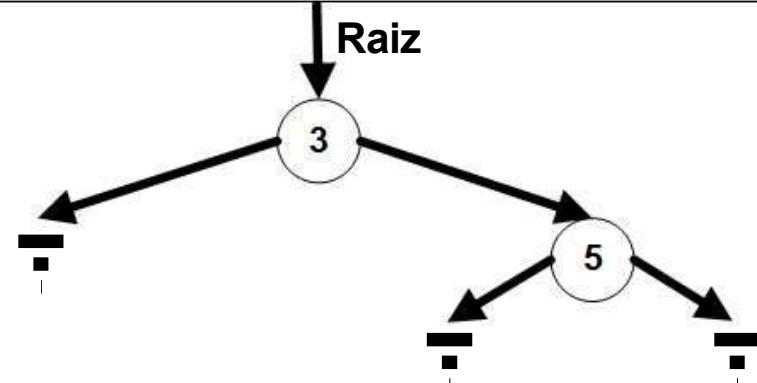
```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {  
    raiz = inserir(x, raiz);  
}
```

```
No inserir(int x, No i) throws Exception {  
    if (i == null) {  
        i = new No(x);  
    } else if (x < i.elemento) {  
        i.esq = inserir(x, i.esq);  
    } else if (x > i.elemento) {  
        i.dir = inserir(x, i.dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

n(3)

x



Inserção em Java com Retorno de Referência

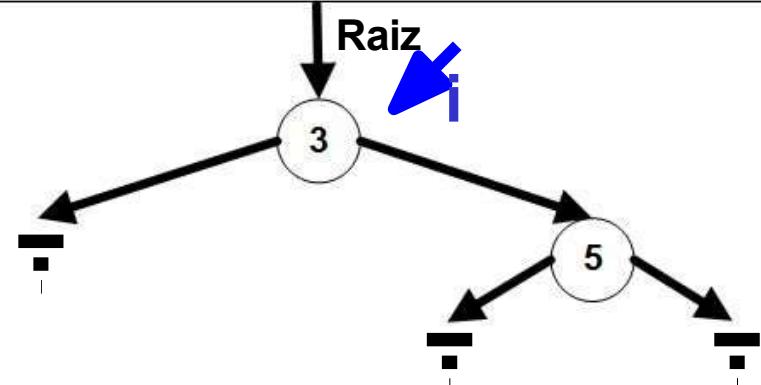
```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
```

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
}
```

raiz n(3) x 1 x 1 i n(3)



Inserção em Java com Retorno de Referência

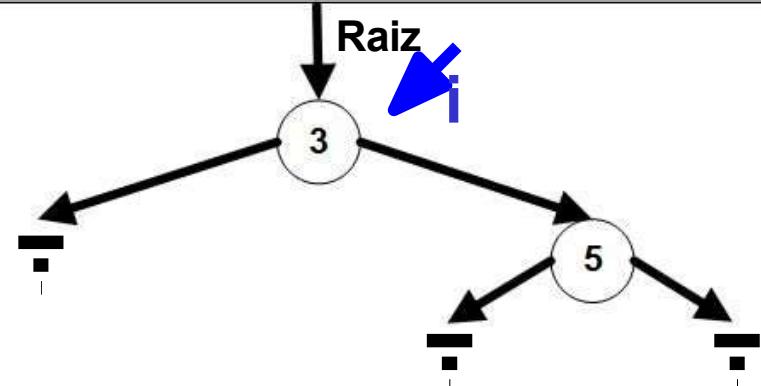
```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
```

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
false: n(3) == null
}
```

raiz n(3) x 1 x 1 i n(3)



Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

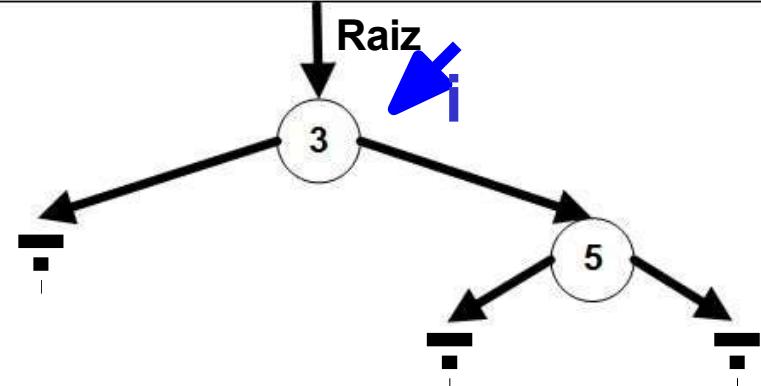
```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
```

```
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

true: 1 < 3

raiz	n(3)	x	1	x	1	i	n(3)
------	------	---	---	---	---	---	------



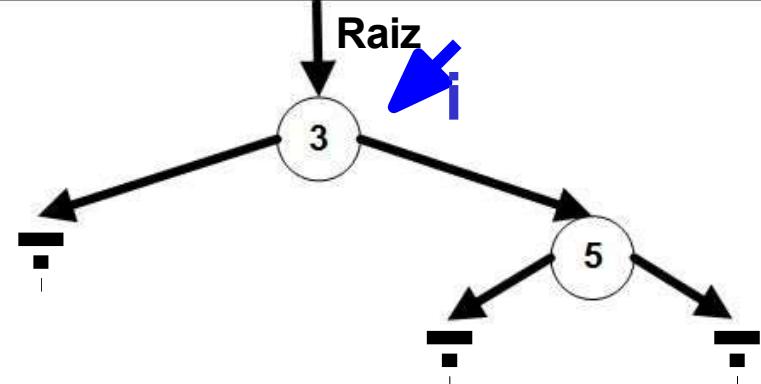
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 1 x 1 i n(3)



Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

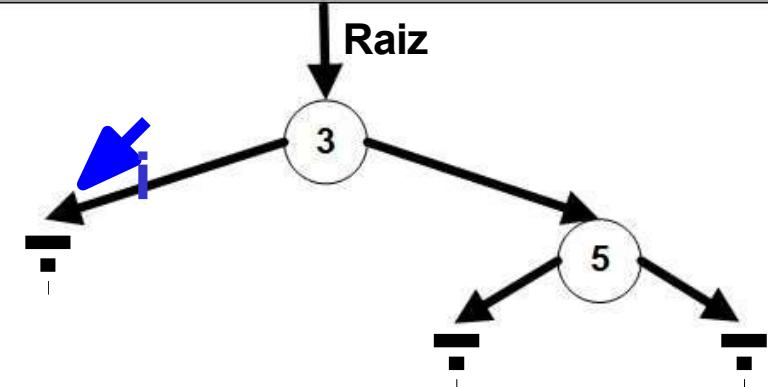
```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
```

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
}
```

raiz n(3) x 1 x 1 i n(3)

x 1 i null



Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

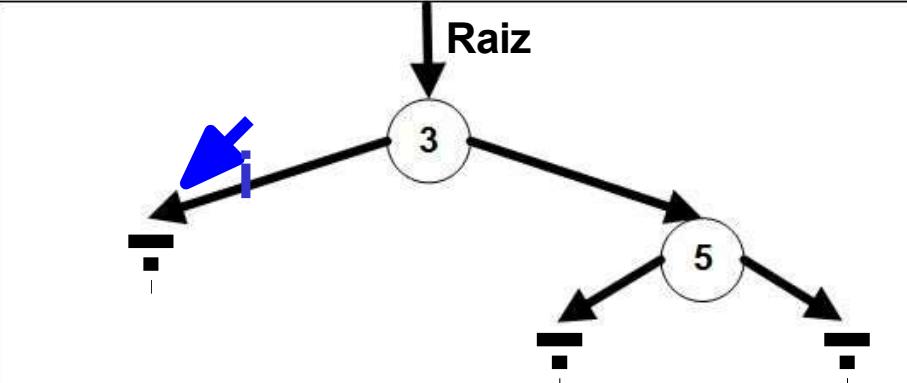
```
No inserir(int x, No i) throws Exception {
```

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
}
```

true: null == null

raiz	n(3)	x	1	x	1	i	n(3)
------	------	---	---	---	---	---	------

x	1	i	null
---	---	---	------



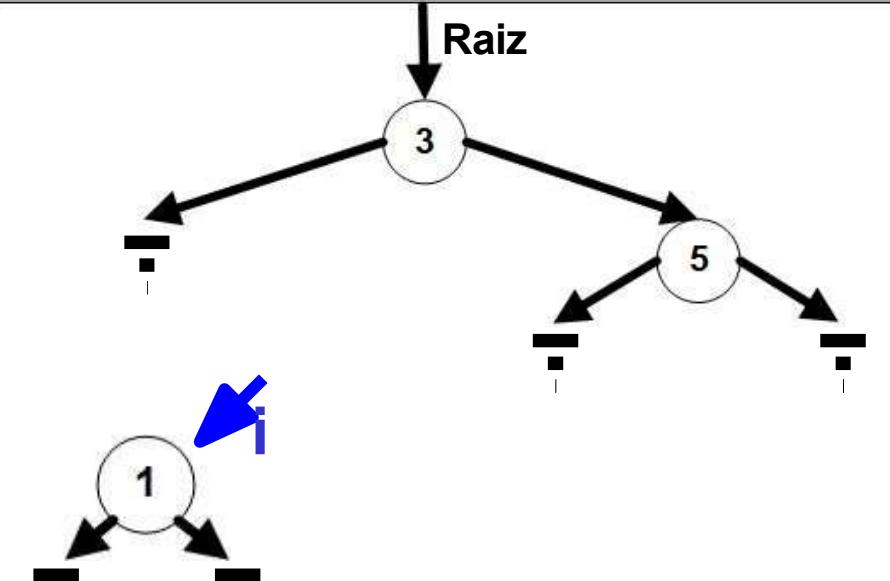
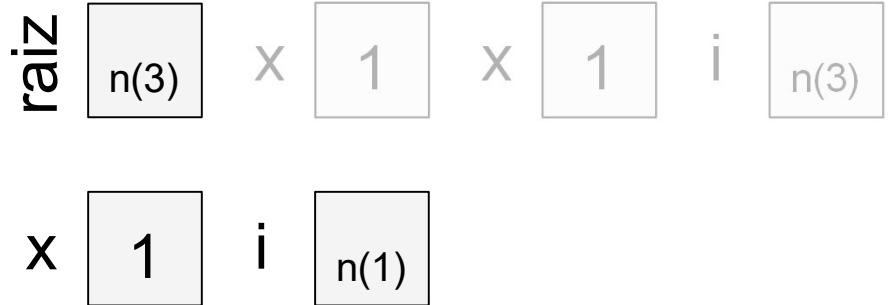
Inserção em Java com Retorno de Referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

```
void inserir(int x) throws Exception {  
    raiz = inserir(x, raiz);  
}
```

No inserir(**int** x, No i) **throws** Exception {

```
if (i == null) {  
    i = new No(x);  
}  
else if (x < i.elemento) {  
    i.esq = inserir(x, i.esq);  
}  
else if (x > i.elemento) {  
    i.dir = inserir(x, i.dir);  
}  
else {  
    throw new Exception("Erro!");  
}  
return i;
```



Inserção em Java com Retorno de Referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

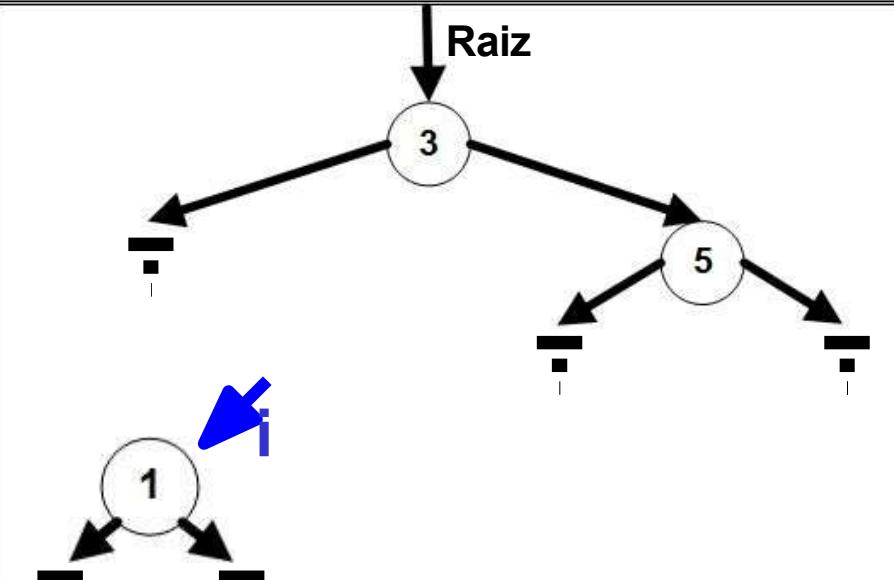
```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

No inserir(int x, No i) throws Exception {

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
```

return i;

retorna o endereço de n(1)



Inserção em Java com Retorno de Referência

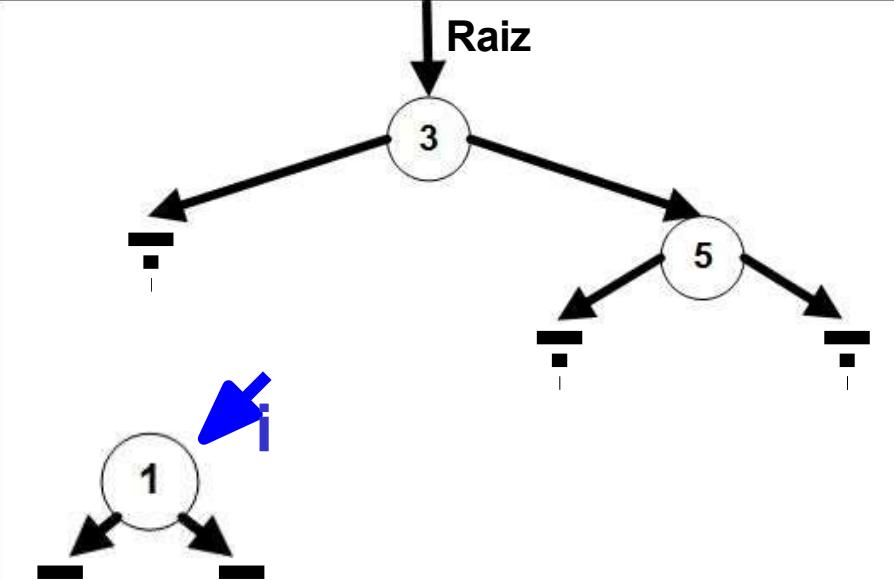
```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 1 x 1 i n(3)

x 1 i n(1)



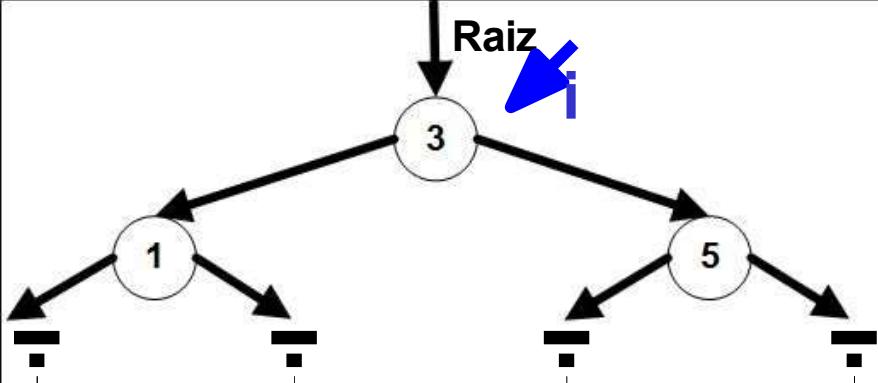
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 1 x 1 i n(3)



Inserção em Java com Retorno de Referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

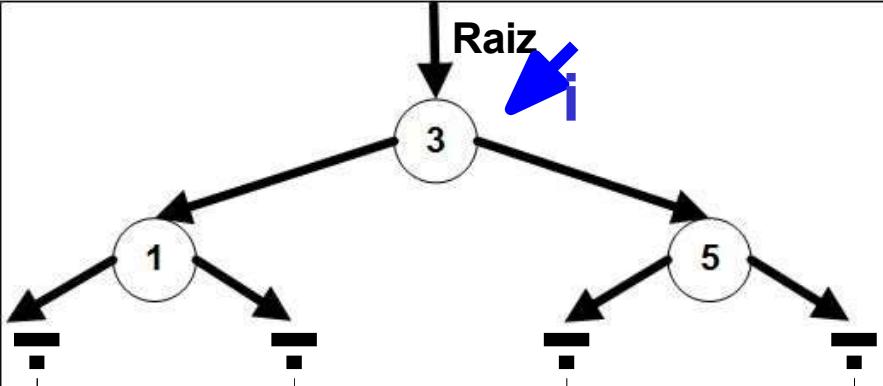
```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

No inserir(int x, No i) throws Exception {

```
if (i == null) {
    i = new No(x);
} else if (x < i.elemento) {
    i.esq = inserir(x, i.esq);
} else if (x > i.elemento) {
    i.dir = inserir(x, i.dir);
} else {
    throw new Exception("Erro!");
}
return i;
```

} retorna o endereço de n(3)

raiz n(3) x 1 x 1 i n(3)



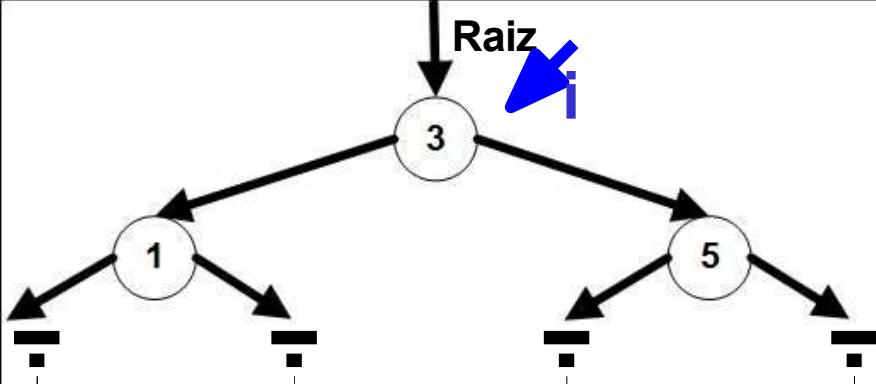
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 1 x 1 i n(3)



Inserção em Java com Retorno de Referência

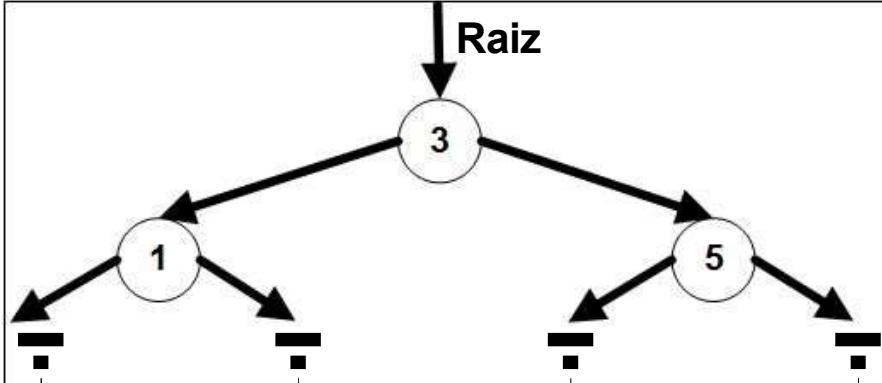
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

```
void inserir(int x) throws Exception {
    raiz = inserir(x, raiz);
}
```

```
No inserir(int x, No i) throws Exception {
    if (i == null) {
        i = new No(x);
    } else if (x < i.elemento) {
        i.esq = inserir(x, i.esq);
    } else if (x > i.elemento) {
        i.dir = inserir(x, i.dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

n(3)

x



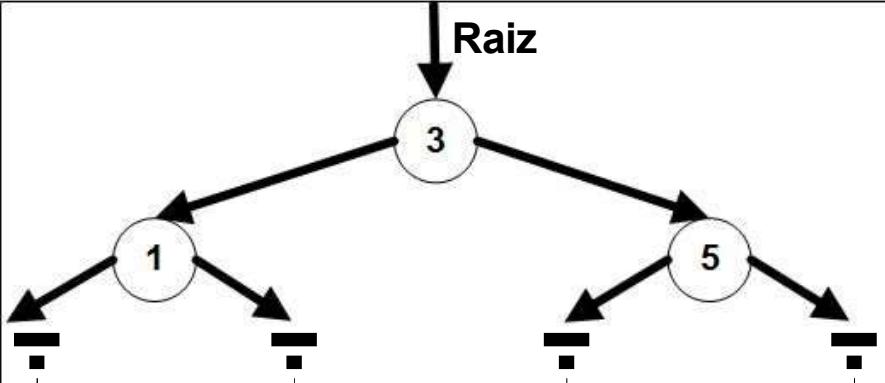
Inserção em Java com Retorno de Referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
void inserir(int x) throws Exception {  
    raiz = inserir(x, raiz);  
}
```

```
No inserir(int x, No i) throws Exception {  
    if (i == null) {  
        i = new No(x);  
    } else if (x < i.elemento) {  
        i.esq = inserir(x, i.esq);  
    } else if (x > i.elemento) {  
        i.dir = inserir(x, i.dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

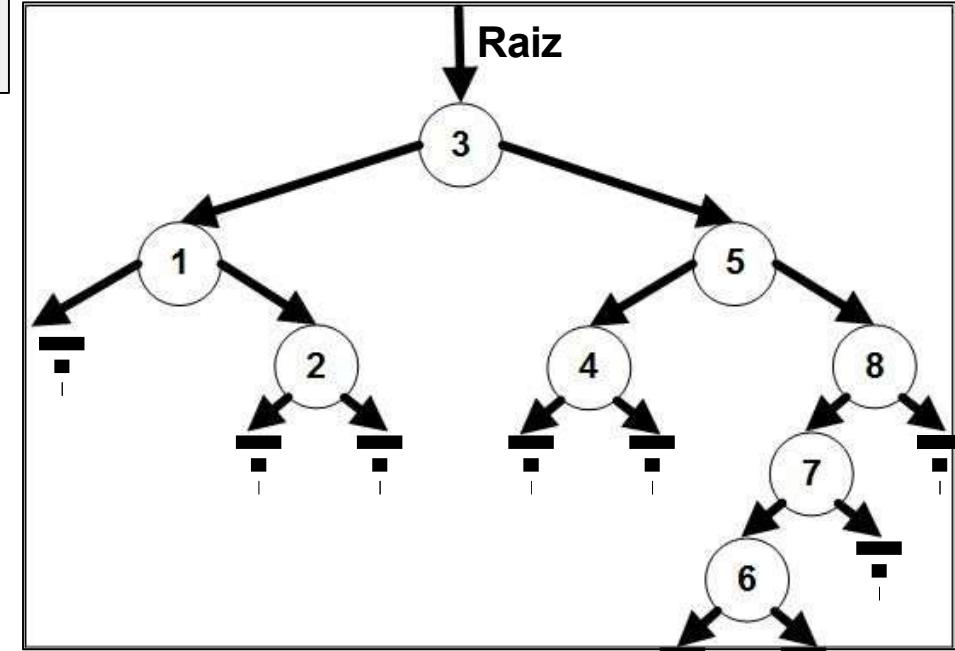
raiz
n(3)



Inserção em Java com Retorno de Referência

```
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) { }  
}
```

Após a inserção do 8, 2, 4, 7 e 6, temos:



Inserção em Java com Passagem de Pai

```
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) { }  
}
```

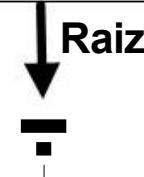


Vamos inserir os elementos
3, 5, 1 e 8
(várias chamadas do inserir)

Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    }  
}  
  
void inserirPai(int x, No i, No pai) throws Exception {  
    ■ ■ ■  
}
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    } }
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    ■ ■ ■  
}
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    } } true: null == null
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    ■ ■ ■  
}
```

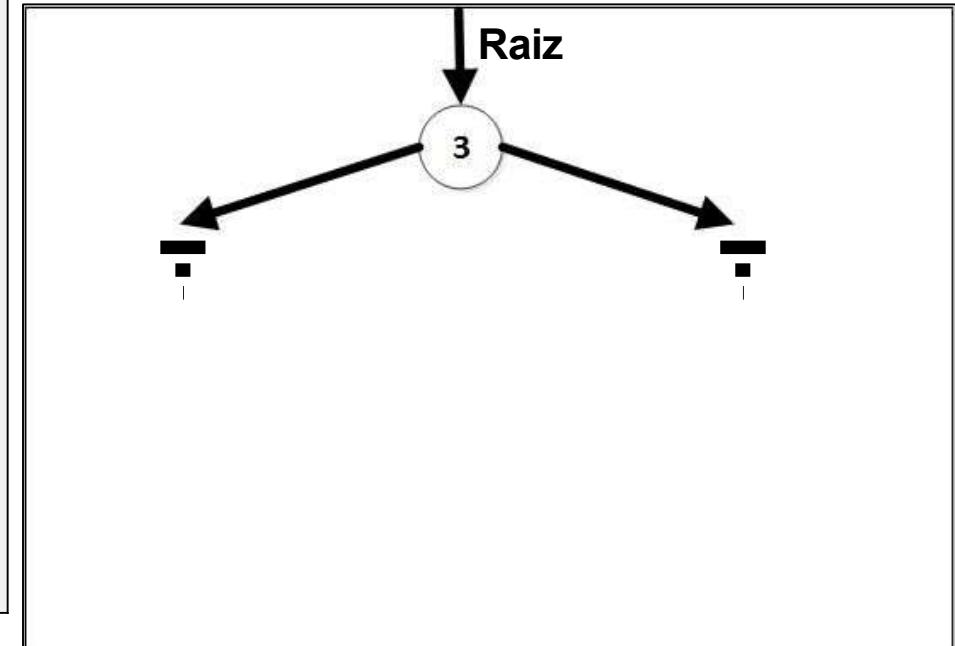


Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    } }
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    ■ ■ ■  
}
```

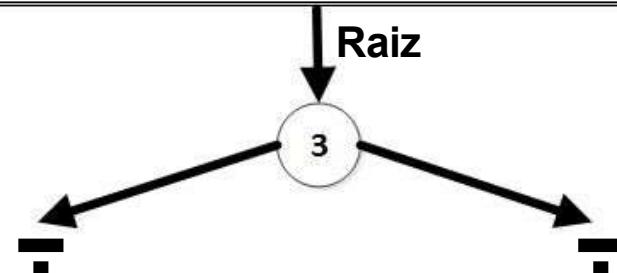


Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    }  
}
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    ■ ■ ■  
}
```

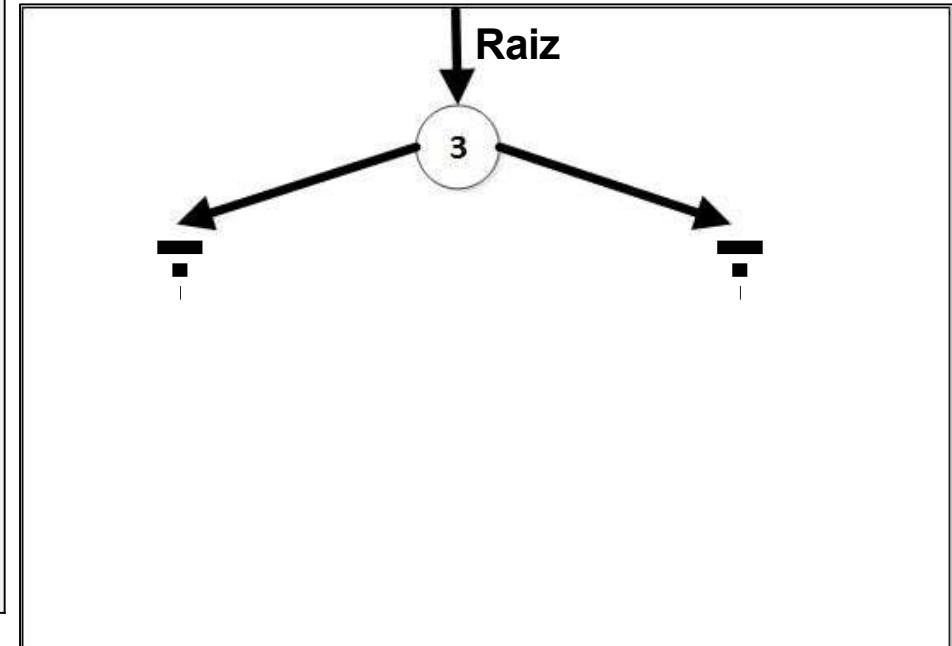


Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    } }
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    ■ ■ ■  
}
```

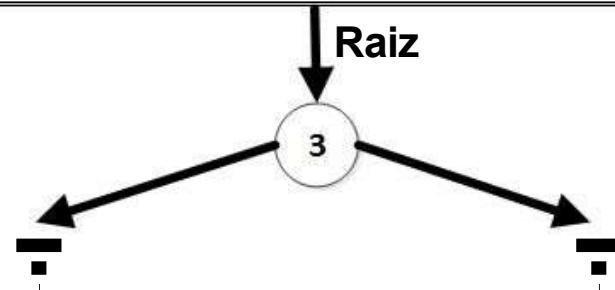


Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    } }  
                                false: n(3) ≠ null
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    ■ ■ ■  
}
```



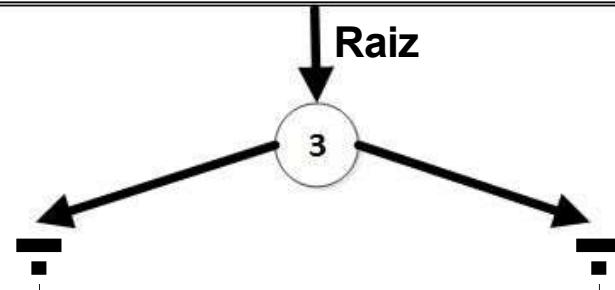
Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {
    if (raiz == null) {
        raiz = new No(x);
    } else if (x < raiz.elemento) {
        inserirPai(x, raiz.esq, raiz);
    } else if (x > raiz.elemento) {
        inserirPai(x, raiz.dir, raiz);
    } else {
        throw new Exception("Erro!");
    }
} }           false: 5 < 3
```

```
void inserirPai(int x, No i, No pai) throws Exception {
```

```
}
```

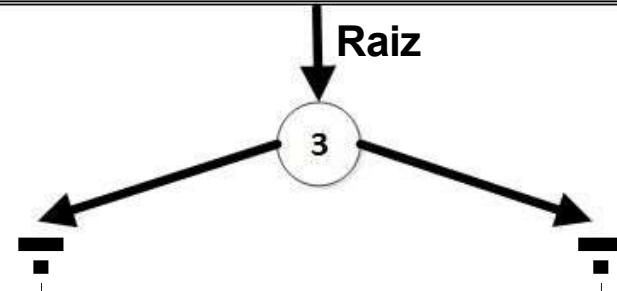


Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    } }  
true: 5 > 3
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    ■ ■ ■  
}
```

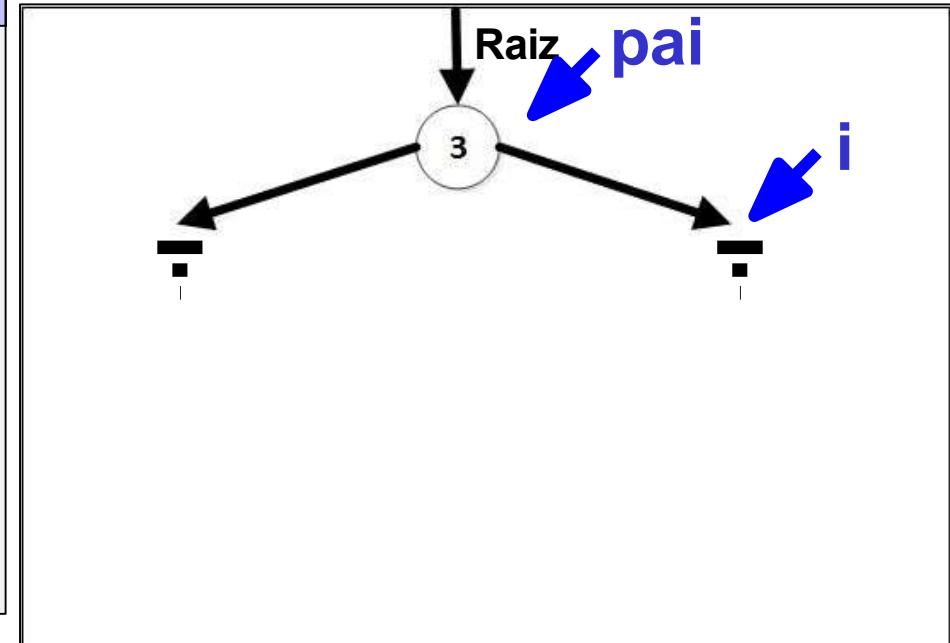


Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    }  
}
```

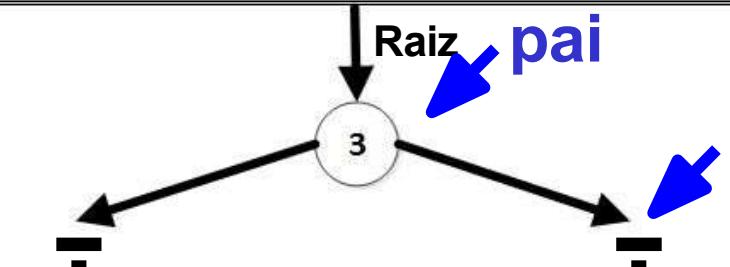
```
void inserirPai(int x, No i, No pai) throws Exception {  
    ...  
}
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

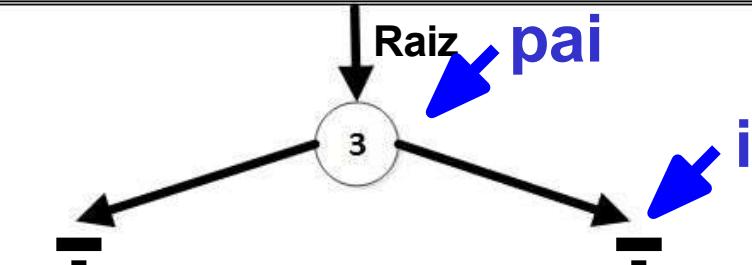
```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        If (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

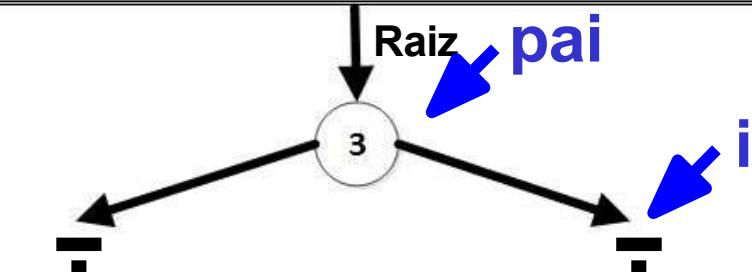
```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        if (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}  
  
true: null == null
```



Inserção em Java com Passagem de Pai

//Inserir 3, 5, 1, 8

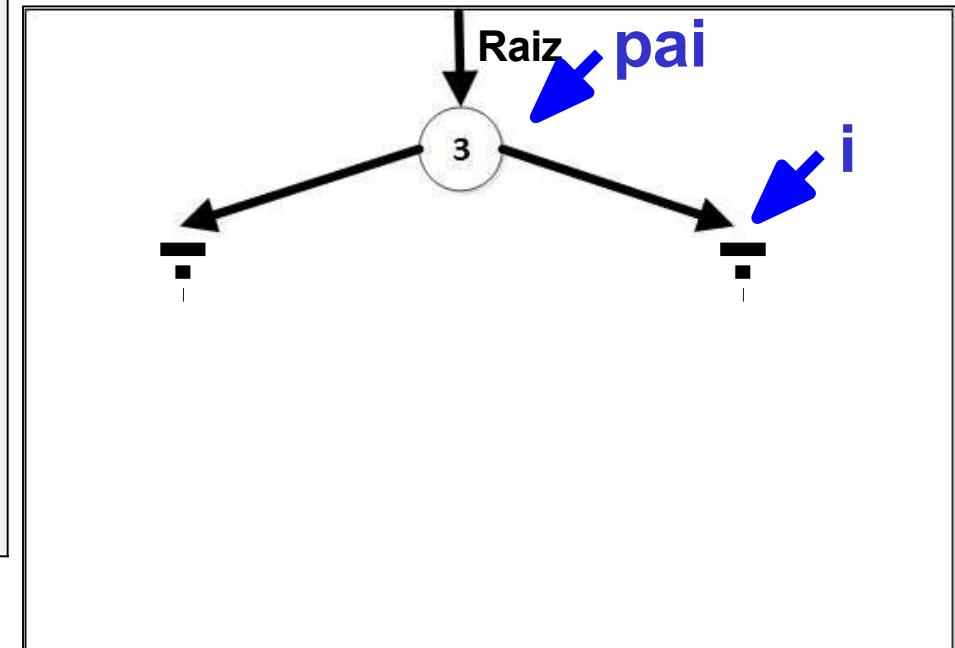
```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        If (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}  
  
false: 5 < 3
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

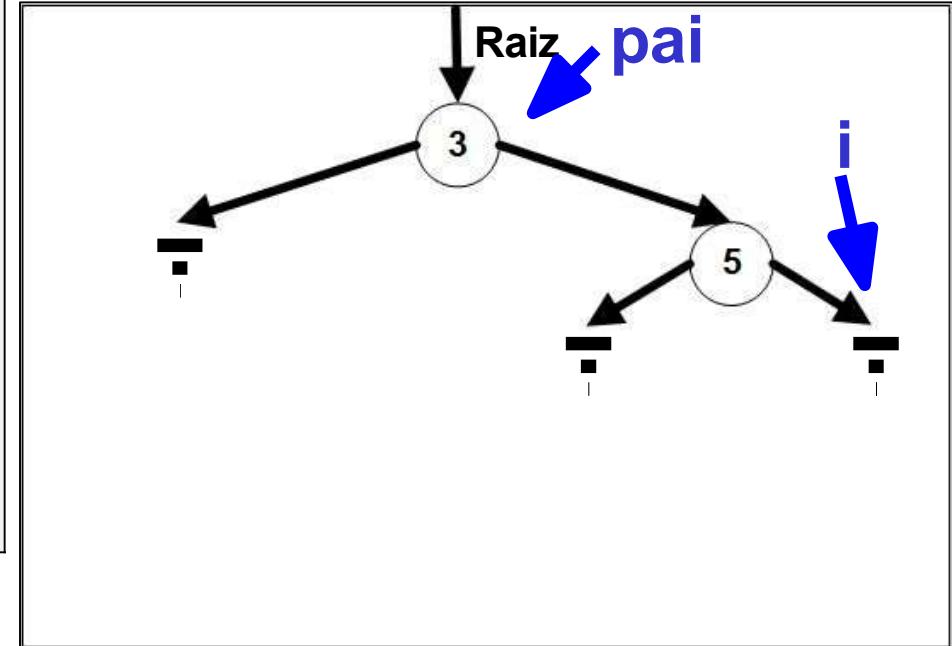
```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        If (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}  
true: 5 > 3
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

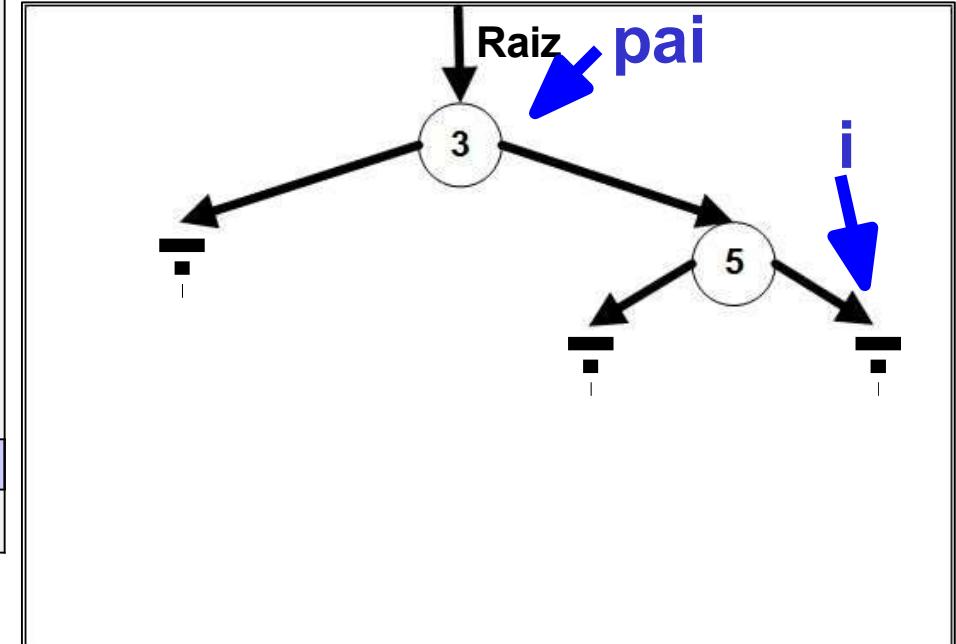
```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        If (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        If (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}
```

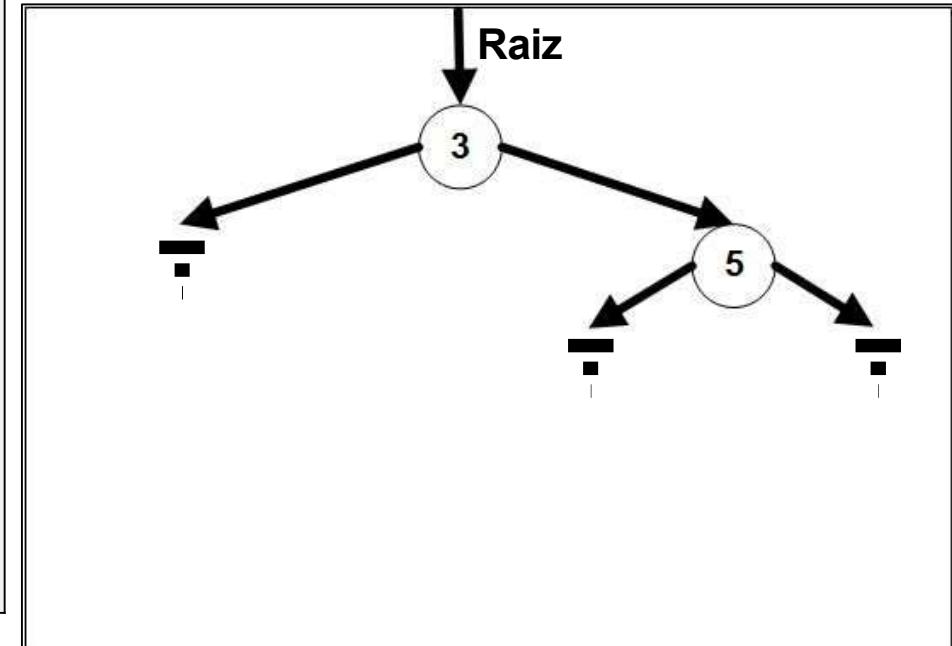


Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    } }
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    ■ ■ ■  
}
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

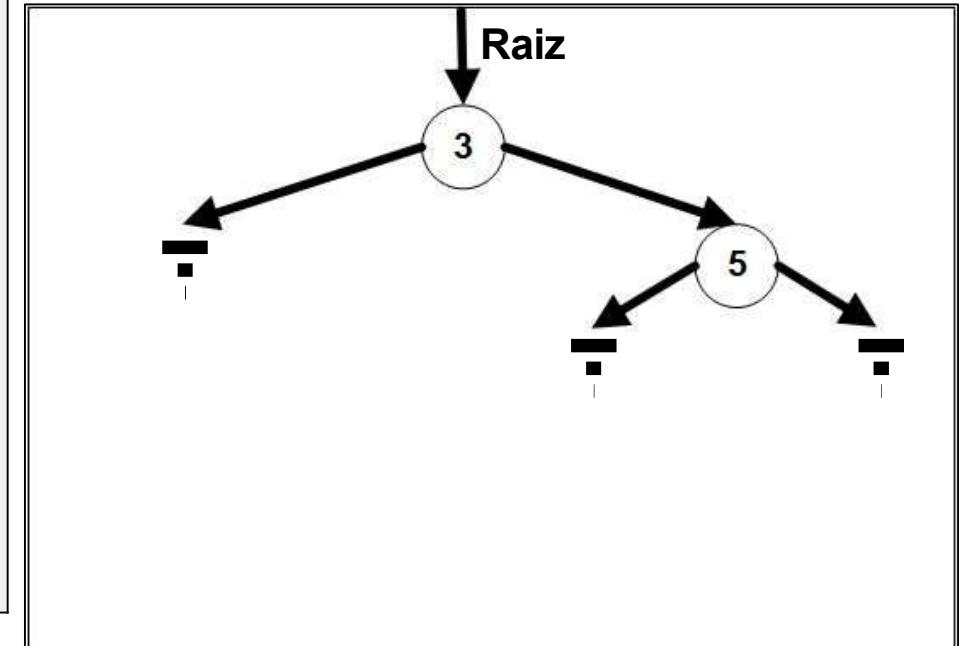
```
void inserirPai(int x) throws Exception {
    if (raiz == null) {
        raiz = new No(x);
    } else if (x < raiz.elemento) {
        inserirPai(x, raiz.esq, raiz);
    } else if (x > raiz.elemento) {
        inserirPai(x, raiz.dir, raiz);
    } else {
        throw new Exception("Erro!");
    }
}
```

false: n(3) ≠ null

```
void inserirPai(int x, No i, No pai) throws Exception {
```

■ ■ ■

```
}
```

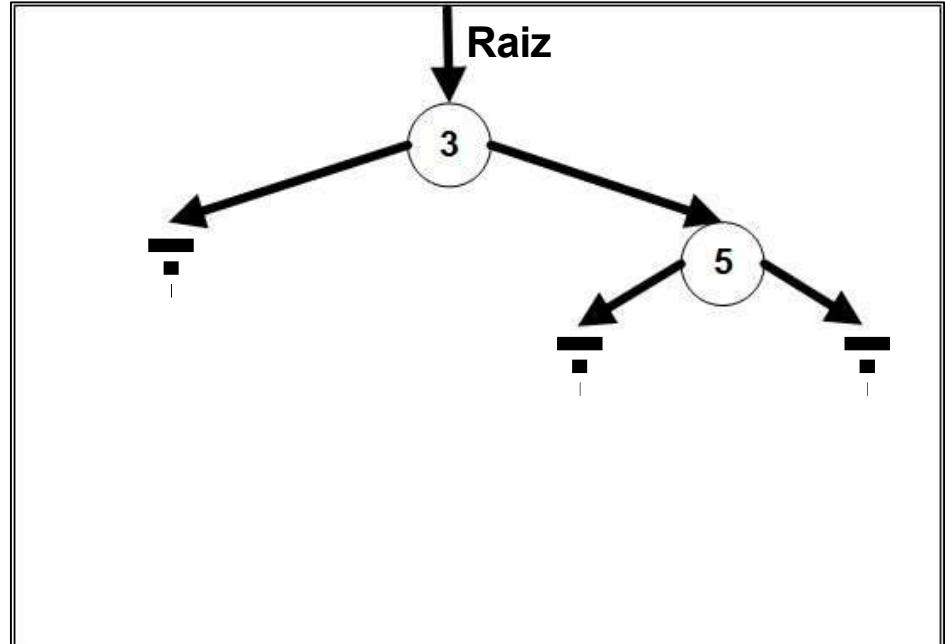


Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    } }  
true: 1 < 3
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    ...  
}
```

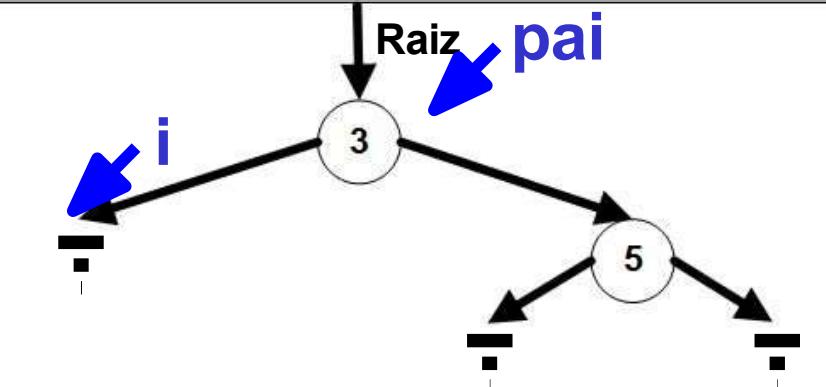


Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {
    if (raiz == null) {
        raiz = new No(x);
    } else if (x < raiz.elemento) {
        inserirPai(x, raiz.esq, raiz);
    } else if (x > raiz.elemento) {
        inserirPai(x, raiz.dir, raiz);
    } else {
        throw new Exception("Erro!");
    }
}

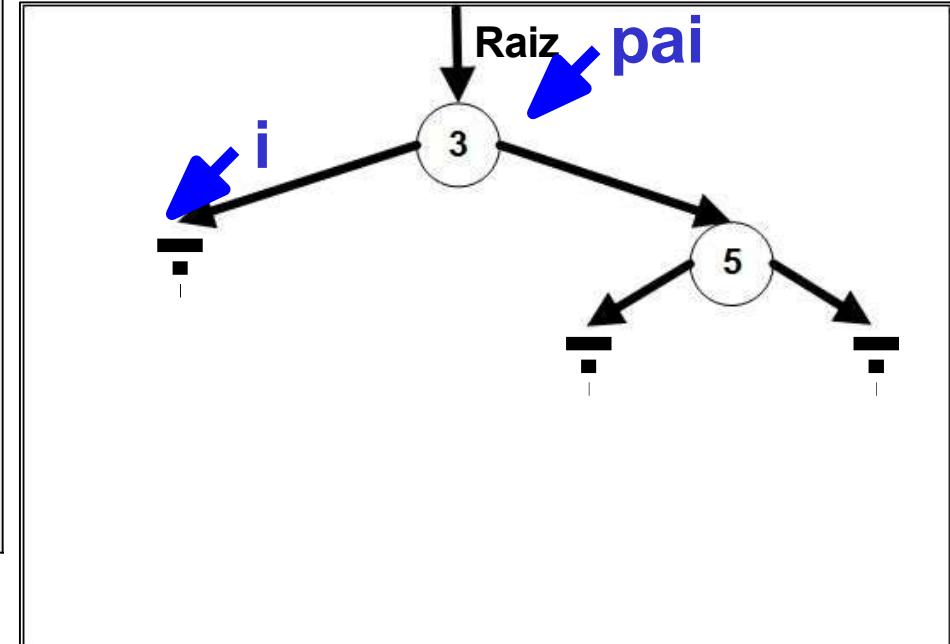
void inserirPai(int x, No i, No pai) throws Exception {
    ...
}
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

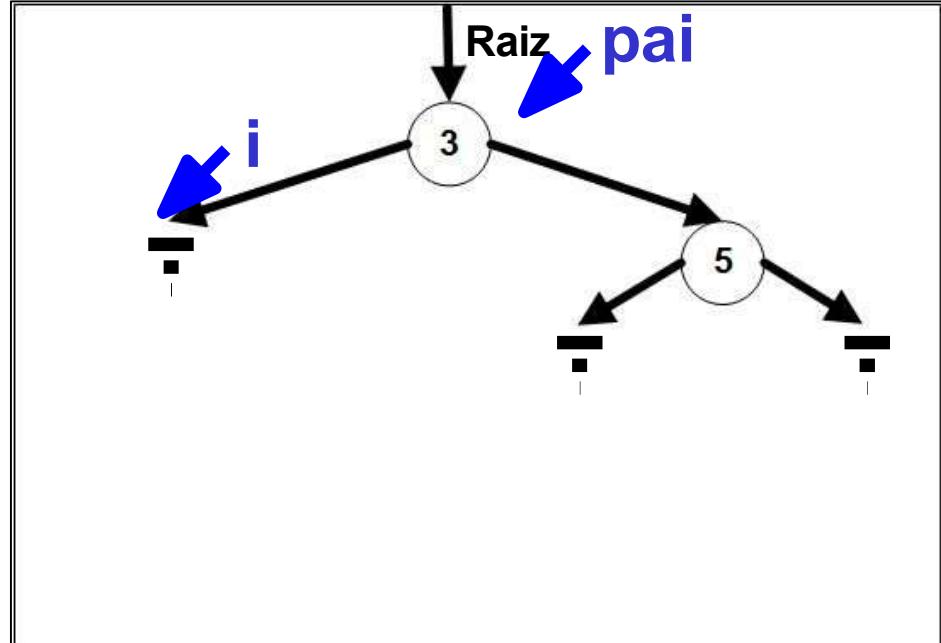
```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        If (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}
```



Inserção em Java com Passagem de Pai

//Inserir 3, 5, 1, 8

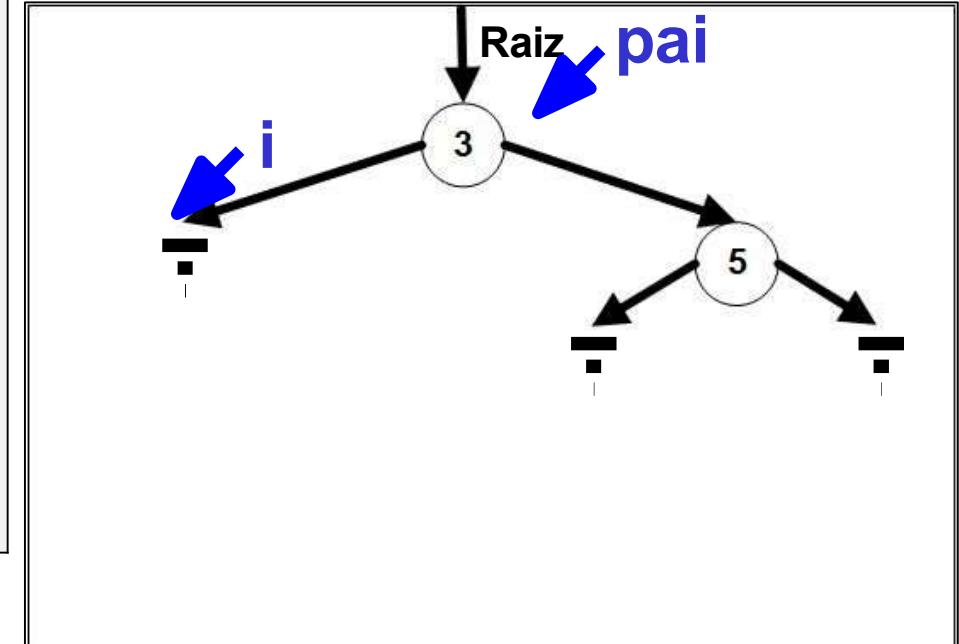
```
void inserirPai(int x, No i, No pai) throws Exception {
    if (i == null) {
        if (x < pai.elemento){
            pai.esq = new No(x);
        } else {
            pai.dir = new No(x);
        }
    } else if (x < i.elemento) {
        inserirPai(x, i.esq, i);
    } else if (x > i.elemento) {
        inserirPai(x, i.dir, i);
    } else {
        throw new Exception("Erro!");
    }
}
true: null == null
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

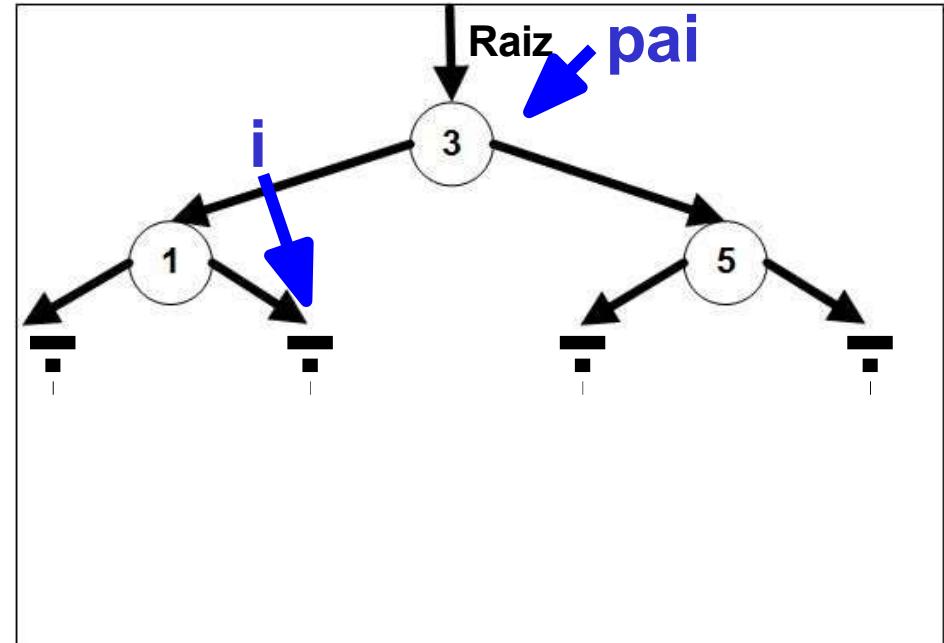
```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        If (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}  
  
true: 1 < 3
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

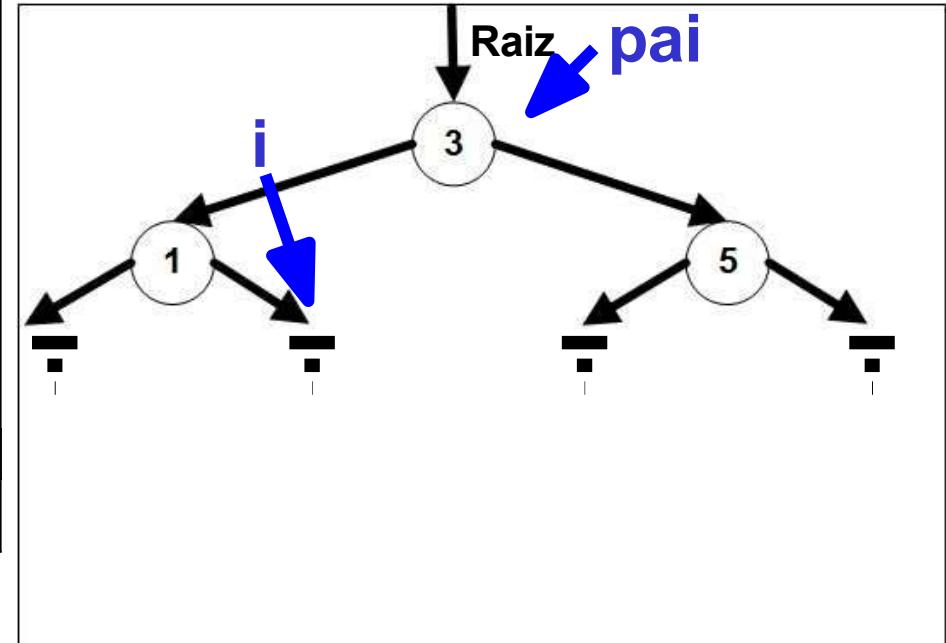
```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        If (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        If (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}
```

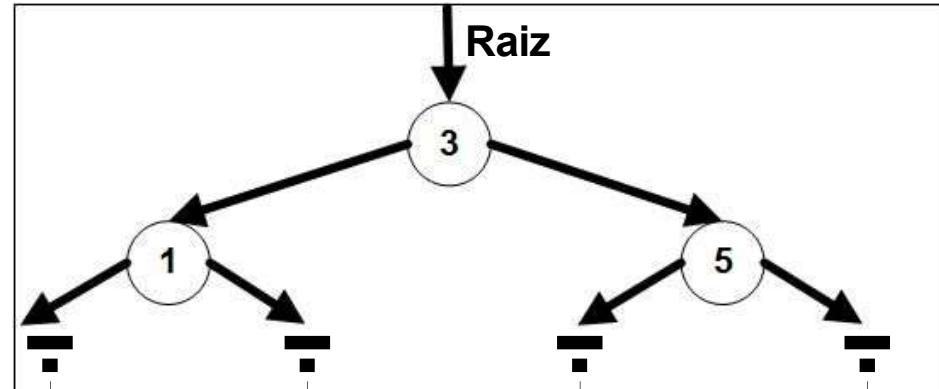


Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    } }
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    ■ ■ ■  
}
```



Inserção em Java com Passagem de Pai

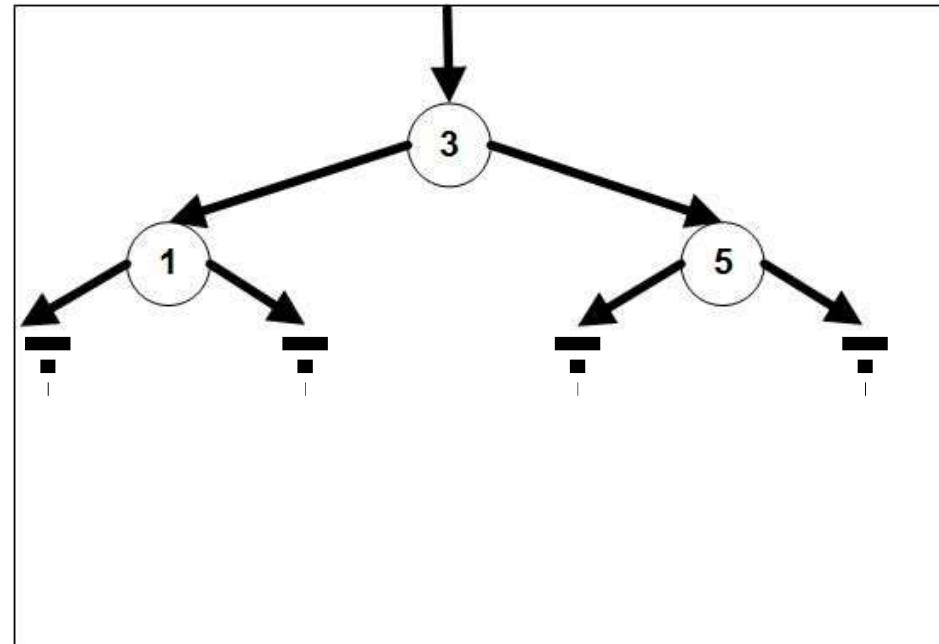
//Inserir 3, 5, 1, 8

```
void inserirPai(int x) throws Exception {
    if (raiz == null) {
        raiz = new No(x);
    } else if (x < raiz.elemento) {
        inserirPai(x, raiz.esq, raiz);
    } else if (x > raiz.elemento) {
        inserirPai(x, raiz.dir, raiz);
    } else {
        throw new Exception("Erro!");
    }
} } false: n(3) ≠ null
```

```
void inserirPai(int x, No i, No pai) throws Exception {
```

■ ■ ■

}

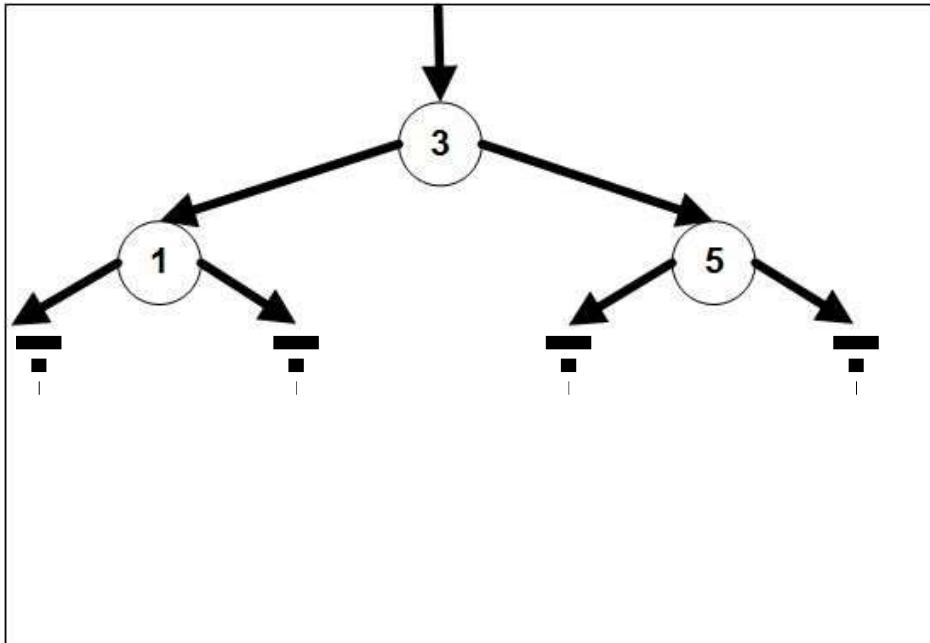


Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {  
    if (raiz == null) {  
        raiz = new No(x);  
    } else if (x < raiz.elemento) {  
        inserirPai(x, raiz.esq, raiz);  
    } else if (x > raiz.elemento) {  
        inserirPai(x, raiz.dir, raiz);  
    } else {  
        throw new Exception("Erro!");  
    } }  
                                false: 8 < 3
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    ...  
}
```



Inserção em Java com Passagem de Pai

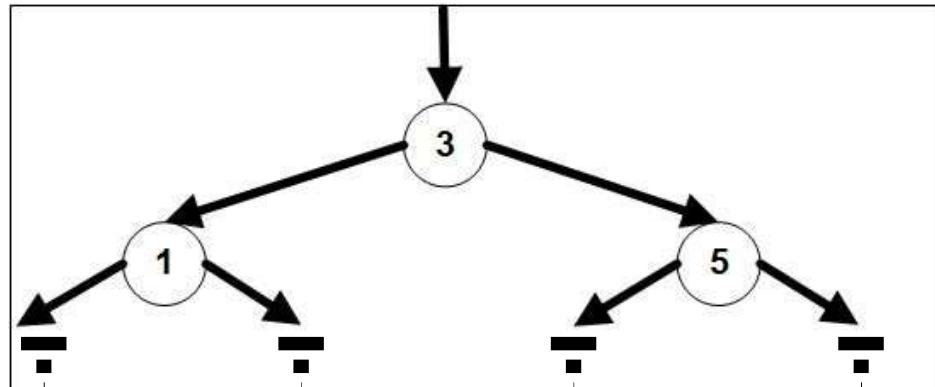
//Inserir 3, 5, 1, 8

```
void inserirPai(int x) throws Exception {
    if (raiz == null) {
        raiz = new No(x);
    } else if (x < raiz.elemento) {
        inserirPai(x, raiz.esq, raiz);
    } else if (x > raiz.elemento) {
        inserirPai(x, raiz.dir, raiz);
    } else {
        throw new Exception("Erro!");
    }
} } true: 8 > 3
```

```
void inserirPai(int x, No i, No pai) throws Exception {
```

■ ■ ■

}

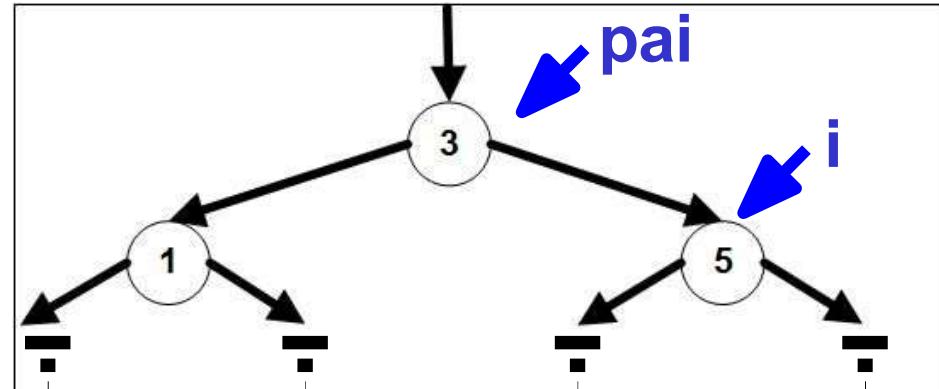


Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x) throws Exception {
    if (raiz == null) {
        raiz = new No(x);
    } else if (x < raiz.elemento) {
        inserirPai(x, raiz.esq, raiz);
    } else if (x > raiz.elemento) {
        inserirPai(x, raiz.dir, raiz);
    } else {
        throw new Exception("Erro!");
    }
}

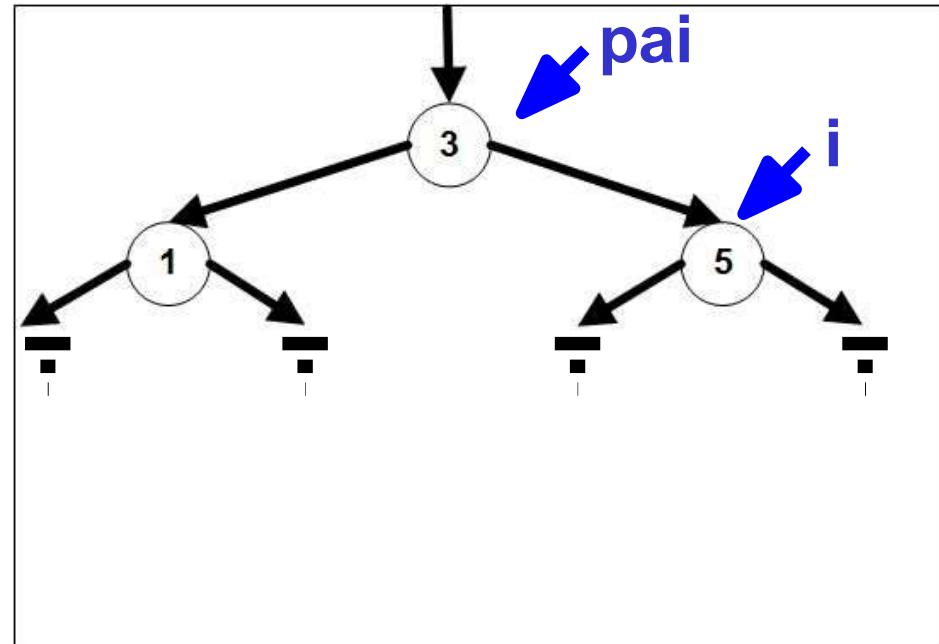
void inserirPai(int x, No i, No pai) throws Exception {
    ...
}
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        if (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}
```

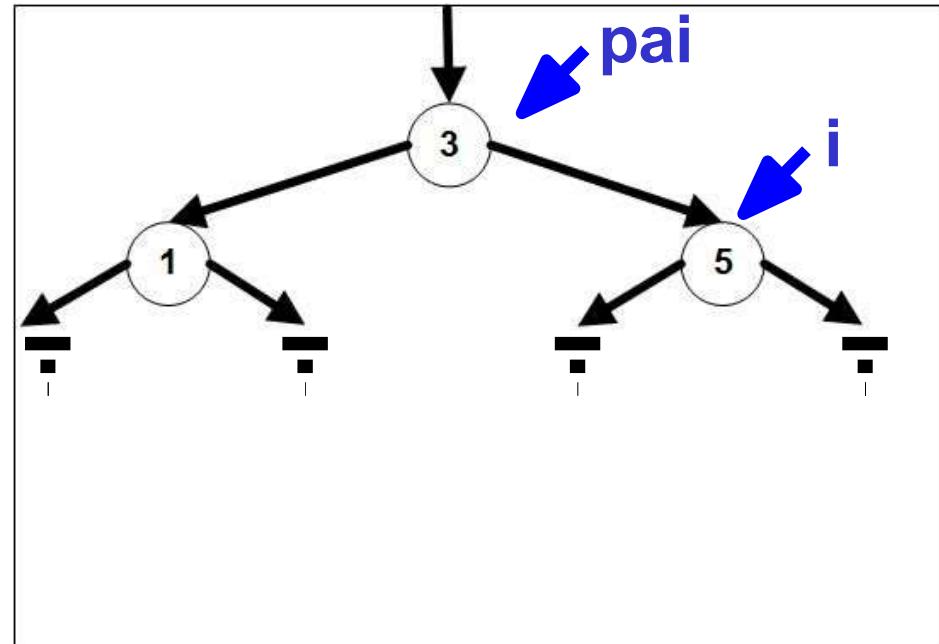


Inserção em Java com Passagem de Pai

//Inserir 3, 5, 1, **8**

```
void inserirPai(int x, No i, No pai) throws Exception {
    if (i == null) {
        if (x < pai.elemento){
            pai.esq = new No(x);
        } else {
            pai.dir = new No(x);
        }
    } else if (x < i.elemento) {
        inserirPai(x, i.esq, i);
    } else if (x > i.elemento) {
        inserirPai(x, i.dir, i);
    } else {
        throw new Exception("Erro!");
    }
}
```

false: n(5) ≠ null

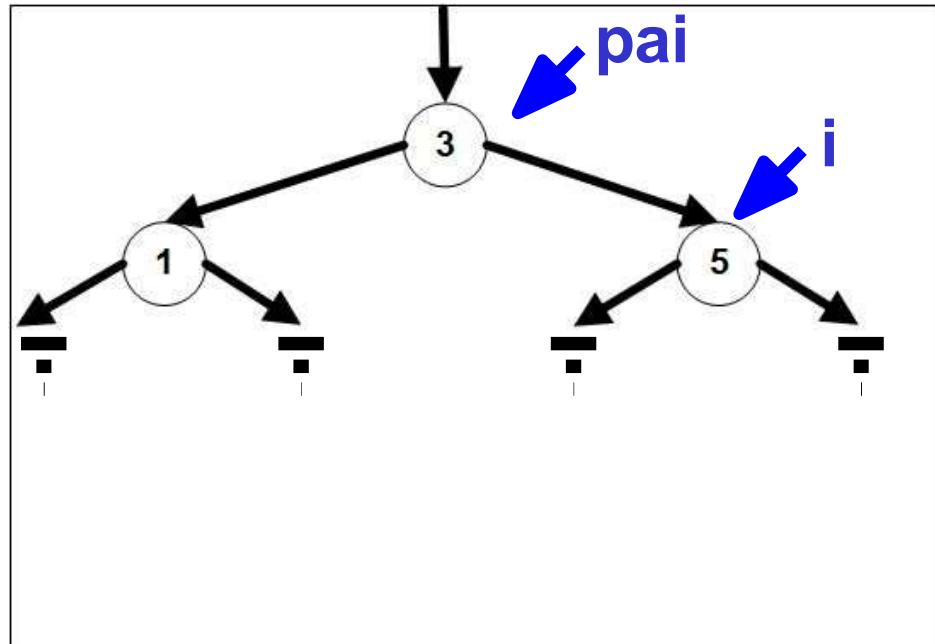


Inserção em Java com Passagem de Pai

//Inserir 3, 5, 1, **8**

```
void inserirPai(int x, No i, No pai) throws Exception {
    if (i == null) {
        If (x < pai.elemento){
            pai.esq = new No(x);
        } else {
            pai.dir = new No(x);
        }
    } else if (x < i.elemento) {
        inserirPai(x, i.esq, i);
    } else if (x > i.elemento) {
        inserirPai(x, i.dir, i);
    } else {
        throw new Exception("Erro!");
    }
}
```

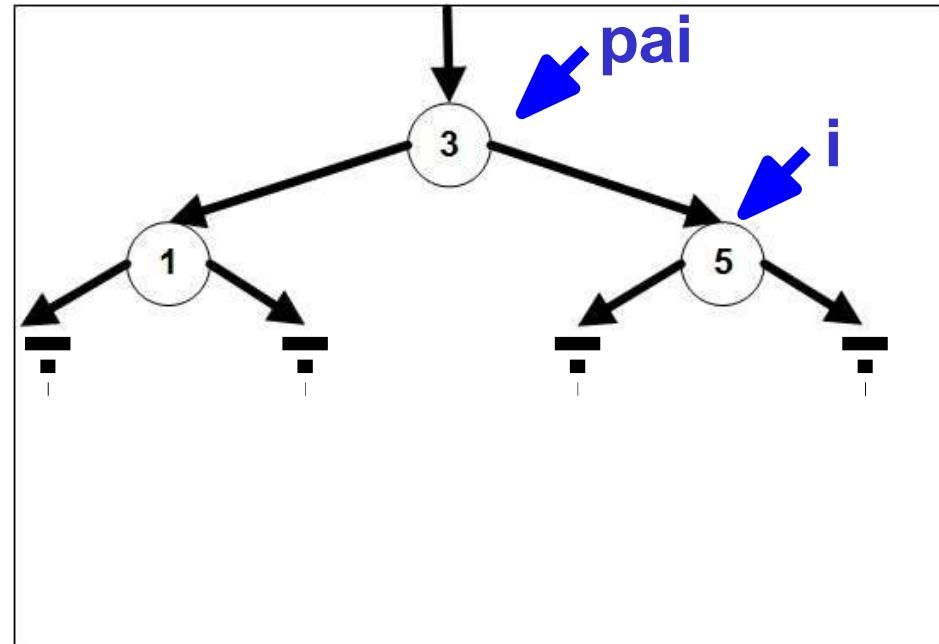
false: **8 < 5**



Inserção em Java com Passagem de Pai

//Inserir 3, 5, 1, **8**

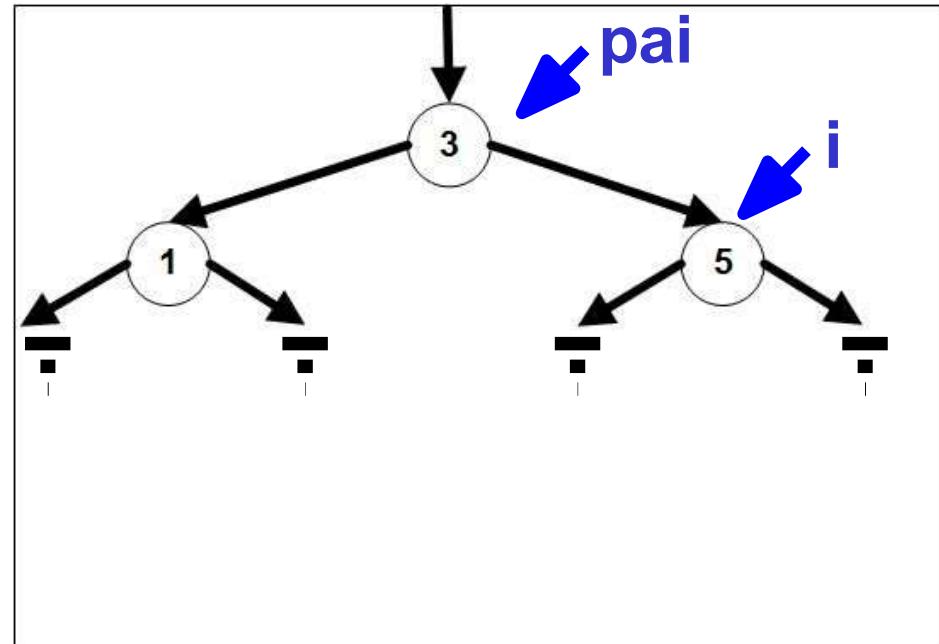
```
void inserirPai(int x, No i, No pai) throws Exception {
    if (i == null) {
        If (x < pai.elemento){
            pai.esq = new No(x);
        } else {
            pai.dir = new No(x);
        }
    } else if (x < i.elemento) {
        inserirPai(x, i.esq, i);
    } else if (x > i.elemento) {
        inserirPai(x, i.dir, i);
    } else {
        throw new Exception("Erro!");
    }
}
true: 8 > 5
```



Inserção em Java com Passagem de Pai

//Inserir 3, 5, 1, **8**

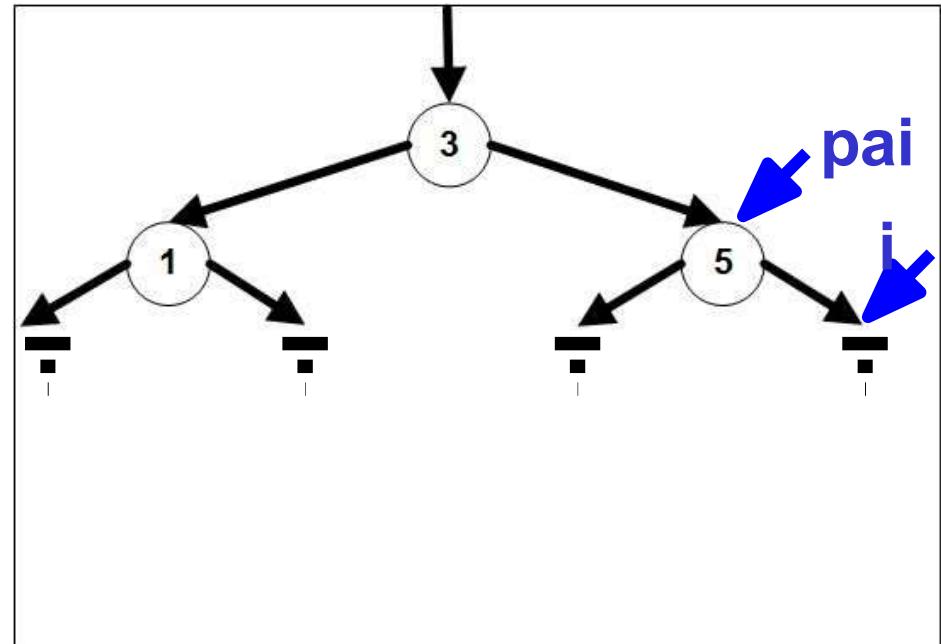
```
void inserirPai(int x, No i, No pai) throws Exception {
    if (i == null) {
        if (x < pai.elemento) {
            pai.esq = new No(x);
        } else {
            pai.dir = new No(x);
        }
    } else if (x < i.elemento) {
        inserirPai(x, i.esq, i);
    } else if (x > i.elemento) {
        inserirPai(x, i.dir, i);
    } else {
        throw new Exception("Erro!");
    }
}
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

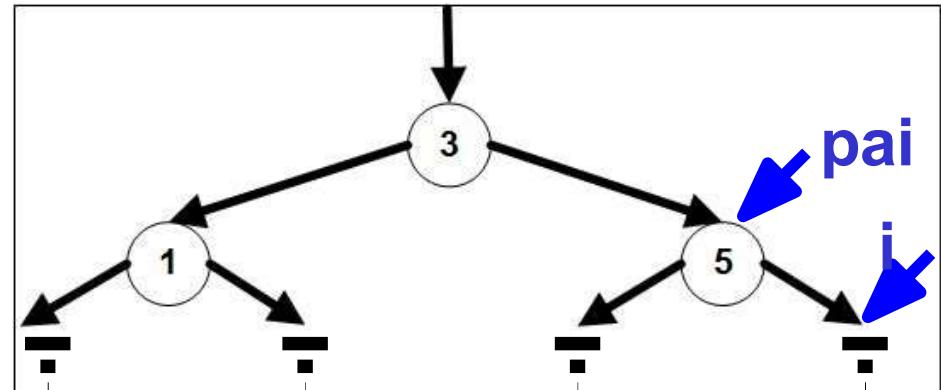
```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        If (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}
```



Inserção em Java com Passagem de Pai

//Inserir 3, 5, 1, 8

```
void inserirPai(int x, No i, No pai) throws Exception {
    if (i == null) {
        if (x < pai.elemento){
            pai.esq = new No(x);
        } else {
            pai.dir = new No(x);
        }
    } else if (x < i.elemento) {
        inserirPai(x, i.esq, i);
    } else if (x > i.elemento) {
        inserirPai(x, i.dir, i);
    } else {
        throw new Exception("Erro!");
    }
}
true: null == null
```

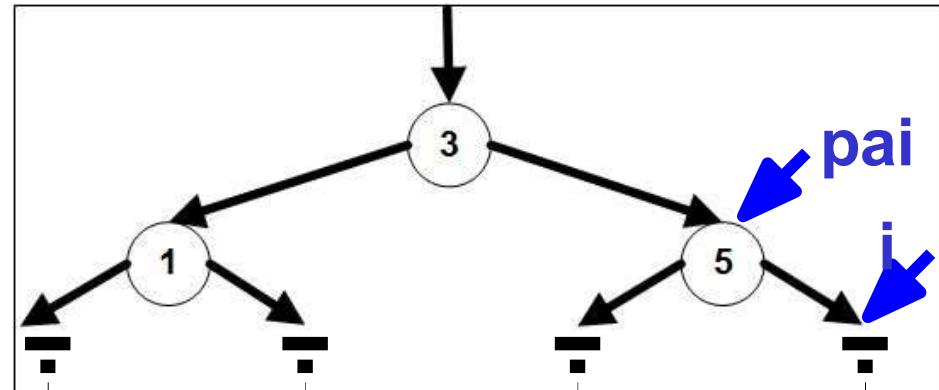


Inserção em Java com Passagem de Pai

//Inserir 3, 5, 1, **8**

```
void inserirPai(int x, No i, No pai) throws Exception {
    if (i == null) {
        If (x < pai.elemento){
            pai.esq = new No(x);
        } else {
            pai.dir = new No(x);
        }
    } else if (x < i.elemento) {
        inserirPai(x, i.esq, i);
    } else if (x > i.elemento) {
        inserirPai(x, i.dir, i);
    } else {
        throw new Exception("Erro!");
    }
}
```

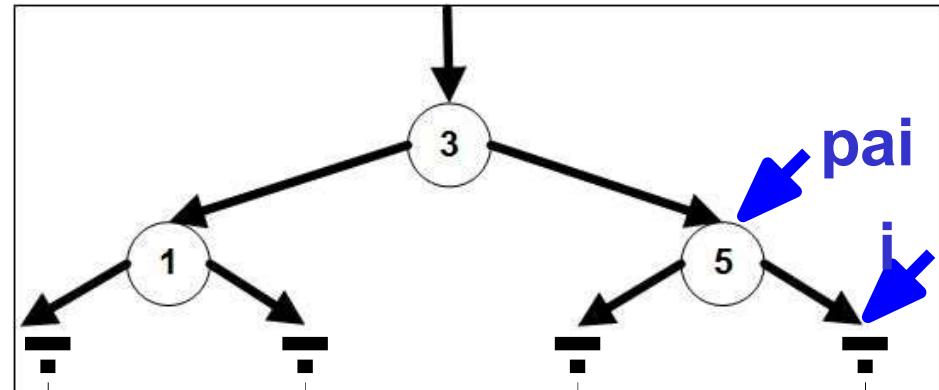
false: **8 < 5**



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

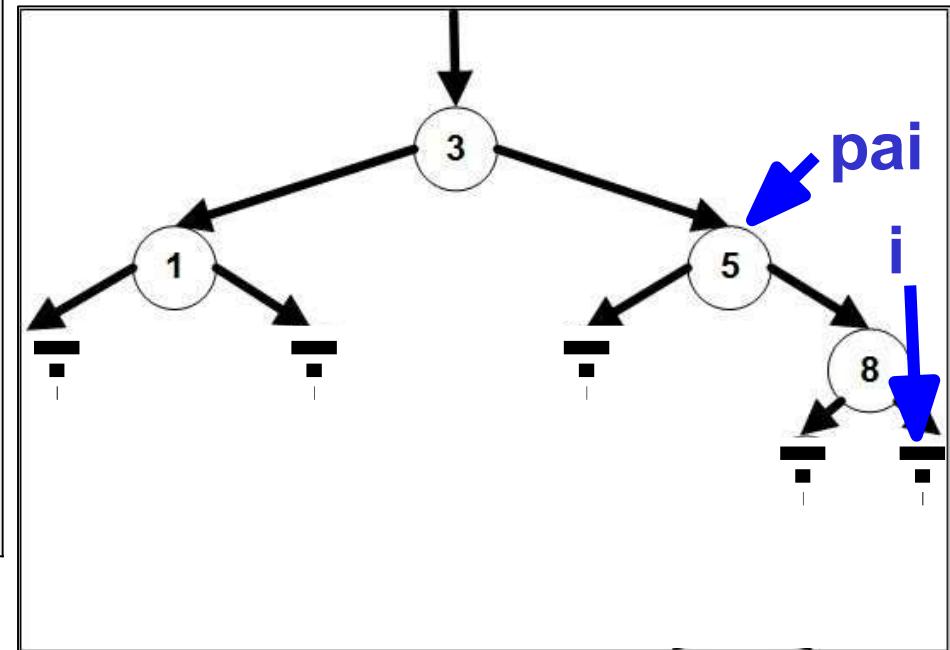
```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        If (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}
```



Inserção em Java com Passagem de Pai

```
//Inserir 3, 5, 1, 8
```

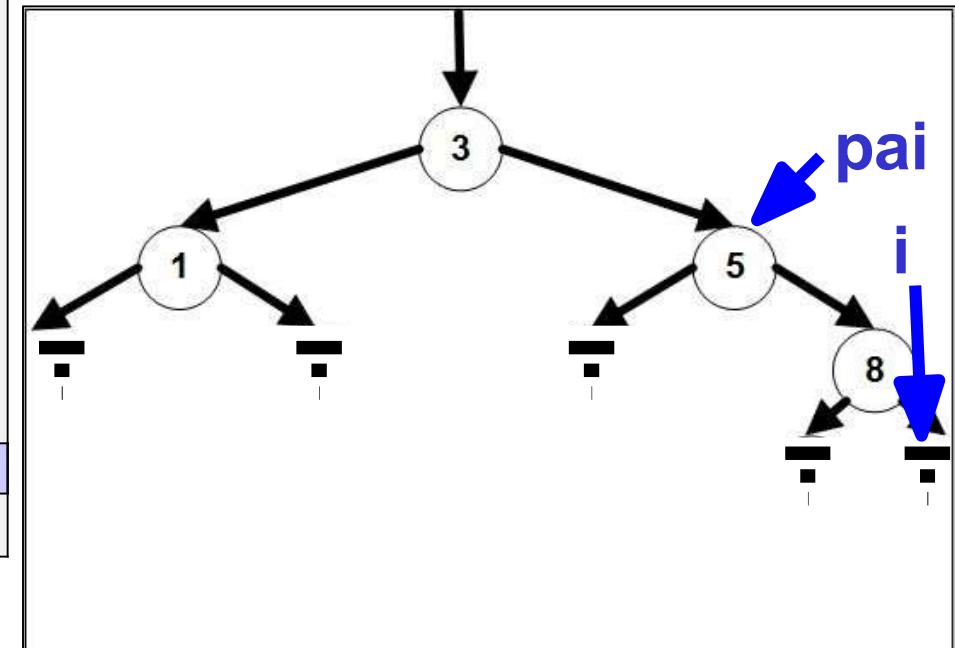
```
void inserirPai(int x, No i, No pai) throws Exception {
    if (i == null) {
        If (x < pai.elemento){
            pai.esq = new No(x);
        } else {
            pai.dir = new No(x);
        }
    } else if (x < i.elemento) {
        inserirPai(x, i.esq, i);
    } else if (x > i.elemento) {
        inserirPai(x, i.dir, i);
    } else {
        throw new Exception("Erro!");
    }
}
```



Inserção em Java com Passagem de Pai

//Inserir 3, 5, 1, 8

```
void inserirPai(int x, No i, No pai) throws Exception {  
    if (i == null) {  
        If (x < pai.elemento){  
            pai.esq = new No(x);  
        } else {  
            pai.dir = new No(x);  
        }  
    } else if (x < i.elemento) {  
        inserirPai(x, i.esq, i);  
    } else if (x > i.elemento) {  
        inserirPai(x, i.dir, i);  
    } else {  
        throw new Exception("Erro!");  
    }  
}
```



Análise de Complexidade da Inserção

- **Melhor Caso:** $\Theta(1)$ comparações e acontece, por exemplo, inserindo na raiz
- **Pior Caso:** $\Theta(n)$ comparações e acontece, por exemplo, quando inserimos os elementos na ordem crescente ou decrescente
- **Caso Médio:** $\Theta(\lg(n))$ comparações e acontece, por exemplo, quando inserimos um elemento na folha de uma árvore balanceada. Lembrando que a altura da árvore balanceada é $\Theta(\lg(n))$

Análise de Complexidade da Inserção

- **Melhor Caso:** $\Theta(1)$ comparações e acontece, por exemplo, inserindo na raiz
- **Pior Caso:** $\Theta(n)$ comparações e acontece, por exemplo, quando inserimos os elementos na ordem crescente ou decrescente
- **Caso Médio:** $\Theta(\lg(n))$ comparações e acontece, por exemplo, quando inserimos um elemento na folha de uma árvore balanceada. Lembrando que a altura da árvore balanceada é $\Theta(\lg(n))$

Observação (1): Dependência do formato da árvore

Observação (2): Na inserção aleatória $\approx 1,39 \times \lg(n)$ comparações

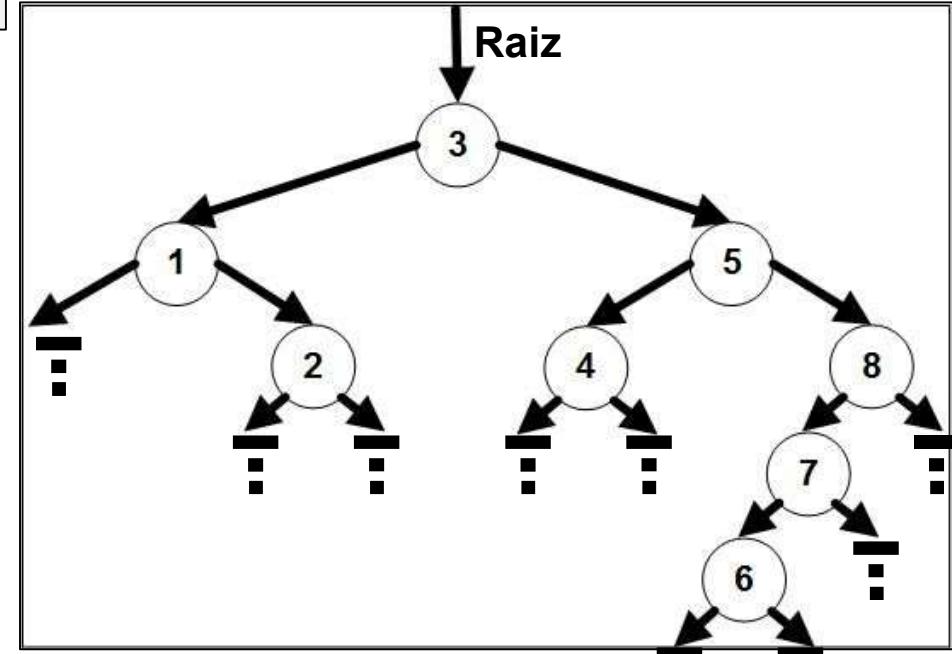
Funcionamento Básico da Pesquisa

- (1) Se a raiz estiver vazia, retornar uma **pesquisa negativa**
- (2) Senão, se o elemento procurado for **igual** ao da raiz, retornar uma **pesquisa positiva**
- (3) Senão, se o elemento procurado for **menor** que o da raiz, chamar o método de pesquisa para a subárvore da **esquerda**
- (4) Senão (elemento procurado é **maior** que o da raiz), chamar o método de pesquisa para a subárvore da **direita**

Classe Árvore Binária

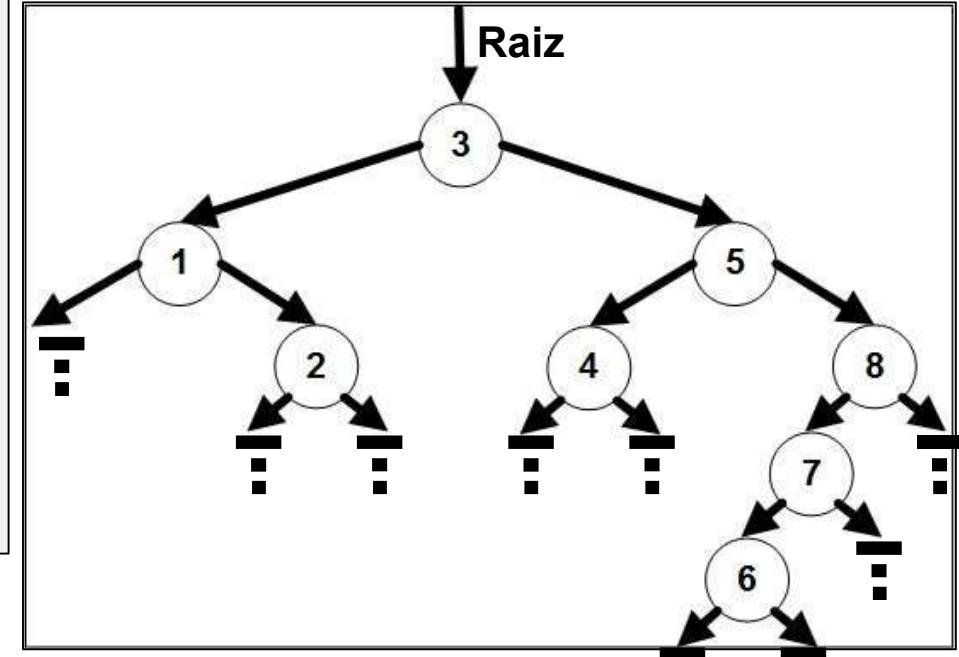
```
class ArvoreBinaria {  
    No raiz;  
    ArvoreBinaria() { raiz = null; }  
    void inserir(int x) { }  
    boolean pesquisar(int x) { }  
    void remover(int x) { }  
    void caminharCentral() { }  
    void caminharPre() { }  
    void caminharPos() { }  
}
```

Vamos pesquisar se o 4 está em nossa árvore



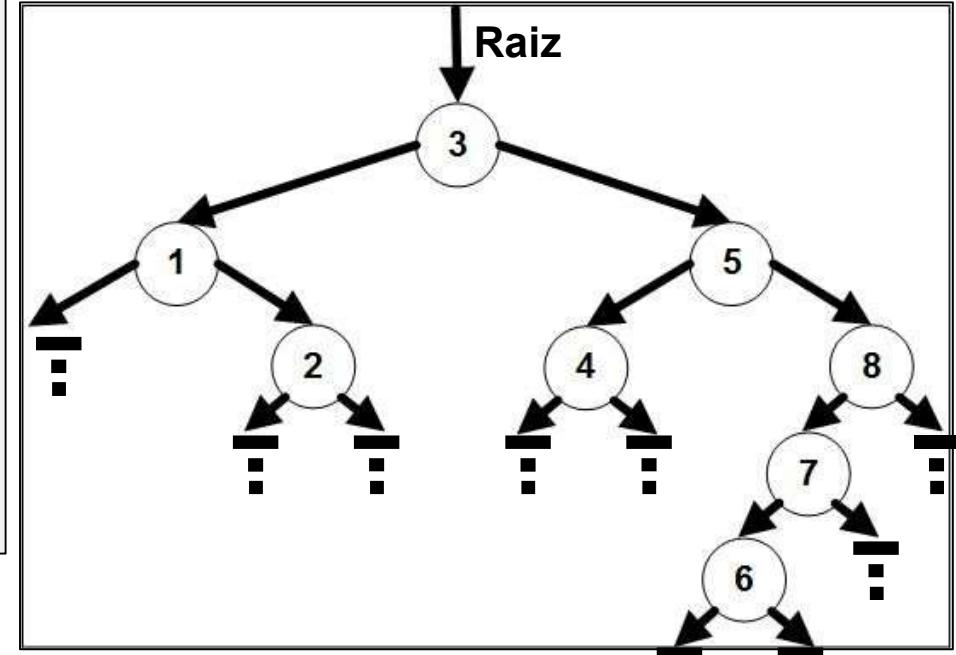
Algoritmo de Pesquisa em Java

```
//Pesquisar(4)  
  
boolean pesquisar(int x) {  
    return pesquisar(x, raiz);  
}  
  
boolean pesquisar(int x, No i) {  
    boolean resp;  
    if (i == null) {  
        resp = false;  
    } else if (x == i.elemento) {  
        resp = true;  
    } else if (x < i.elemento) {  
        resp = pesquisar(x, i.esq);  
    } else {  
        resp = pesquisar(x, i.dir);  
    }  
    return resp;  
}
```



Algoritmo de Pesquisa em Java

```
//Pesquisar(4)  
  
boolean pesquisar(int x) {  
    return pesquisar(x, raiz);  
}  
  
boolean pesquisar(int x, No i) {  
    boolean resp;  
    if (i == null) {  
        resp = false;  
    } else if (x == i.elemento) {  
        resp = true;  
    } else if (x < i.elemento) {  
        resp = pesquisar(x, i.esq);  
    } else {  
        resp = pesquisar(x, i.dir);  
    }  
    return resp;  
}
```

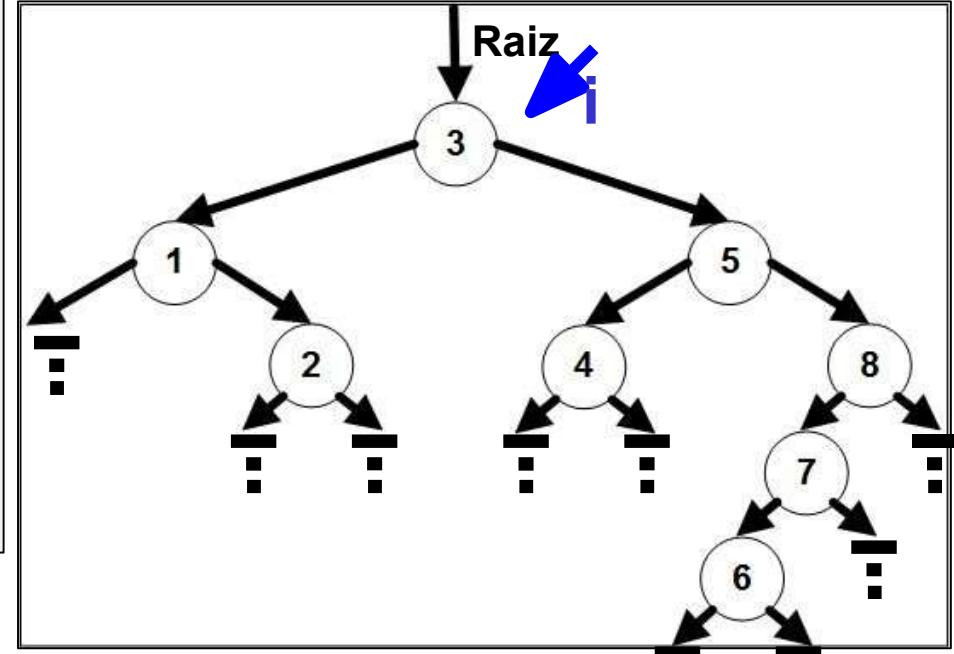


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}
```

```
boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

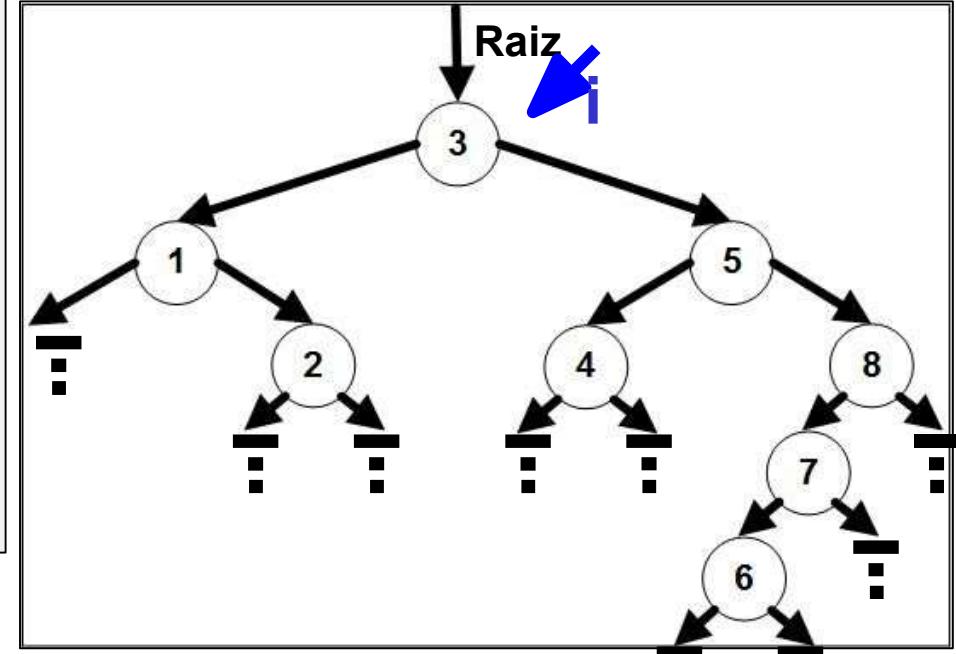


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

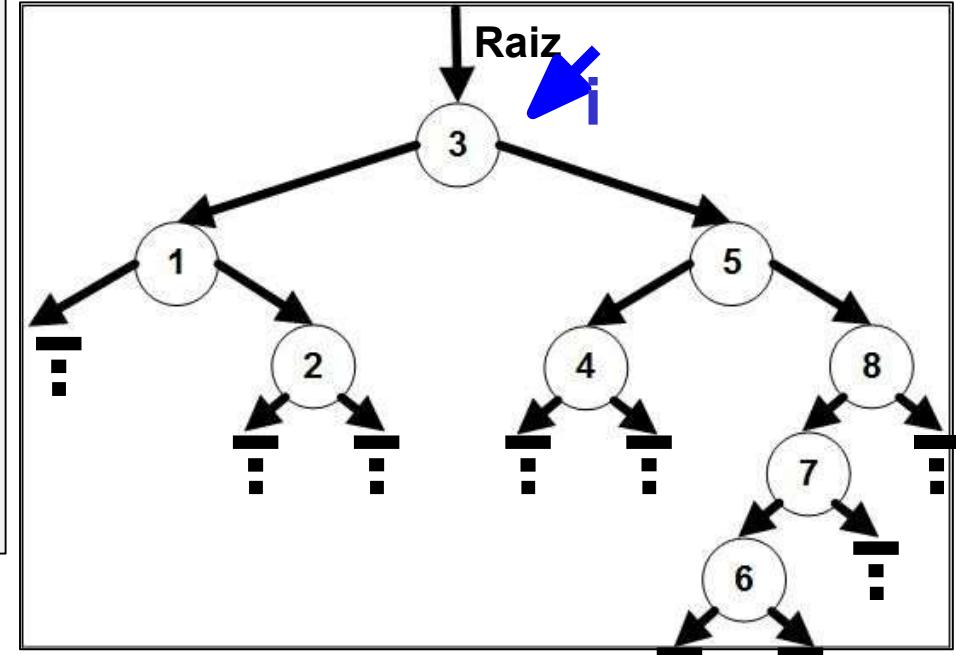


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)

boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
false: n(3) == null
```

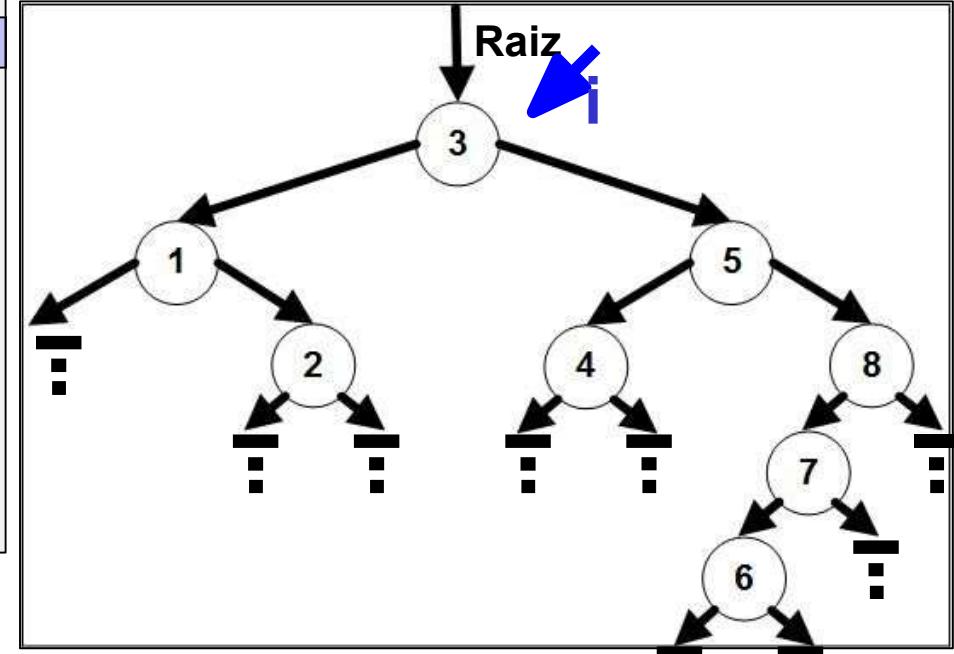


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)

boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
false: 4 == 3
```



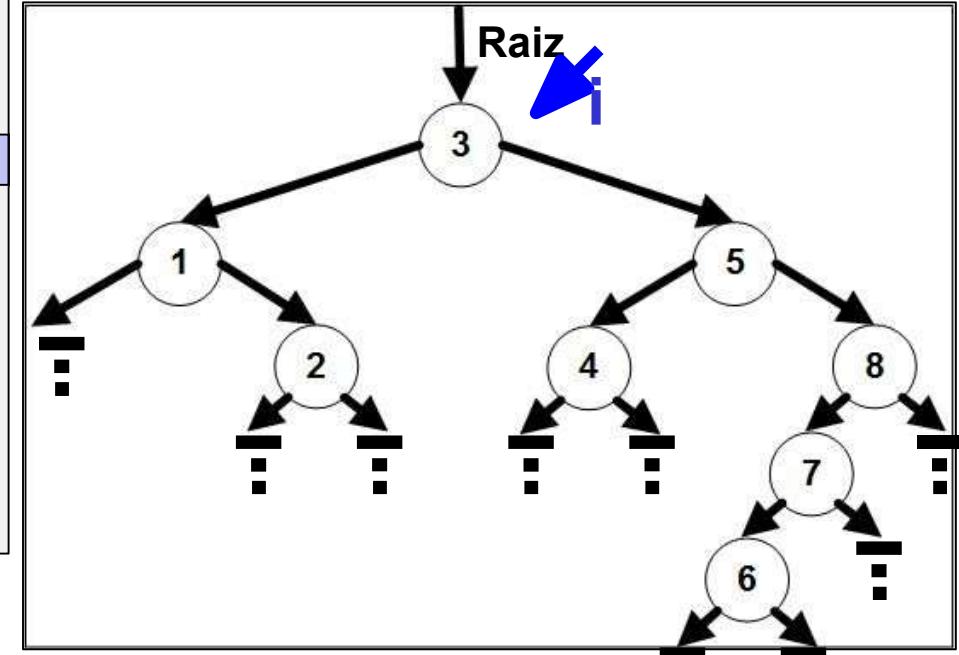
Algoritmo de Pesquisa em Java

```
//Pesquisar(4)

boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

false: 4 < 3

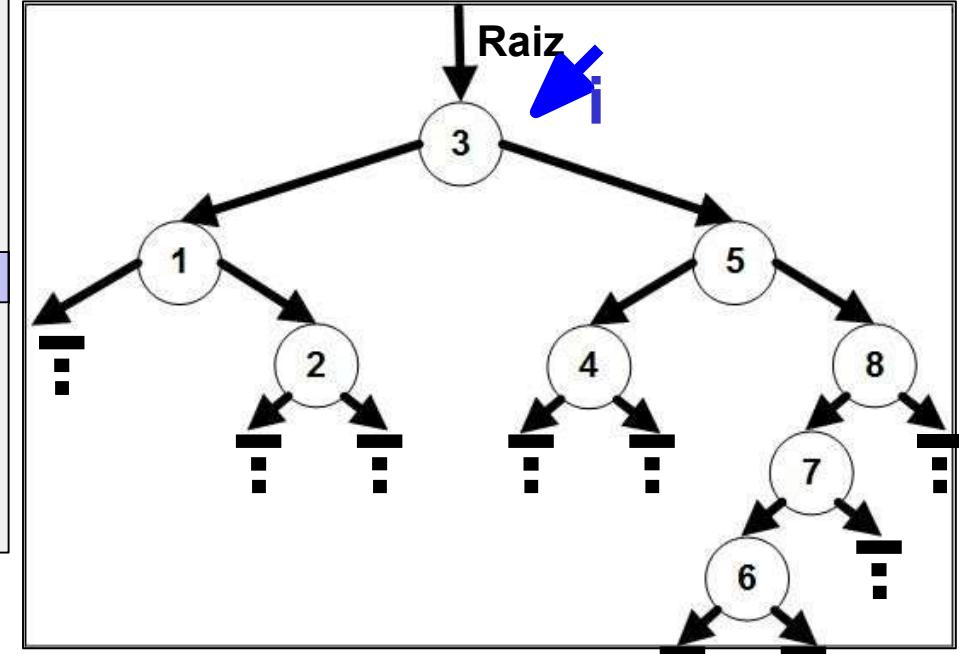


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

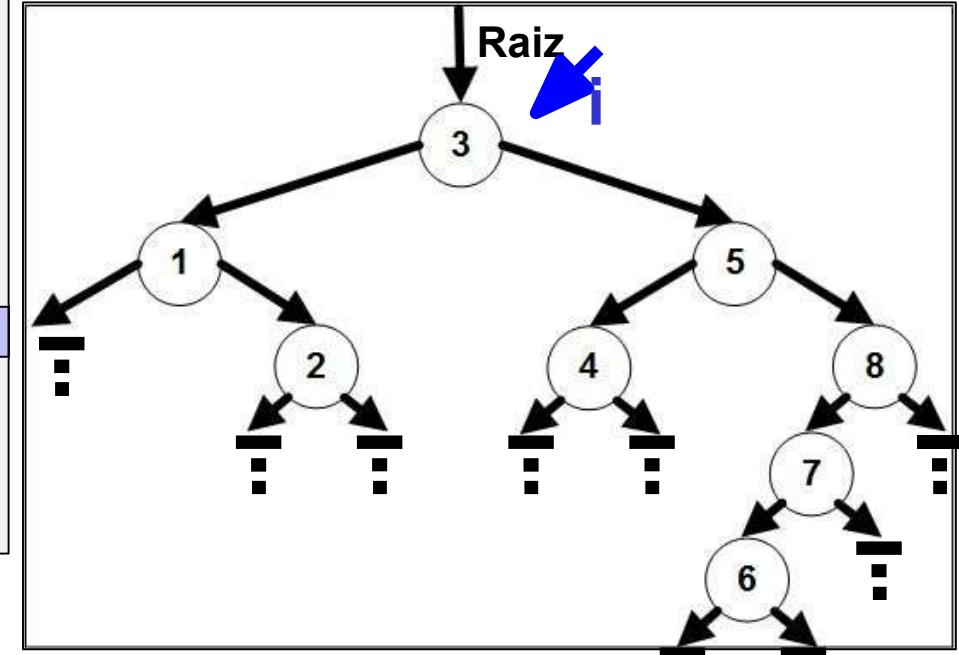


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

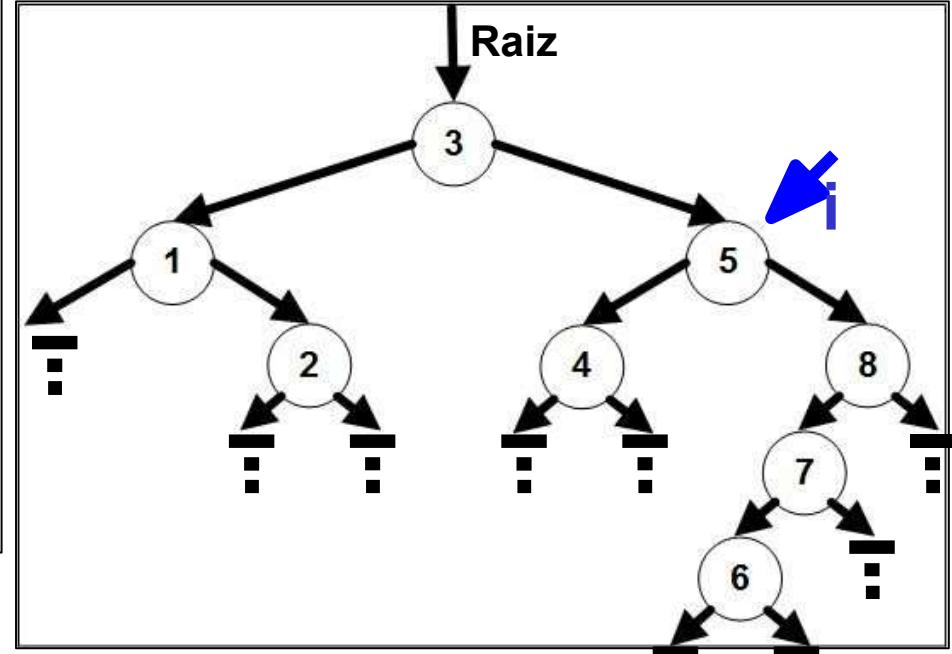


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}
```

```
boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

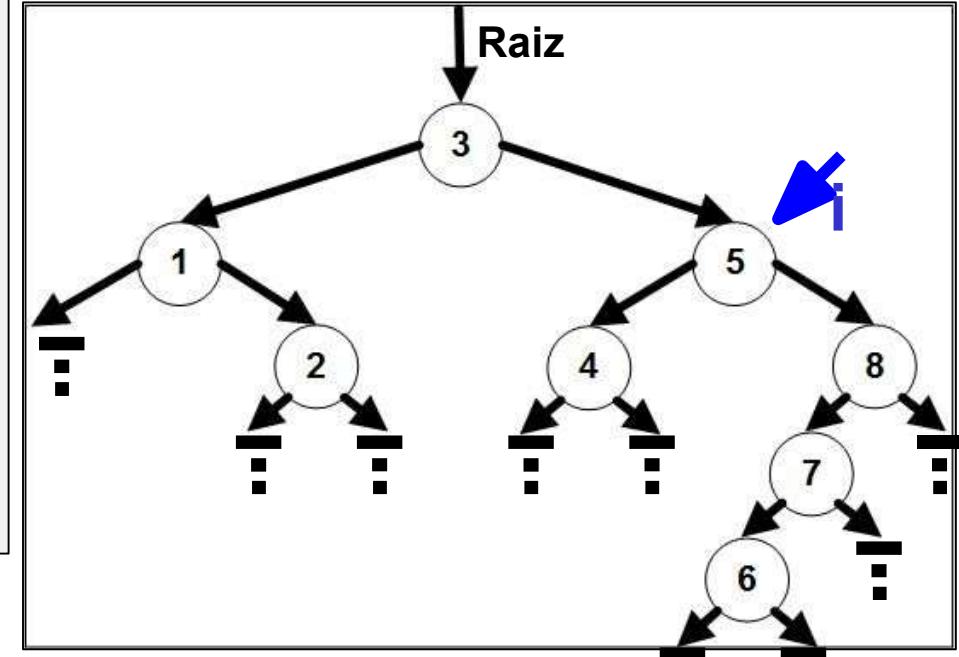


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)

boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

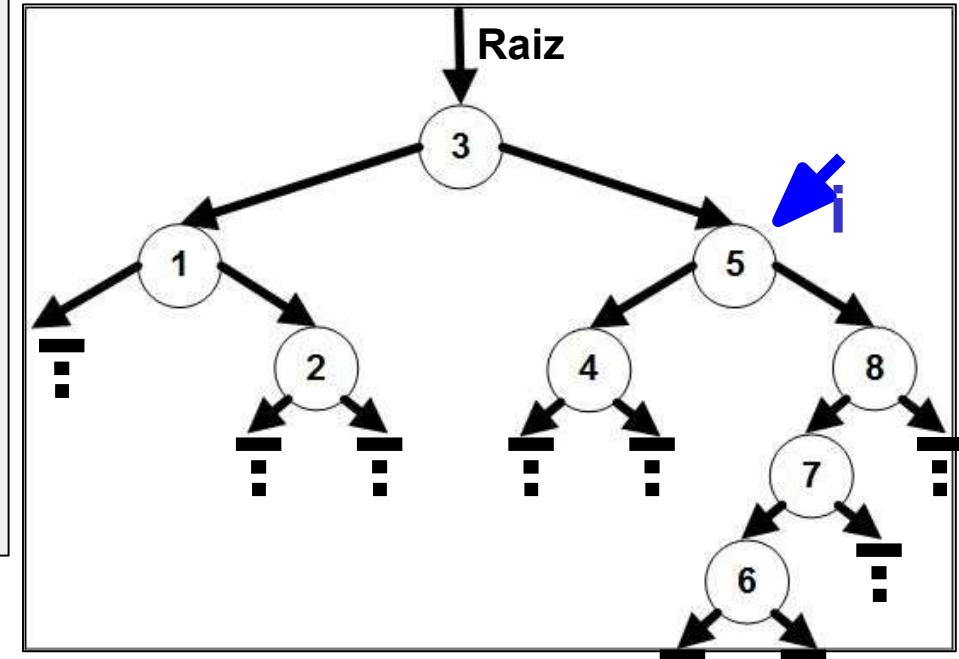


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)

boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
false: n(5) == null
```



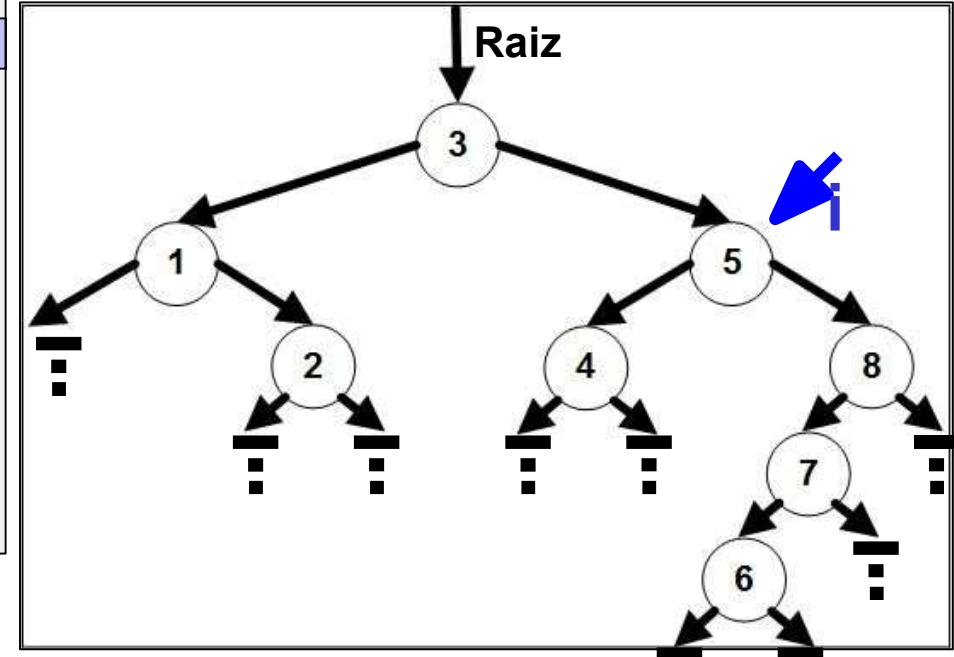
Algoritmo de Pesquisa em Java

```
//Pesquisar(4)

boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

false: 4 == 5



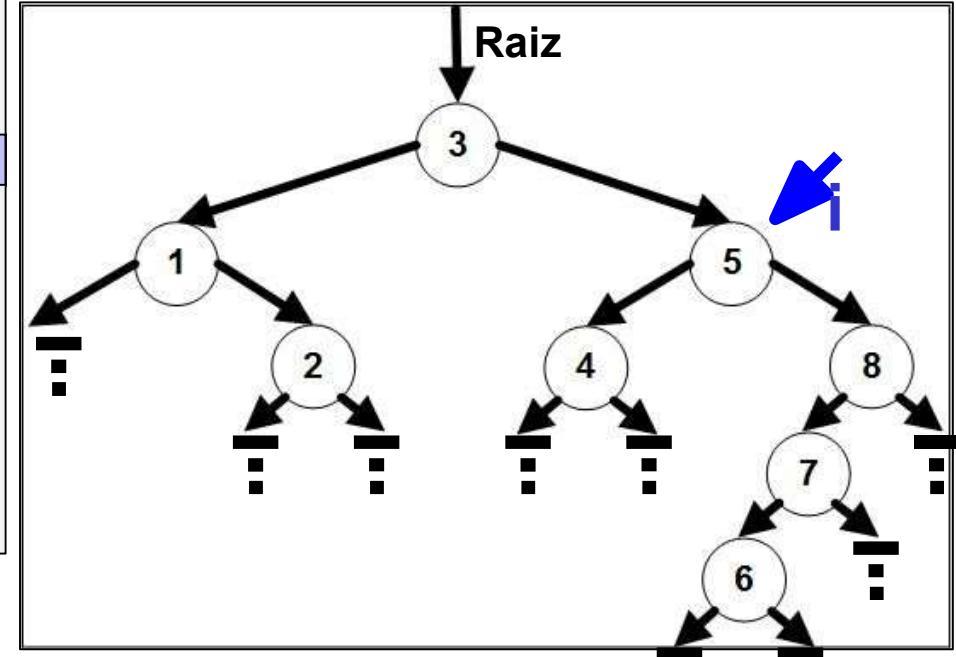
Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

true: 4 < 5

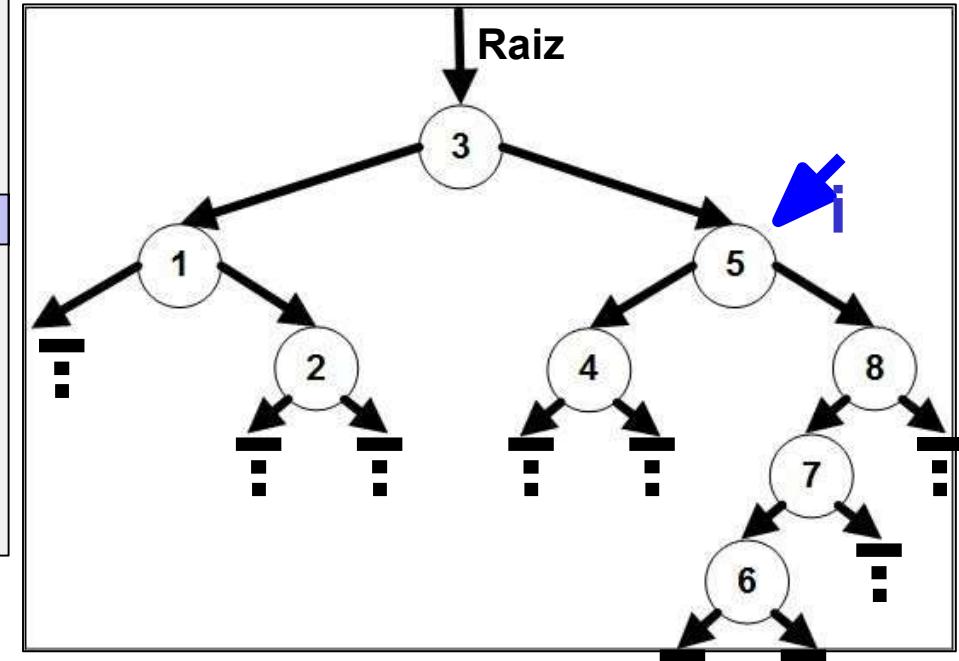


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)

boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

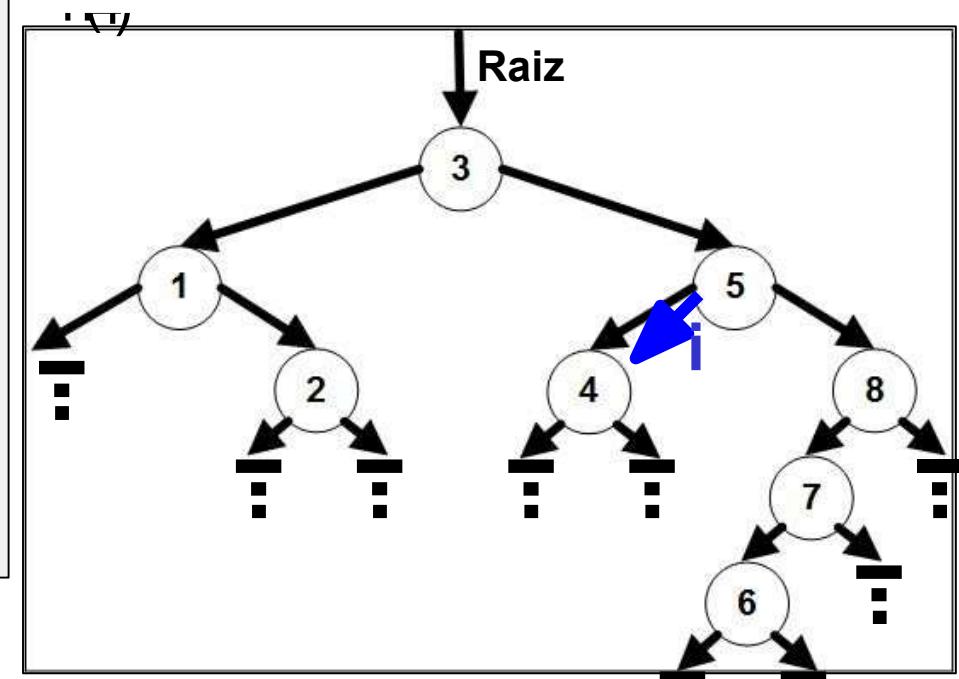


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}
```

```
boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

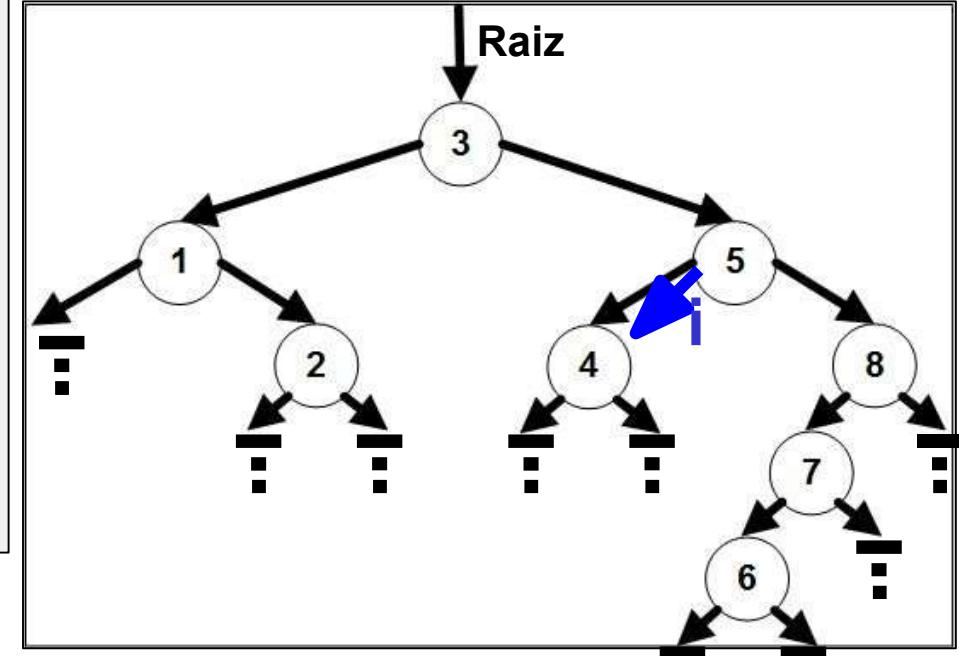


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

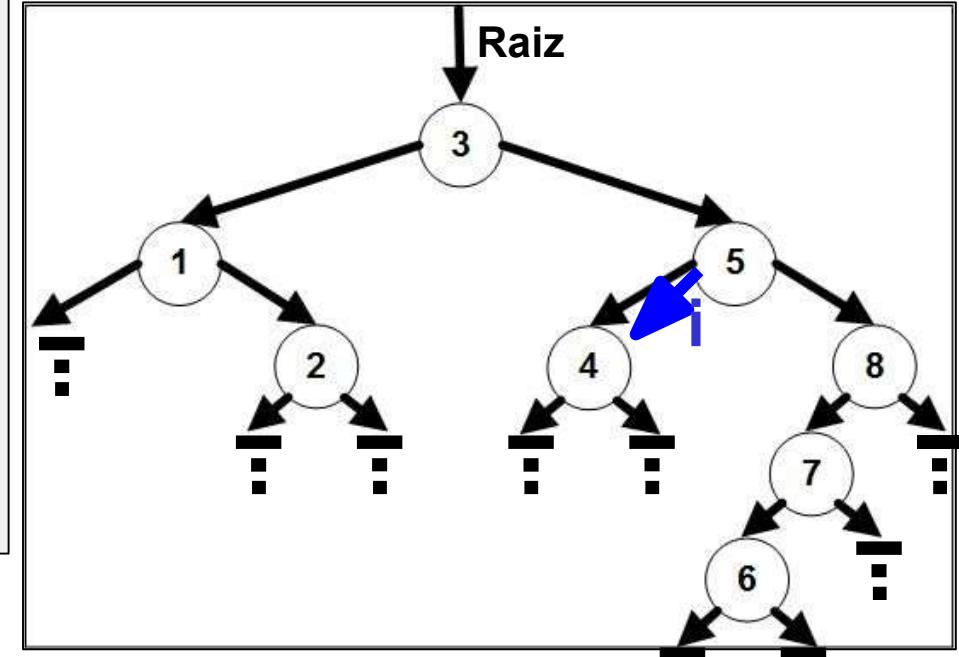


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)

boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
false: n(4) == null
```



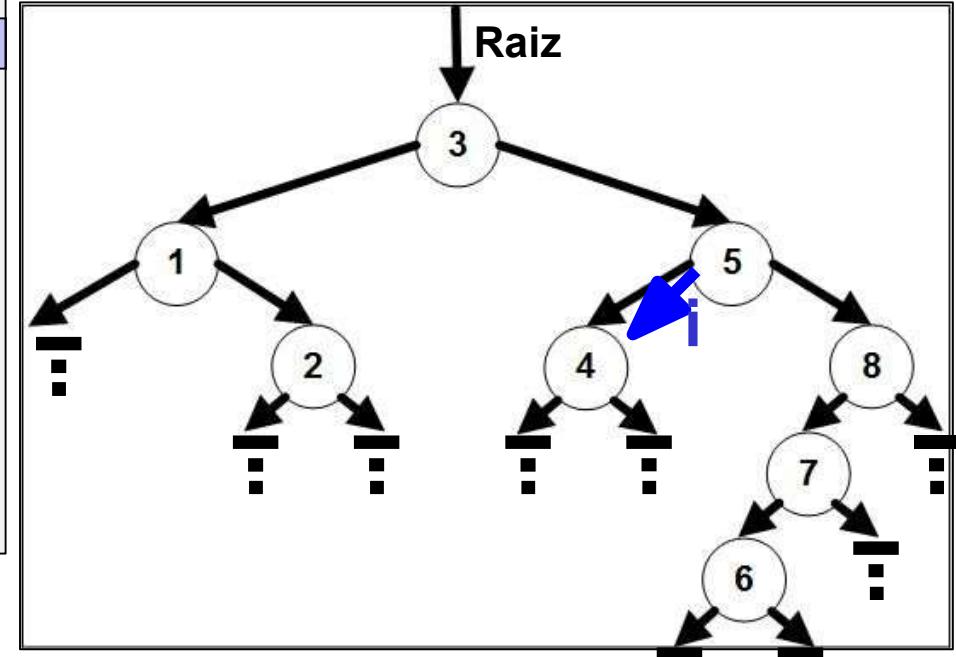
Algoritmo de Pesquisa em Java

```
//Pesquisar(4)

boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

true: 4 == 4

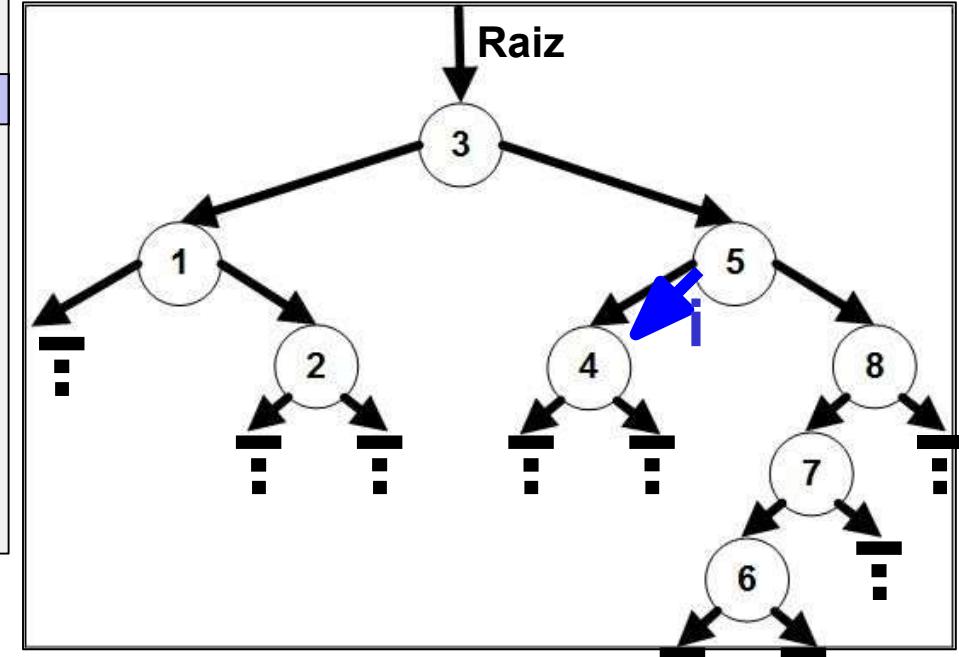


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

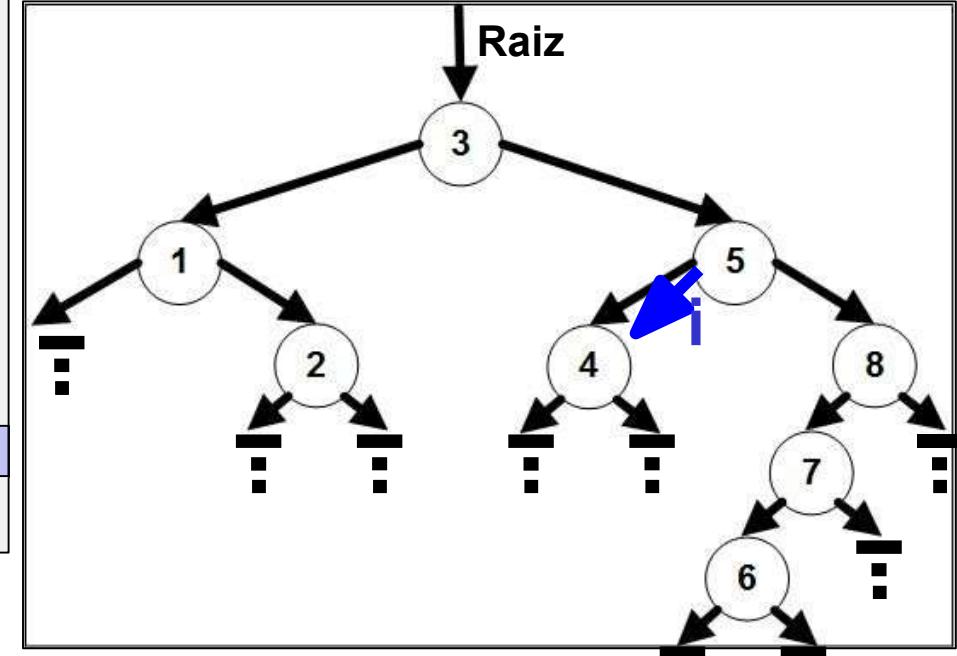


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

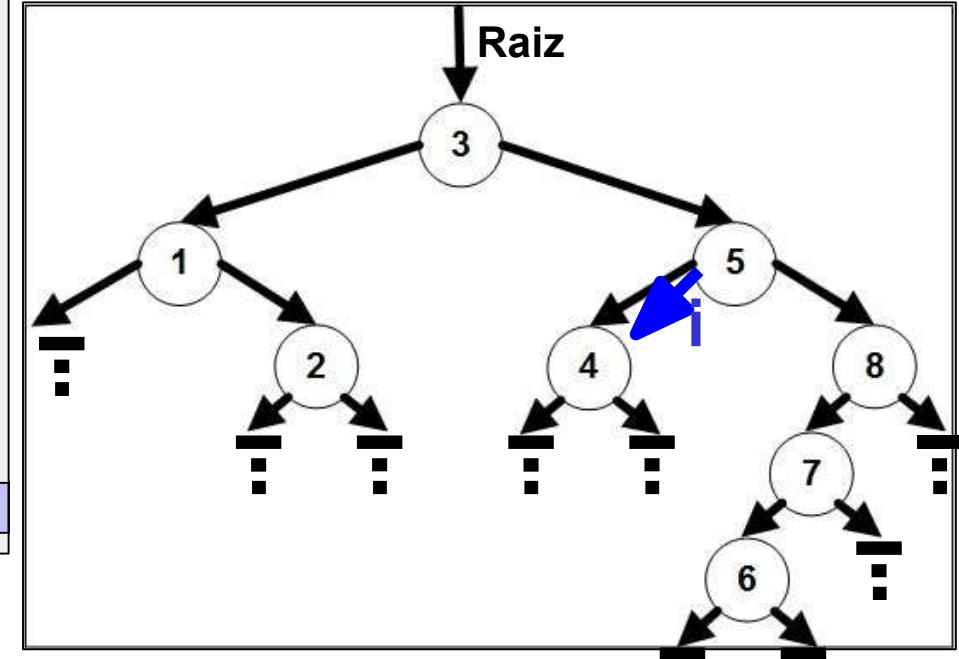


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)

boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

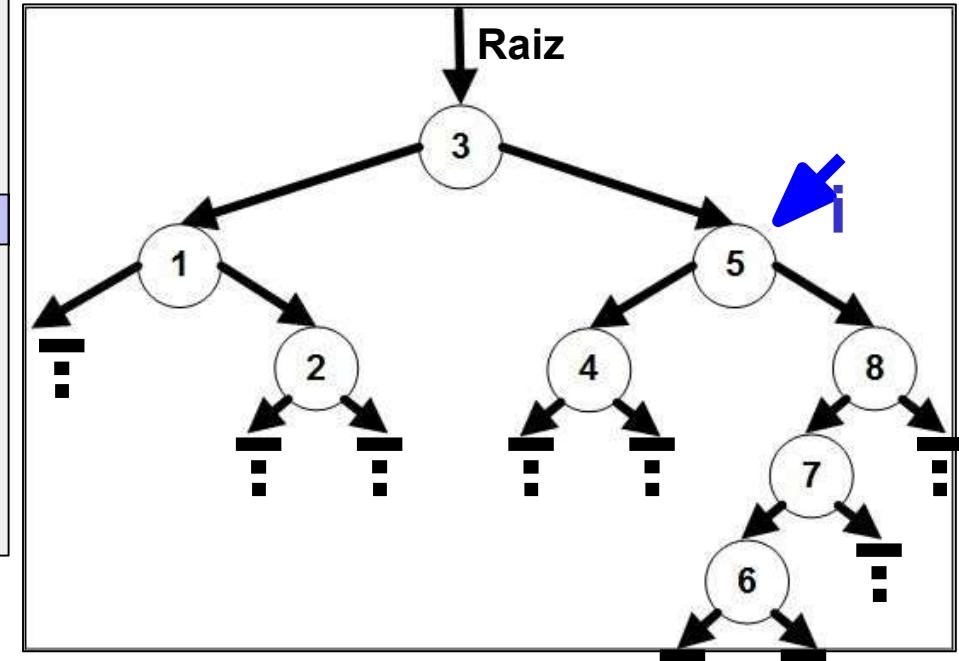


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)

boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

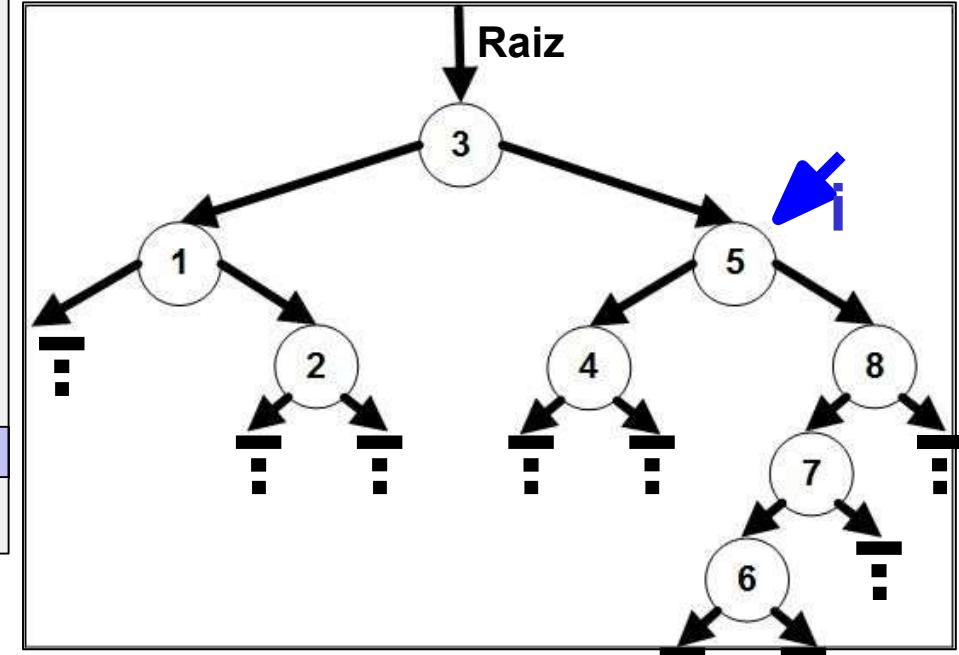


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

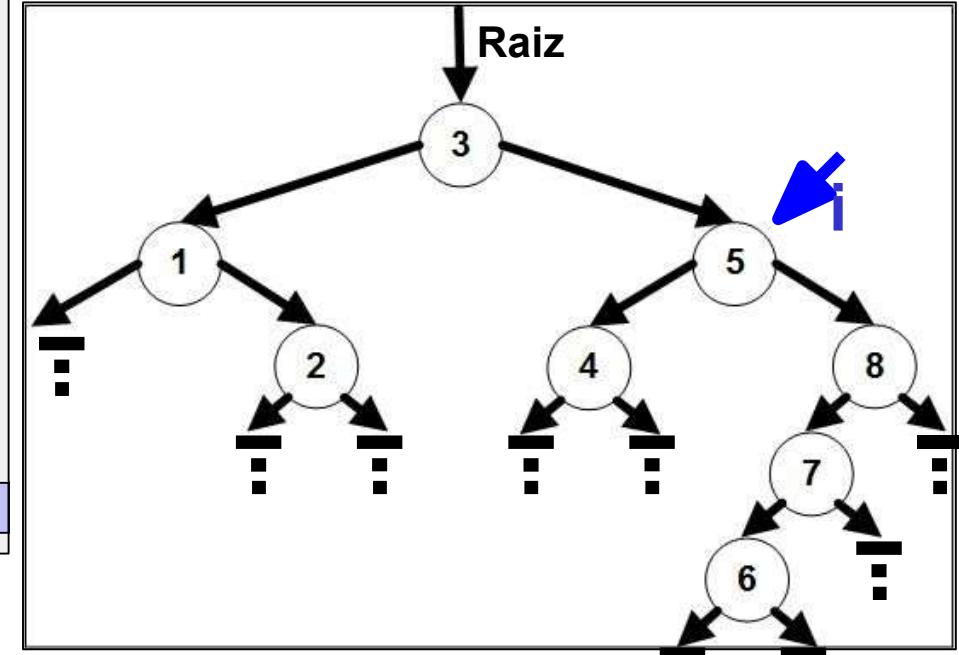
```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```



Algoritmo de Pesquisa em Java

```
//Pesquisar(4)  
  
boolean pesquisar(int x) {  
    return pesquisar(x, raiz);  
}  
  
boolean pesquisar(int x, No i) {  
    boolean resp;  
    if (i == null) {  
        resp = false;  
    } else if (x == i.elemento) {  
        resp = true;  
    } else if (x < i.elemento) {  
        resp = pesquisar(x, i.esq);  
    } else {  
        resp = pesquisar(x, i.dir);  
    }  
    return resp;  
}
```

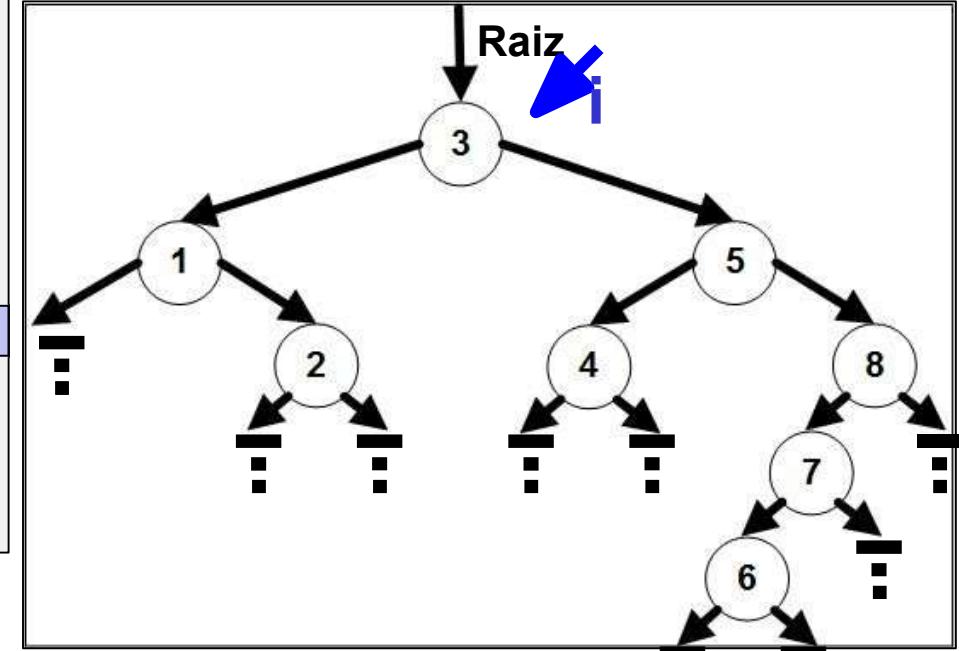


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

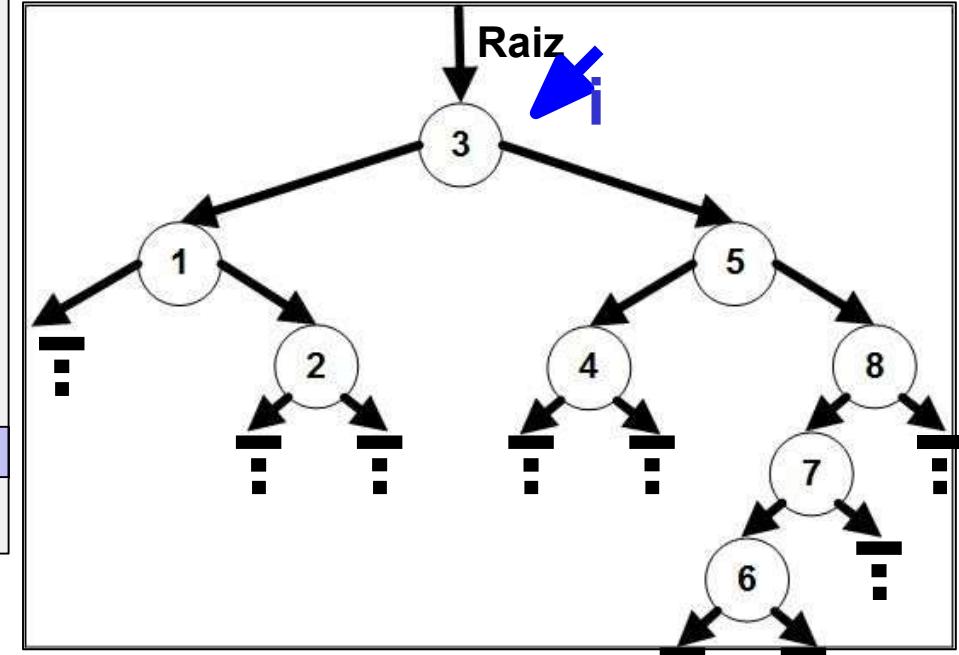


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```

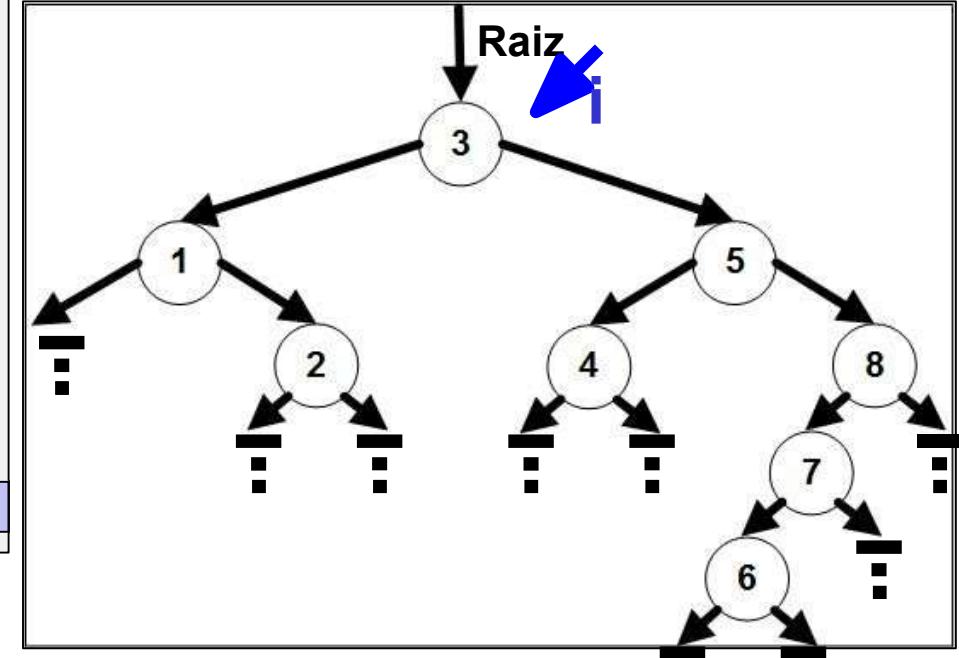


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

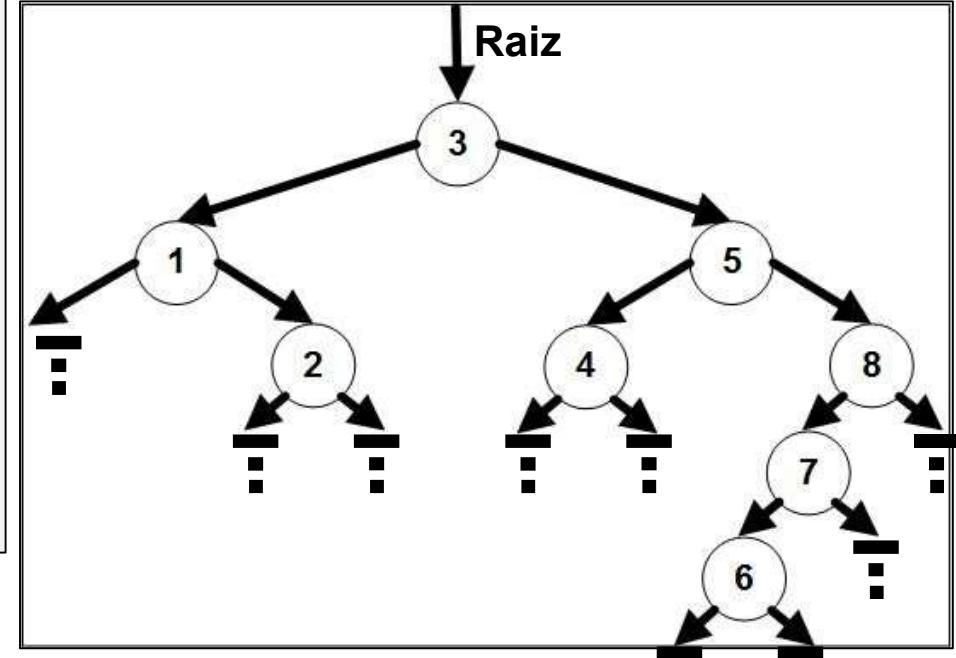
```
boolean pesquisar(int x) {
    return pesquisar(x, raiz);
}

boolean pesquisar(int x, No i) {
    boolean resp;
    if (i == null) {
        resp = false;
    } else if (x == i.elemento) {
        resp = true;
    } else if (x < i.elemento) {
        resp = pesquisar(x, i.esq);
    } else {
        resp = pesquisar(x, i.dir);
    }
    return resp;
}
```



Algoritmo de Pesquisa em Java

```
//Pesquisar(4)  
  
boolean pesquisar(int x) {  
    return pesquisar(x, raiz);  
}  
    retorna true  
  
boolean pesquisar(int x, No i) {  
    boolean resp;  
    if (i == null) {  
        resp = false;  
    } else if (x == i.elemento) {  
        resp = true;  
    } else if (x < i.elemento) {  
        resp = pesquisar(x, i.esq);  
    } else {  
        resp = pesquisar(x, i.dir);  
    }  
    return resp;  
}
```

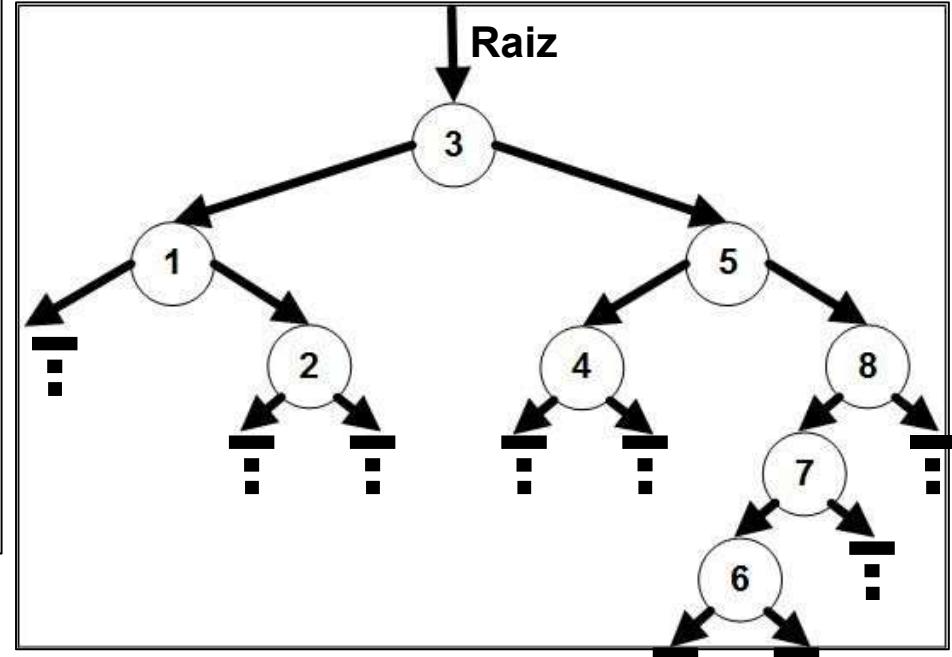


Algoritmo de Pesquisa em Java

```
//Pesquisar(4)
```

```
boolean pesquisar(int x) {  
    return pesquisar(x, raiz);  
}
```

```
boolean pesquisar(int x, No i) {  
    boolean resp;  
    if (i == null) {  
        resp = false;  
    } else if (x == i.elemento) {  
        resp = true;  
    } else if (x < i.elemento) {  
        resp = pesquisar(x, i.esq);  
    } else {  
        resp = pesquisar(x, i.dir);  
    }  
    return resp;  
}
```



Análise de complexidade da Pesquisa

- **Melhor Caso:** $\Theta(1)$ comparações e acontece, por exemplo, na raiz
- **Pior Caso:** $\Theta(n)$ comparações e acontece, por exemplo, quando inserimos os elementos em ordem e o elemento desejado está na folha
- **Caso Médio:** $\Theta(\lg(n))$ comparações e acontece, por exemplo, quando a árvore está balanceada e procuramos um elemento localizado em uma das folhas

Caminhamento

- Consiste em “caminhar” por todos os nós da árvore
- Também chamado de percorrer, buscar, visitar, mostrar,
- Análise de complexidade: $\Theta(n)$ visitas



Alguns Caminhamentos

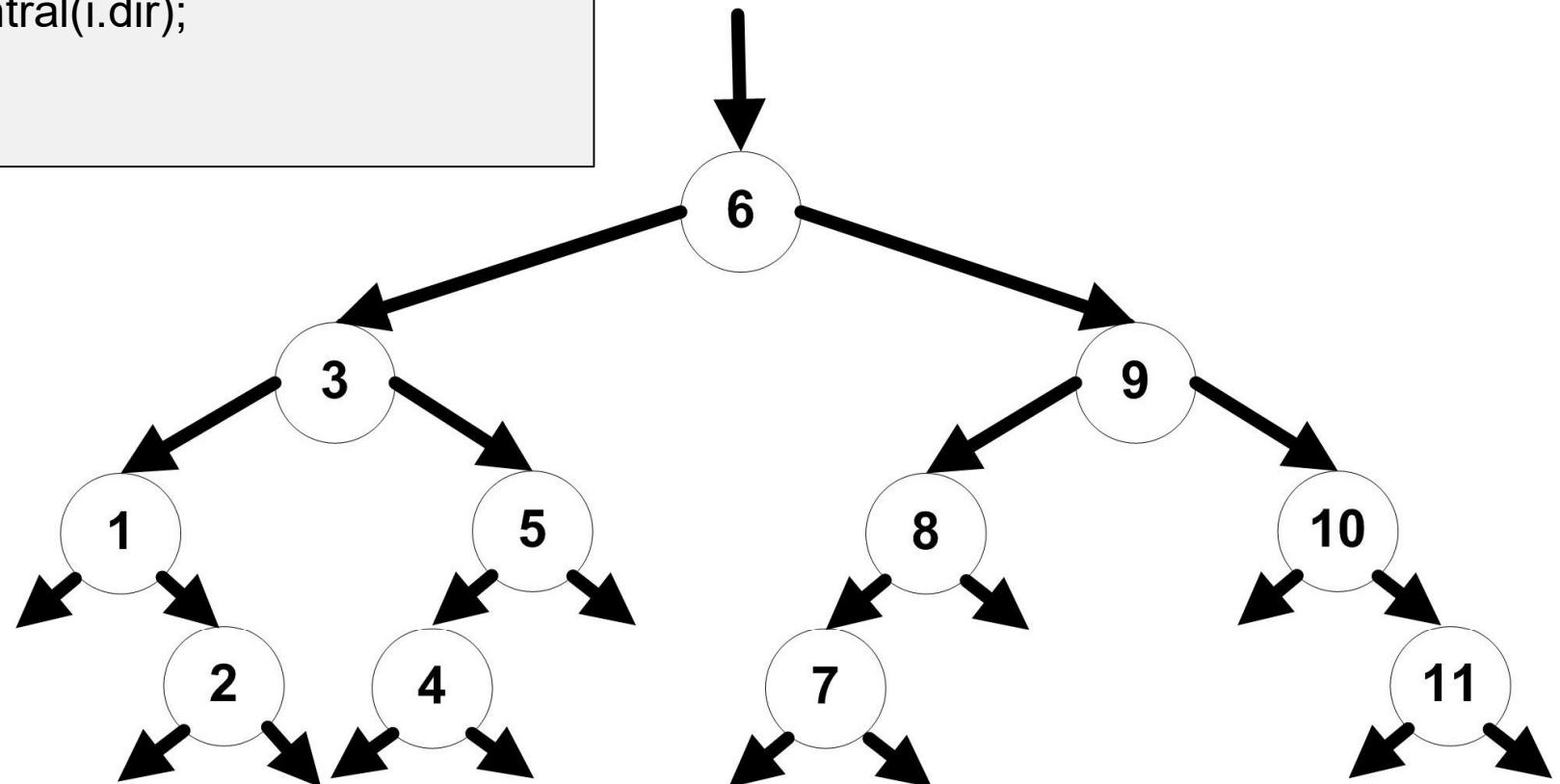
```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}  
  
//central ou em ordem
```

```
void caminharPre(No i) {  
    if (i != null) {  
        System.out.print(i.elemento + " ");  
        caminharPre(i.esq);  
        caminharPre(i.dir);  
    }  
}  
  
//pré-fixado ou pré-ordem
```

```
void caminharPos(No i) {  
    if (i != null) {  
        caminharPos(i.esq);  
        caminharPos(i.dir);  
        System.out.print(i.elemento + " ");  
    }  
}  
  
//pós-fixado ou pós-ordem
```

Caminhamento Central ou Em Ordem

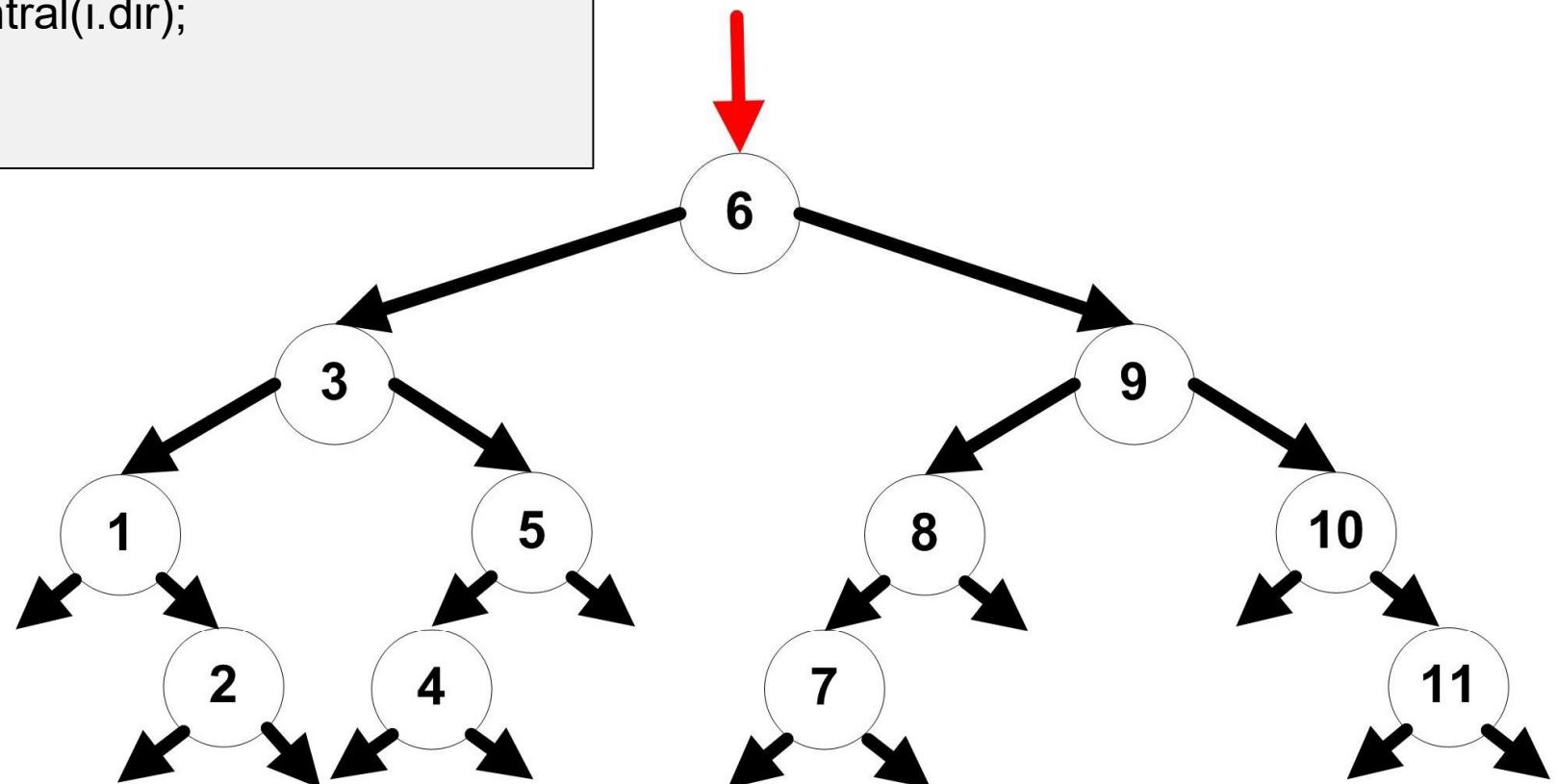
```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Tela

Caminhamento Central ou Em Ordem

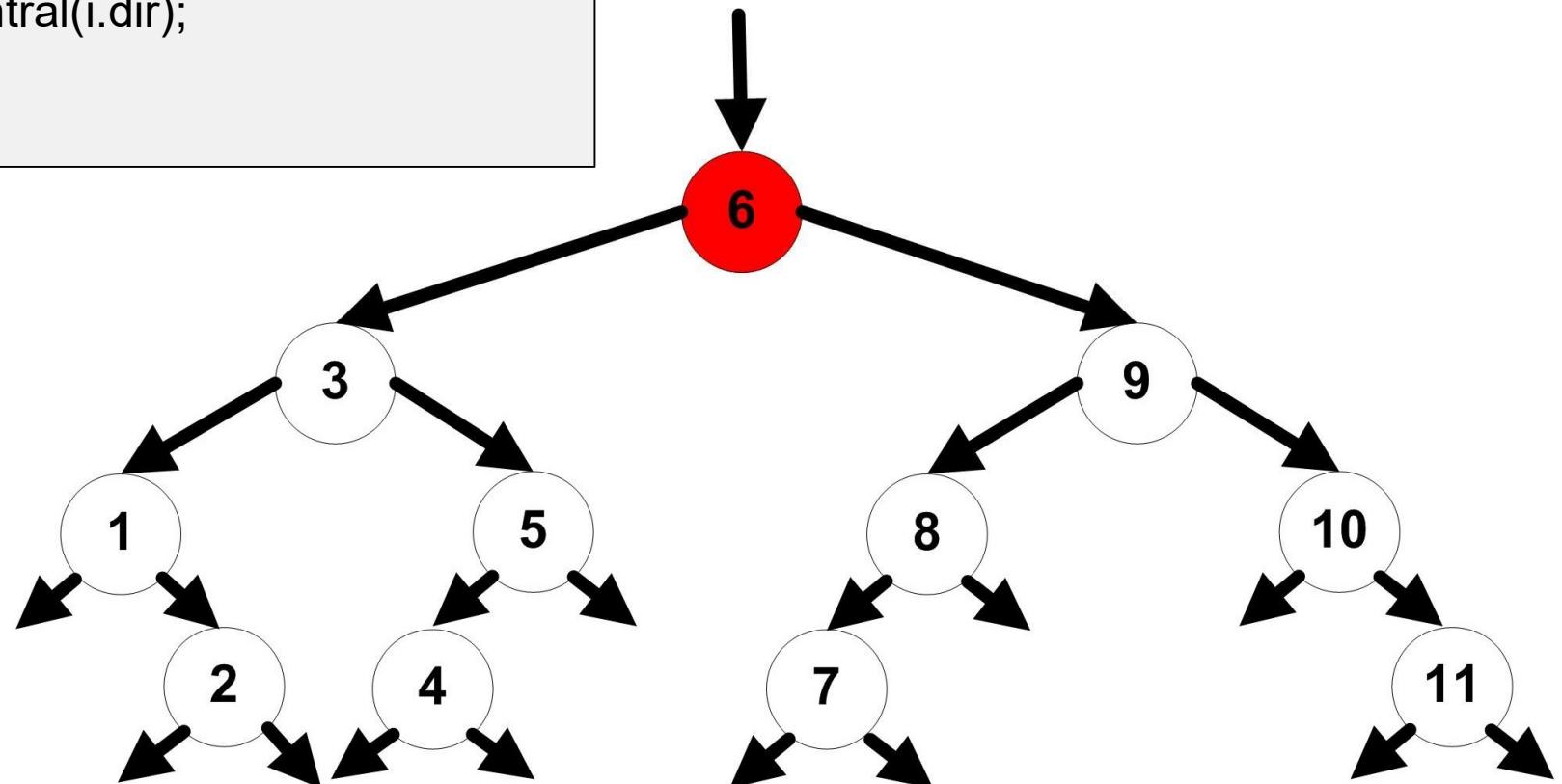
```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Tela

Caminhamento Central ou Em Ordem

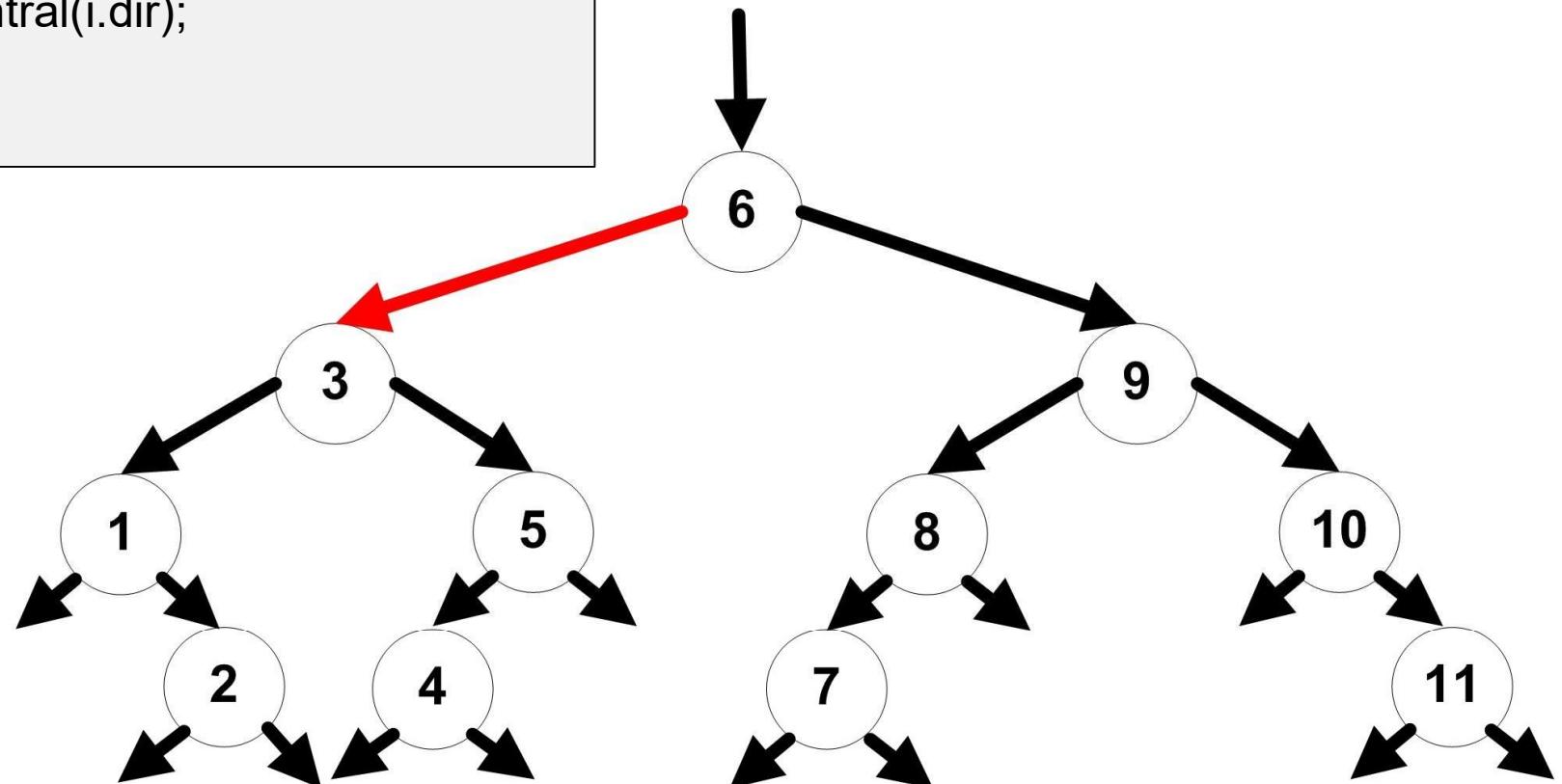
```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Tela

Caminhamento Central ou Em Ordem

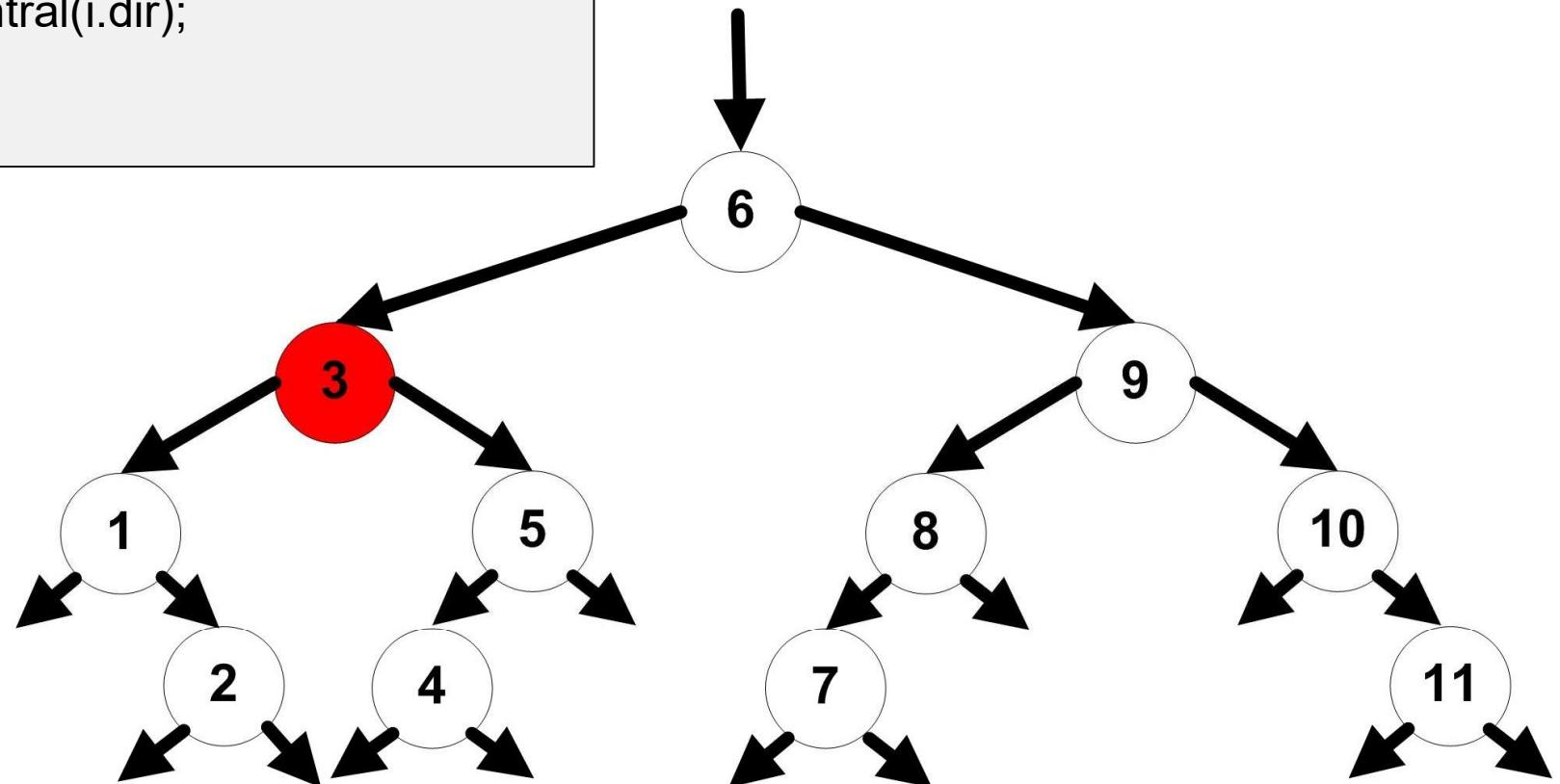
```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Tela

Caminhamento Central ou Em Ordem

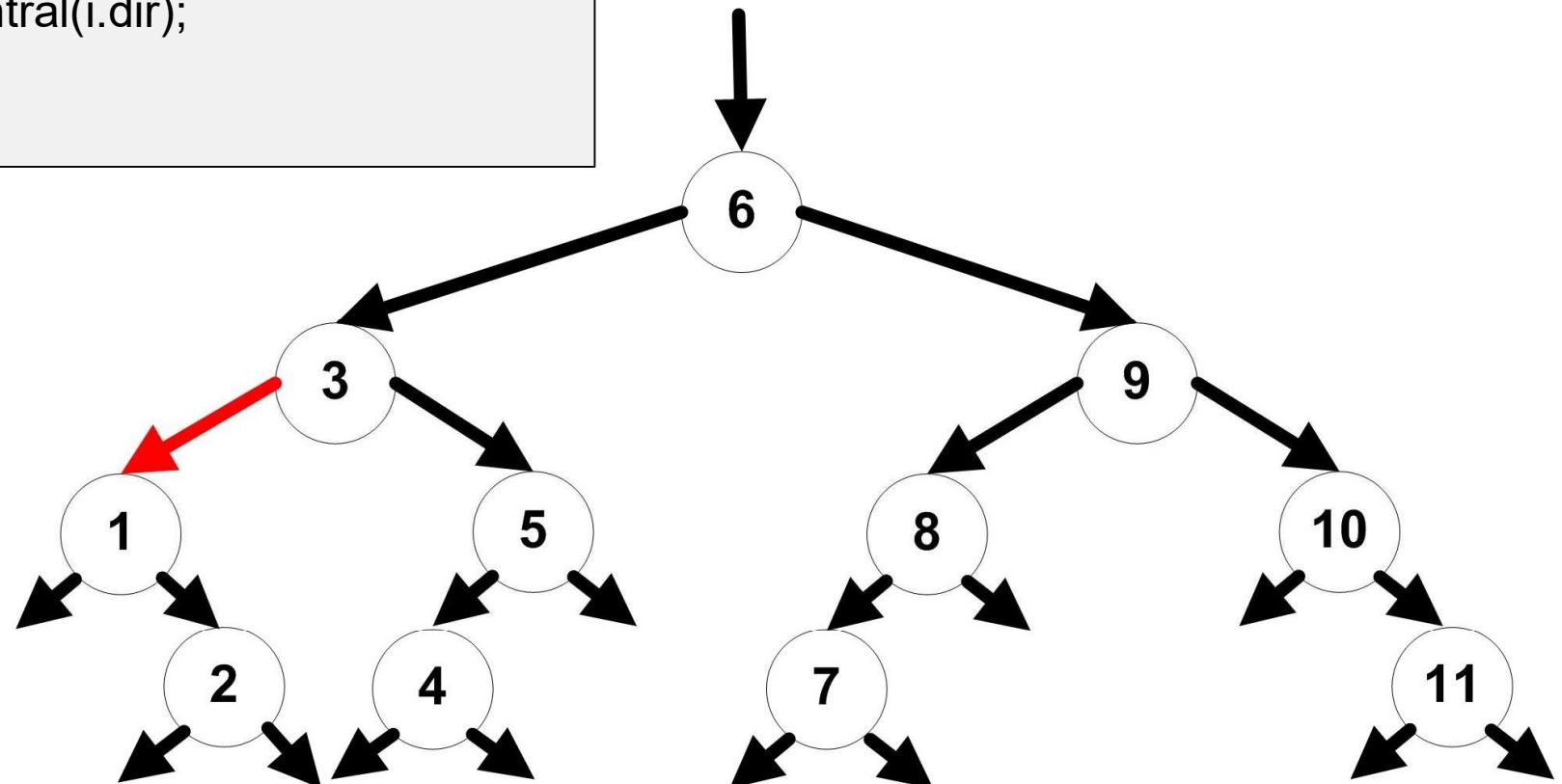
```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Tela

Caminhamento Central ou Em Ordem

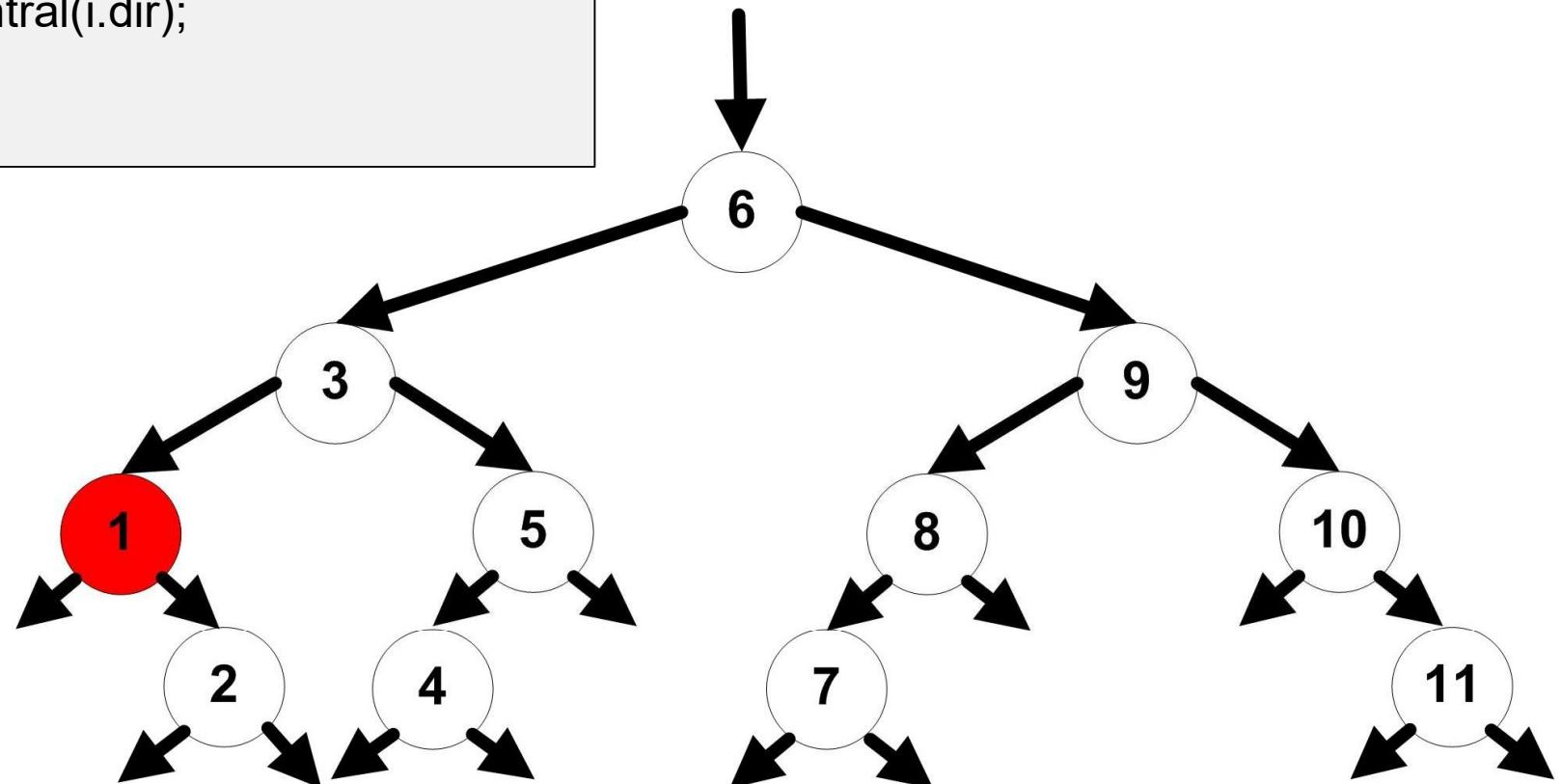
```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Tela

Caminhamento Central ou Em Ordem

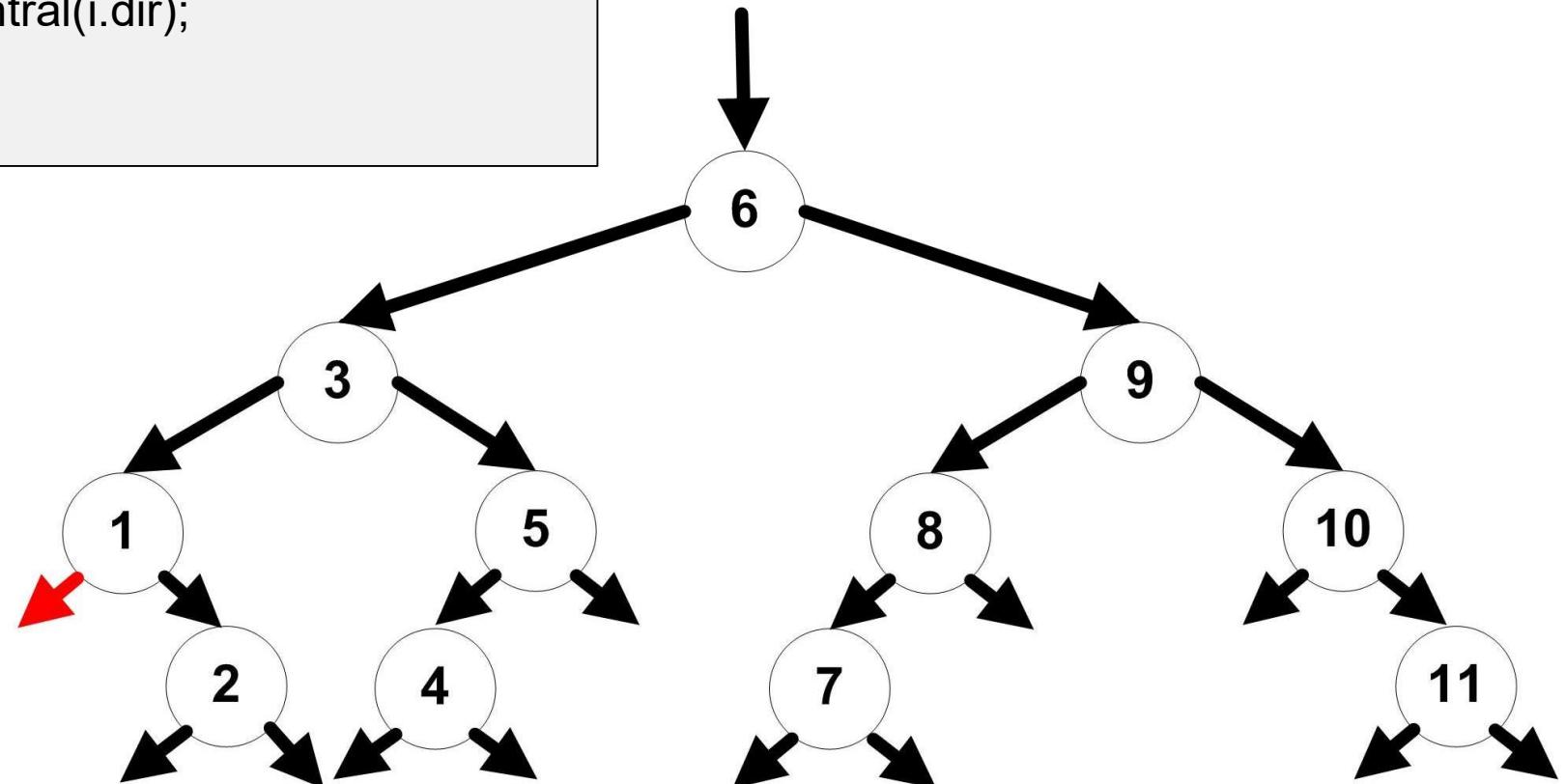
```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Tela

Caminhamento Central ou Em Ordem

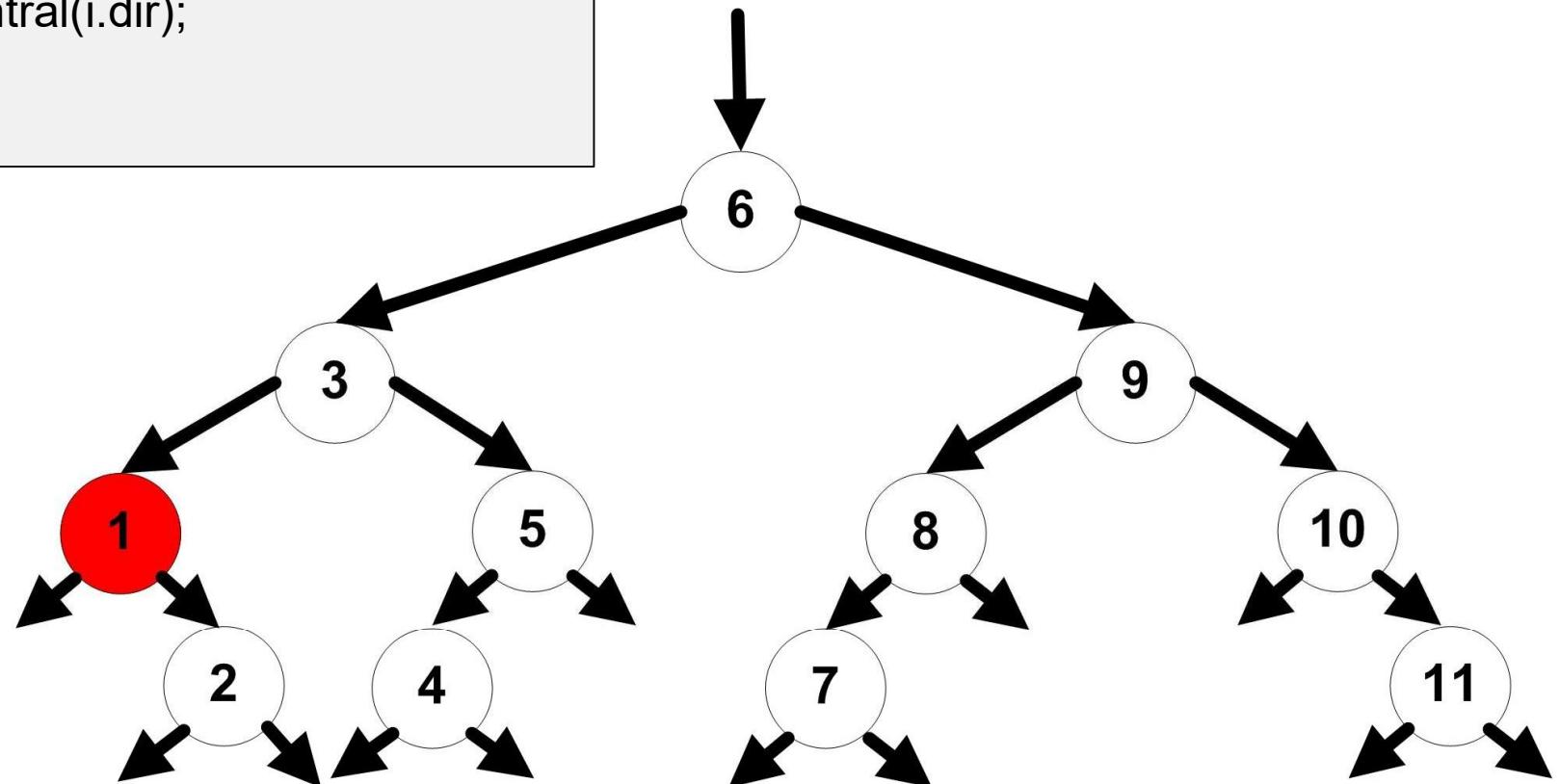
```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Tela

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

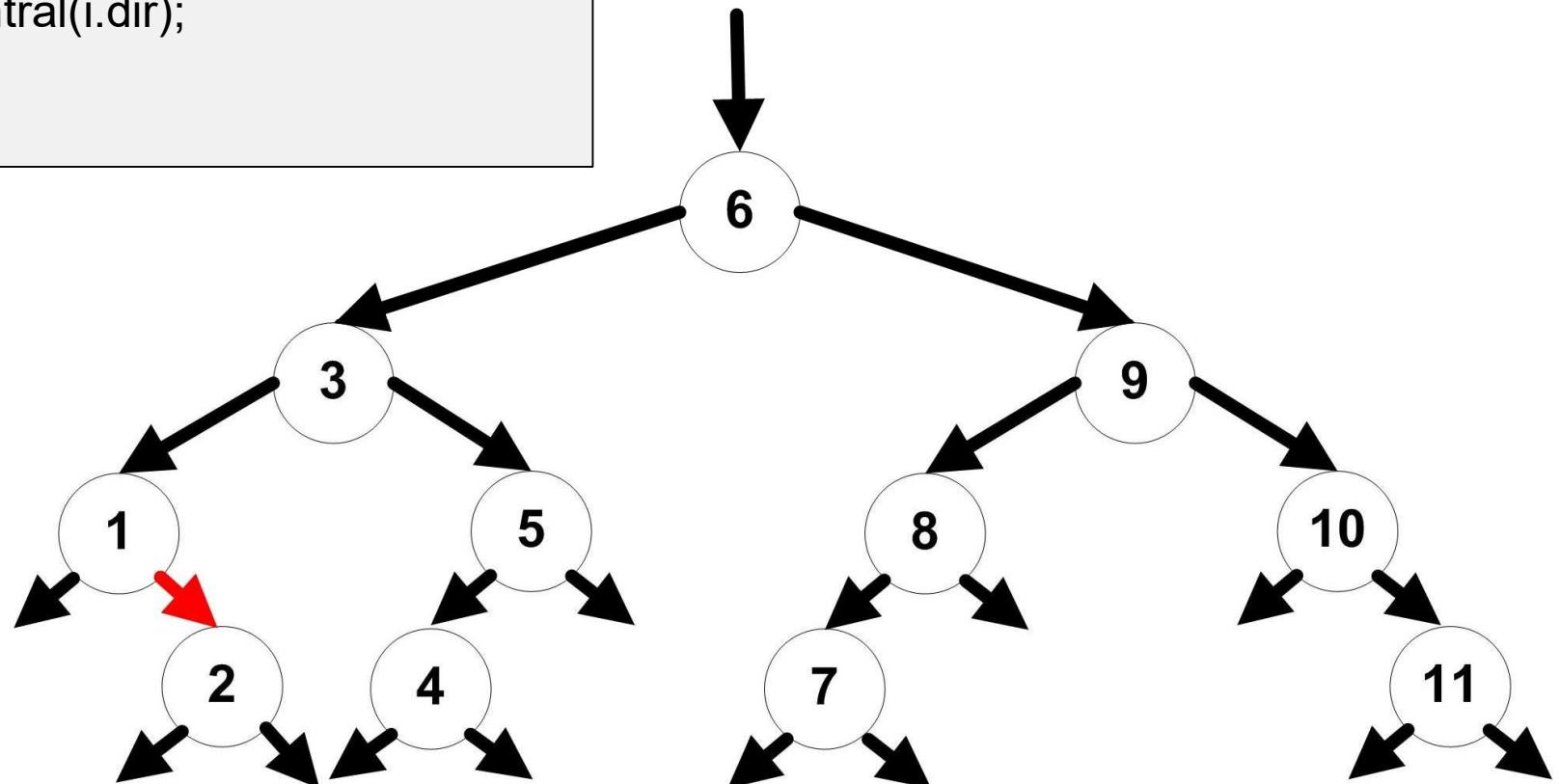


Tela

1

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

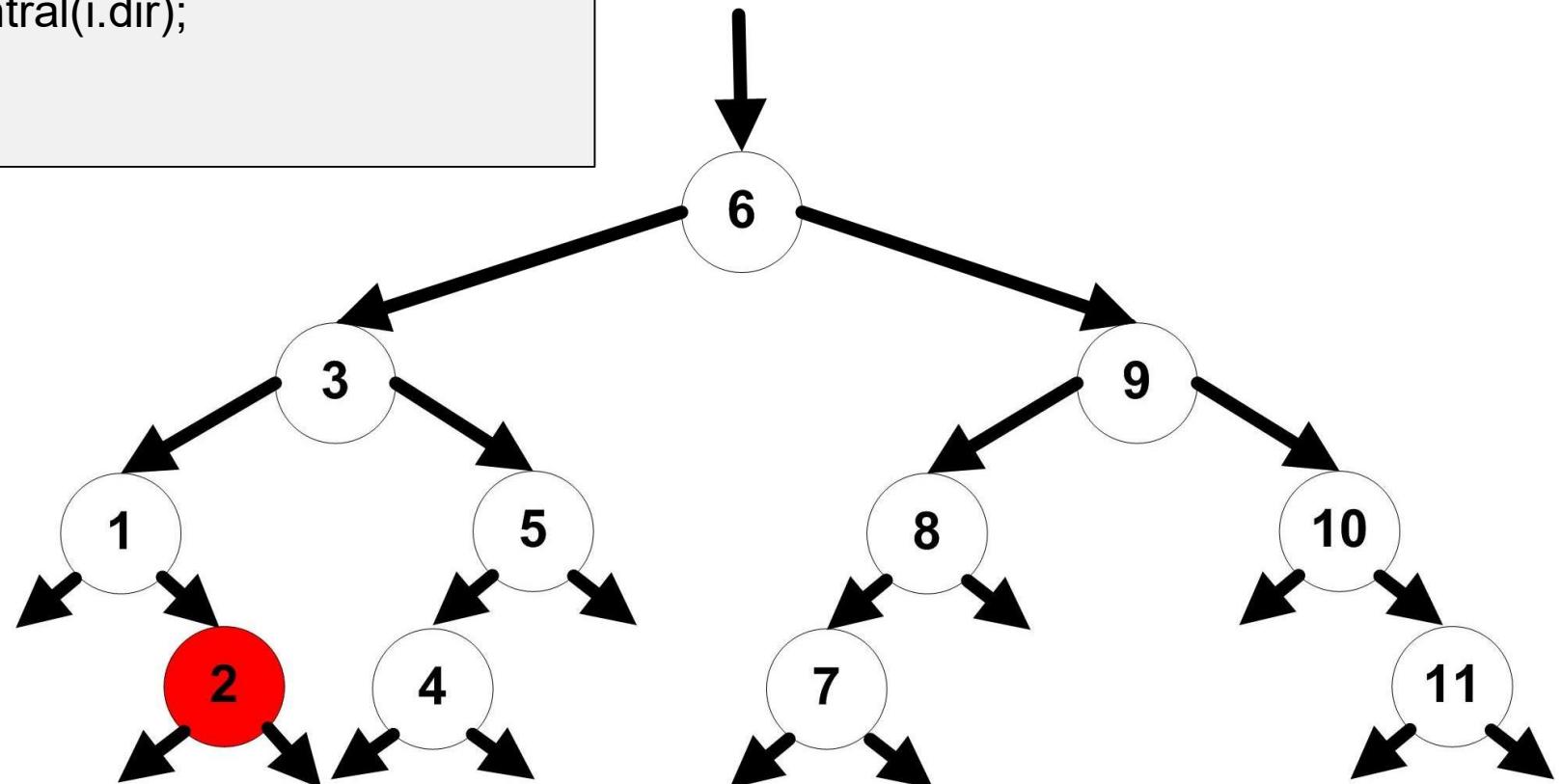


Tela

1

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

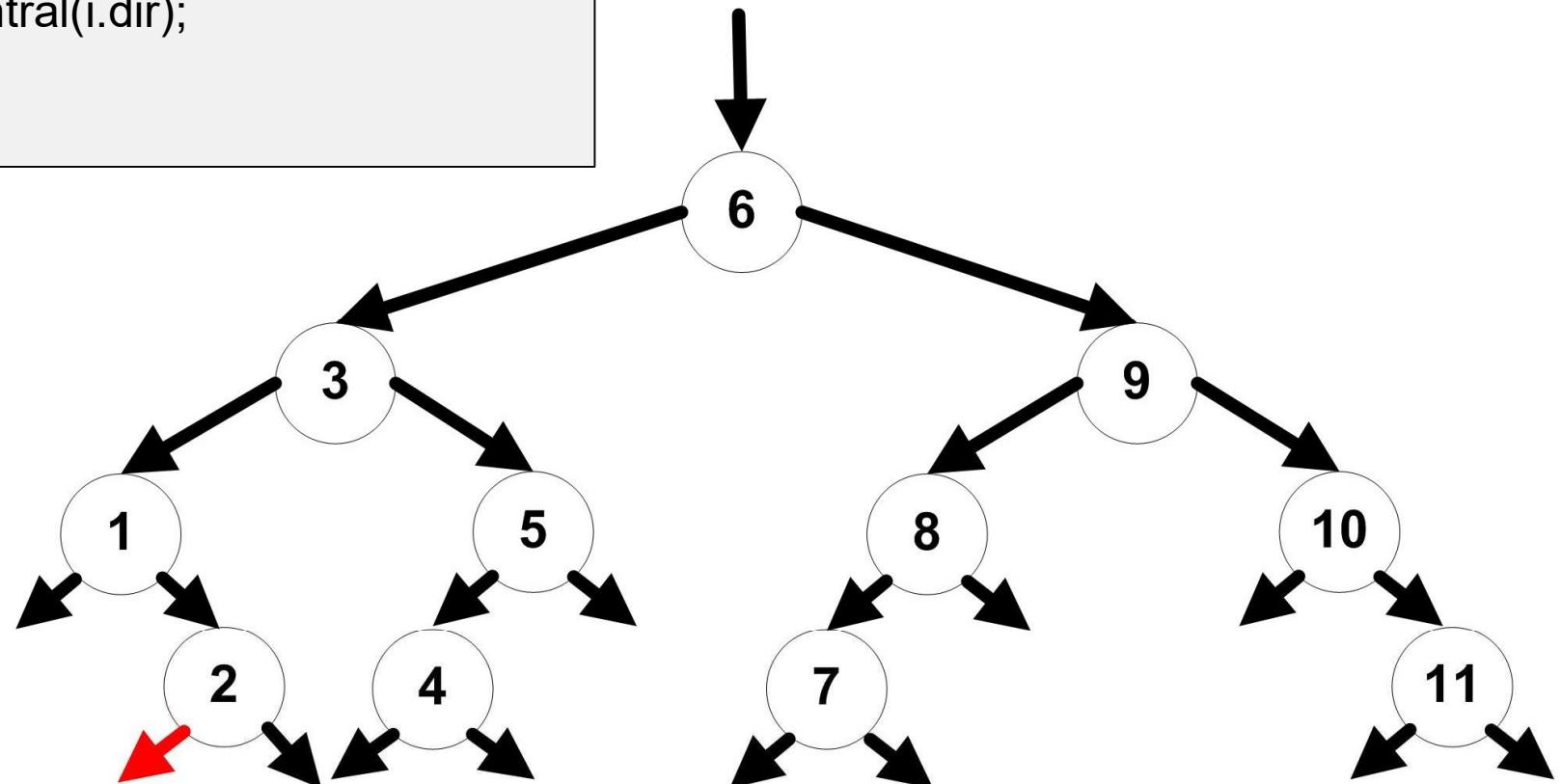


Tela

1

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

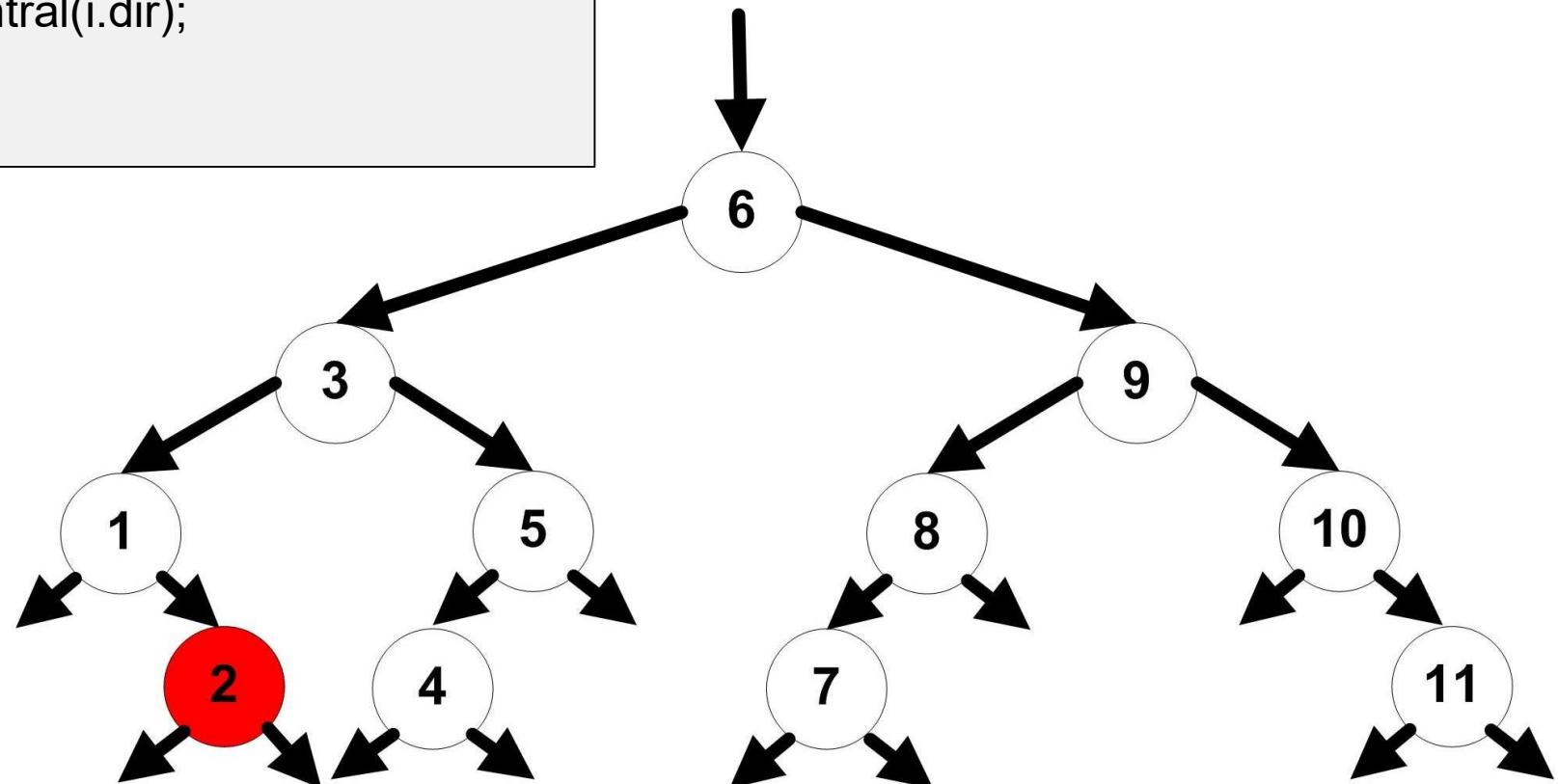


Tela

1

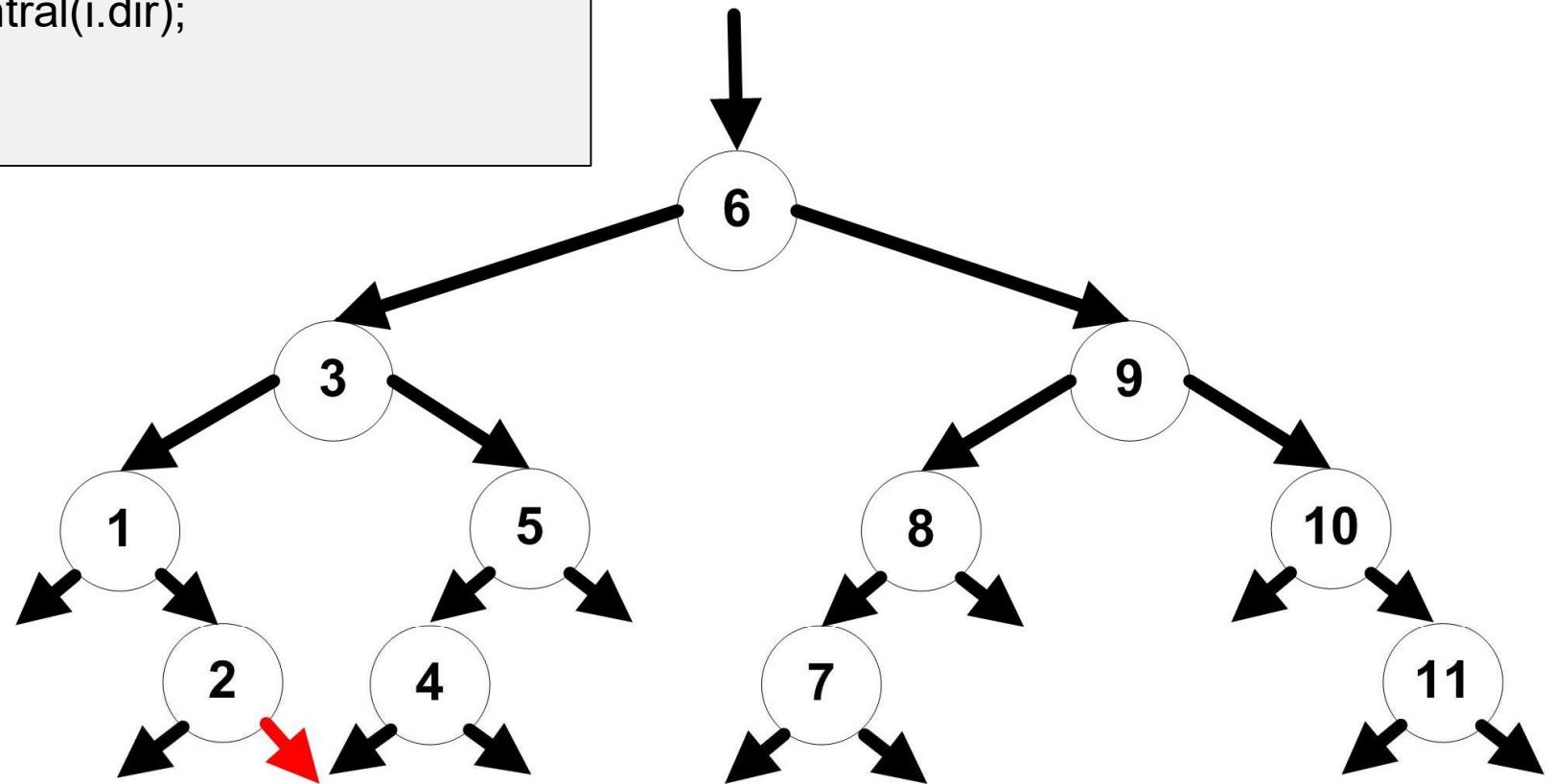
Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

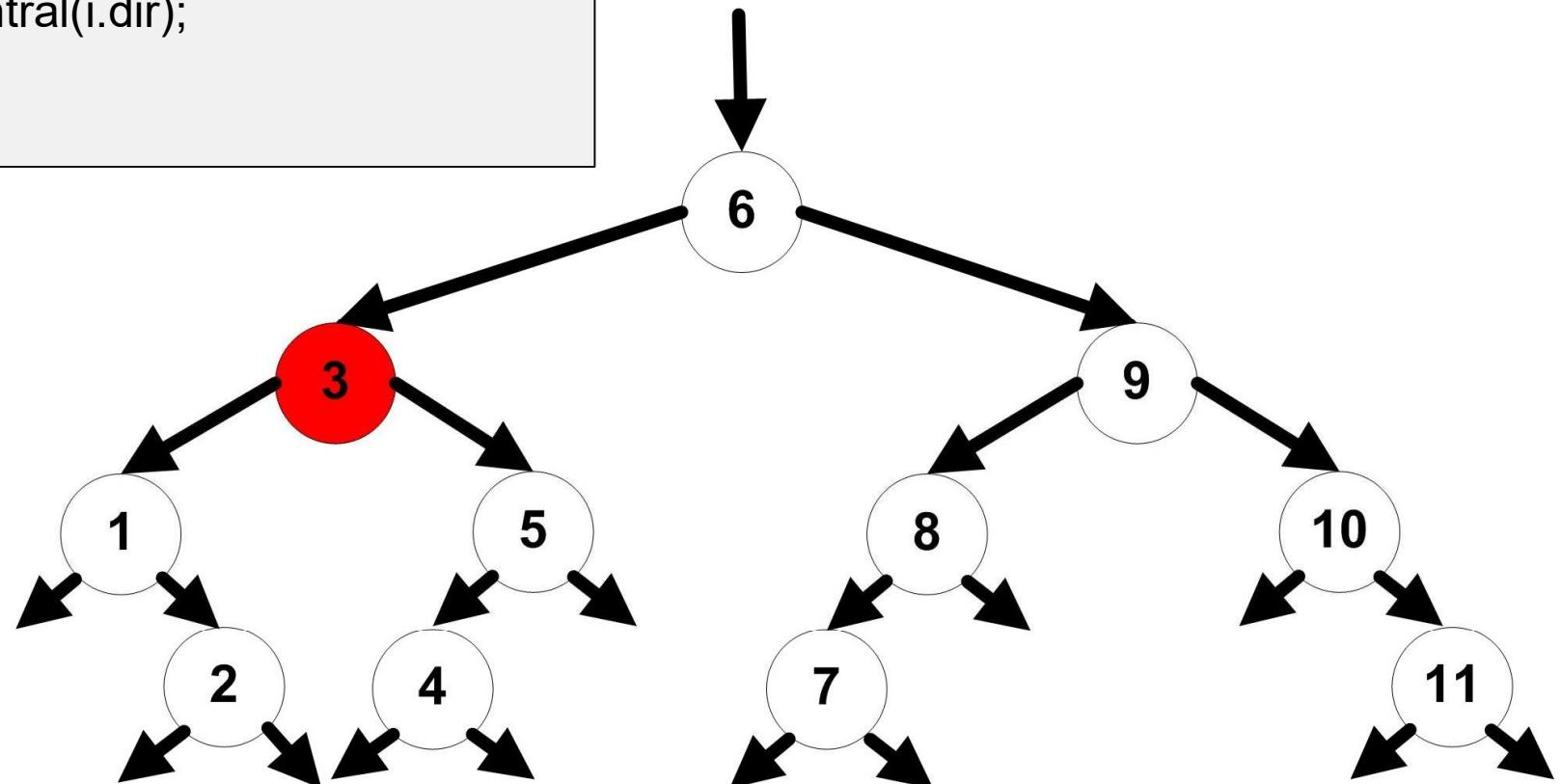


Tela

1 2

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

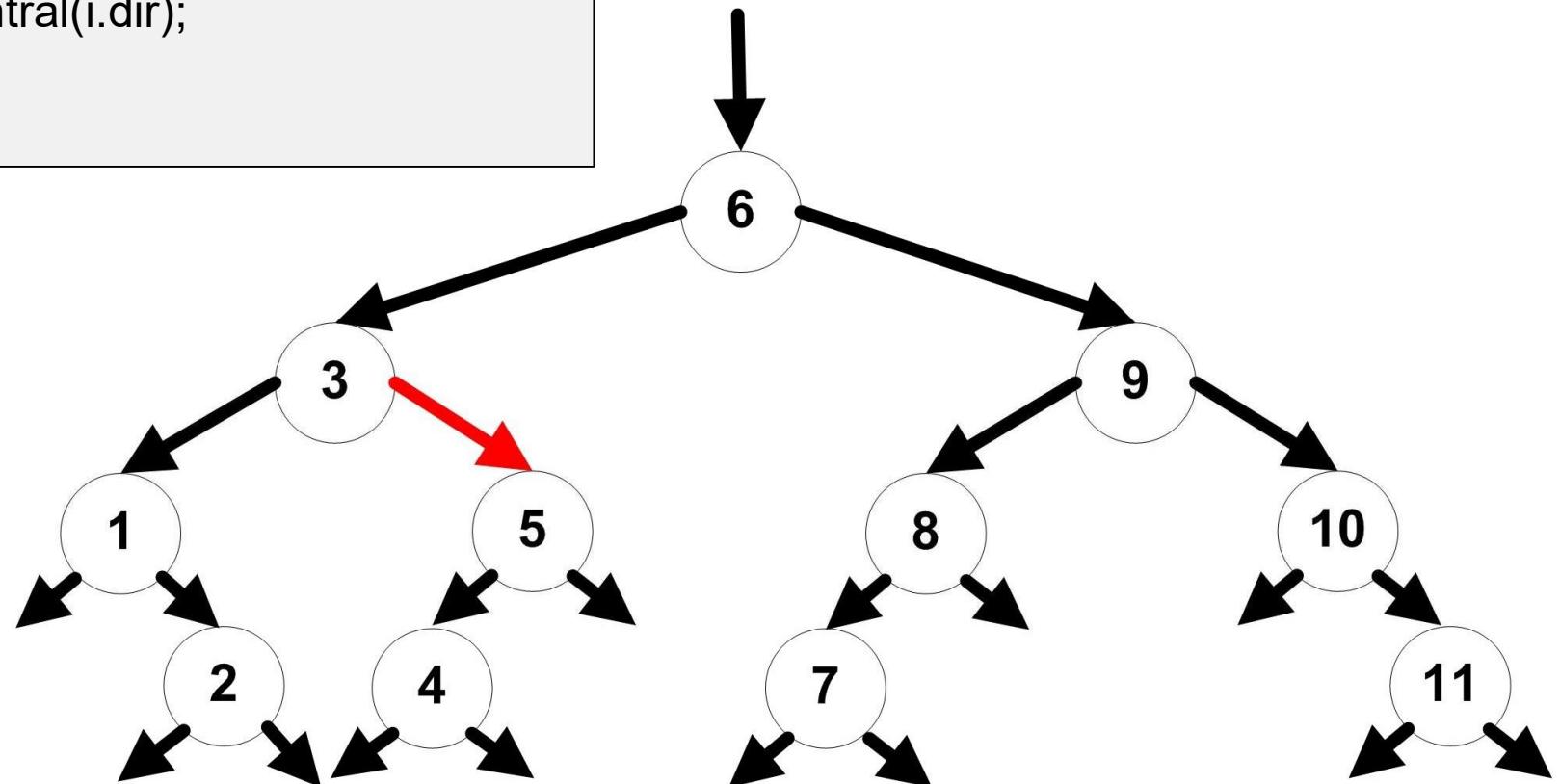


Tela

1 2 3

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

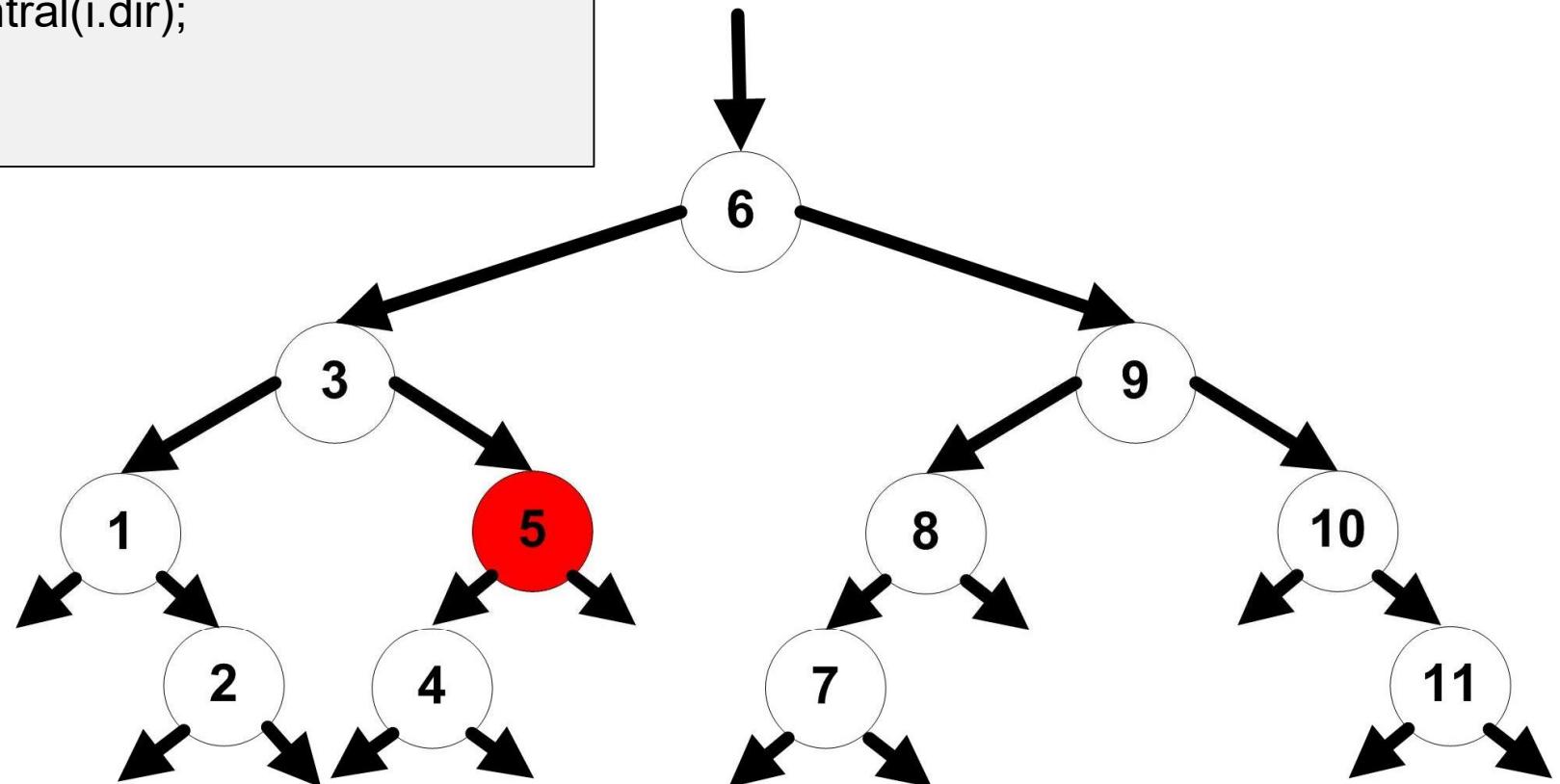


Tela

1 2 3

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

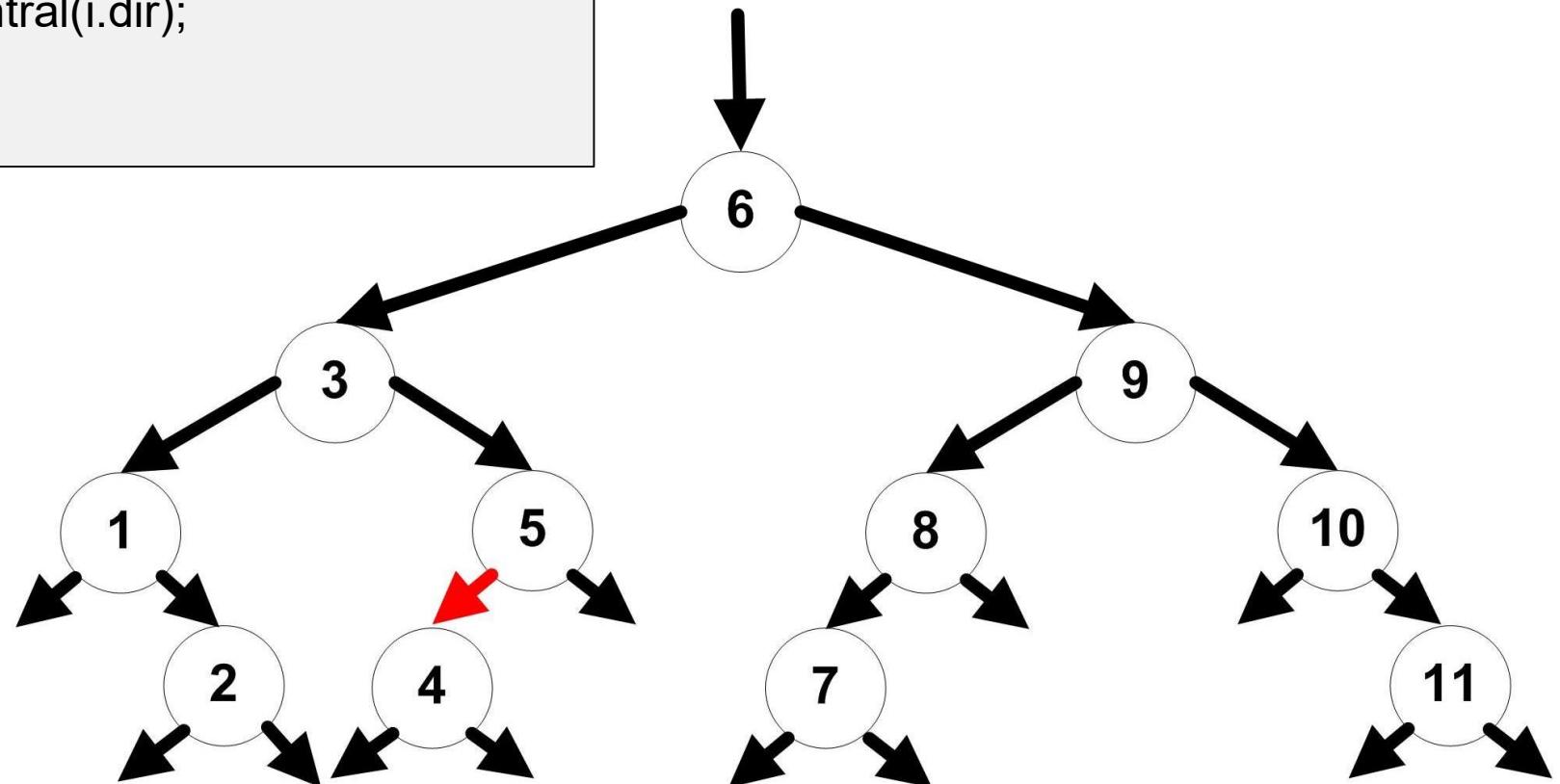


Tela

1 2 3

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

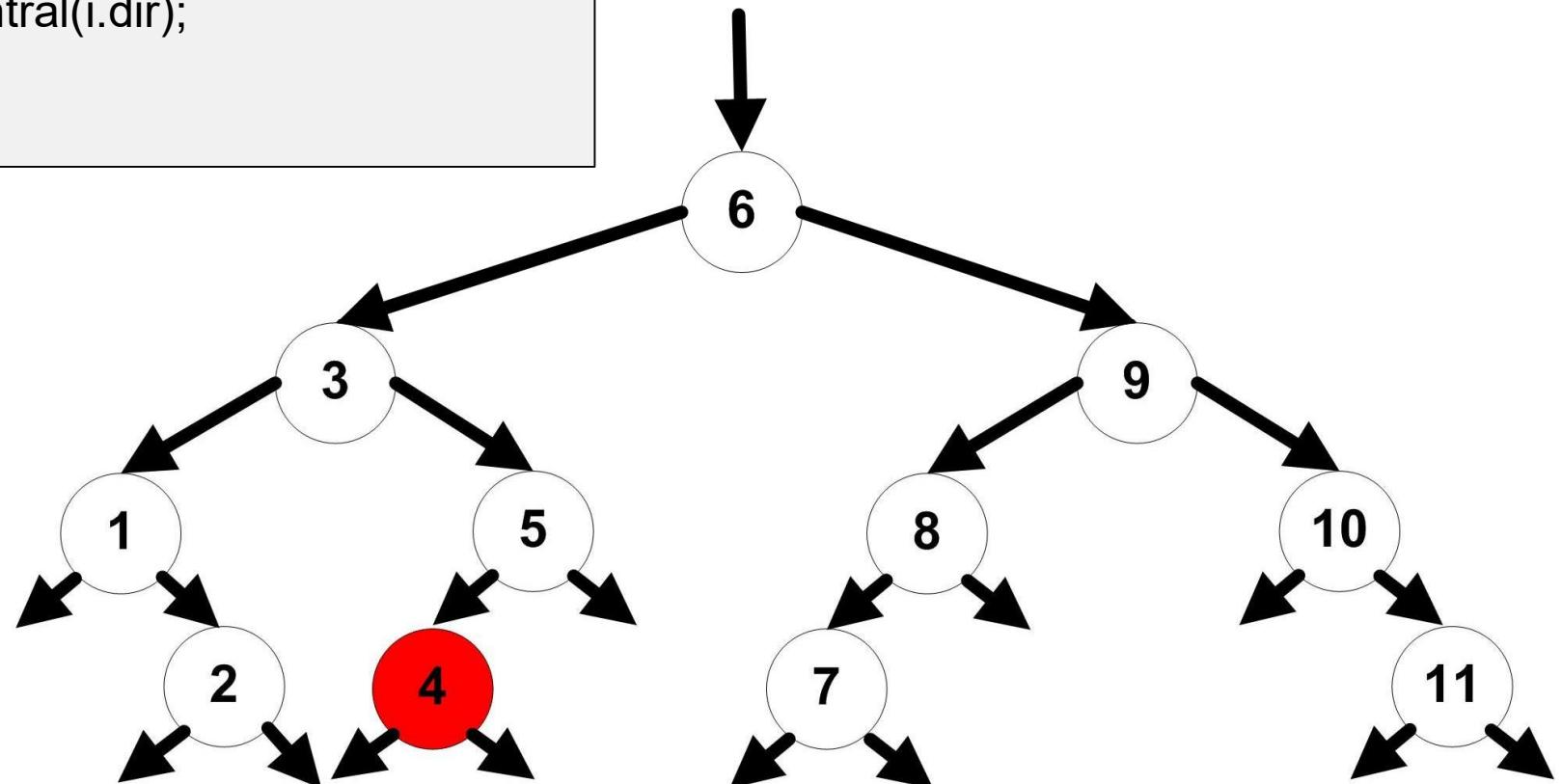


Tela

1 2 3

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

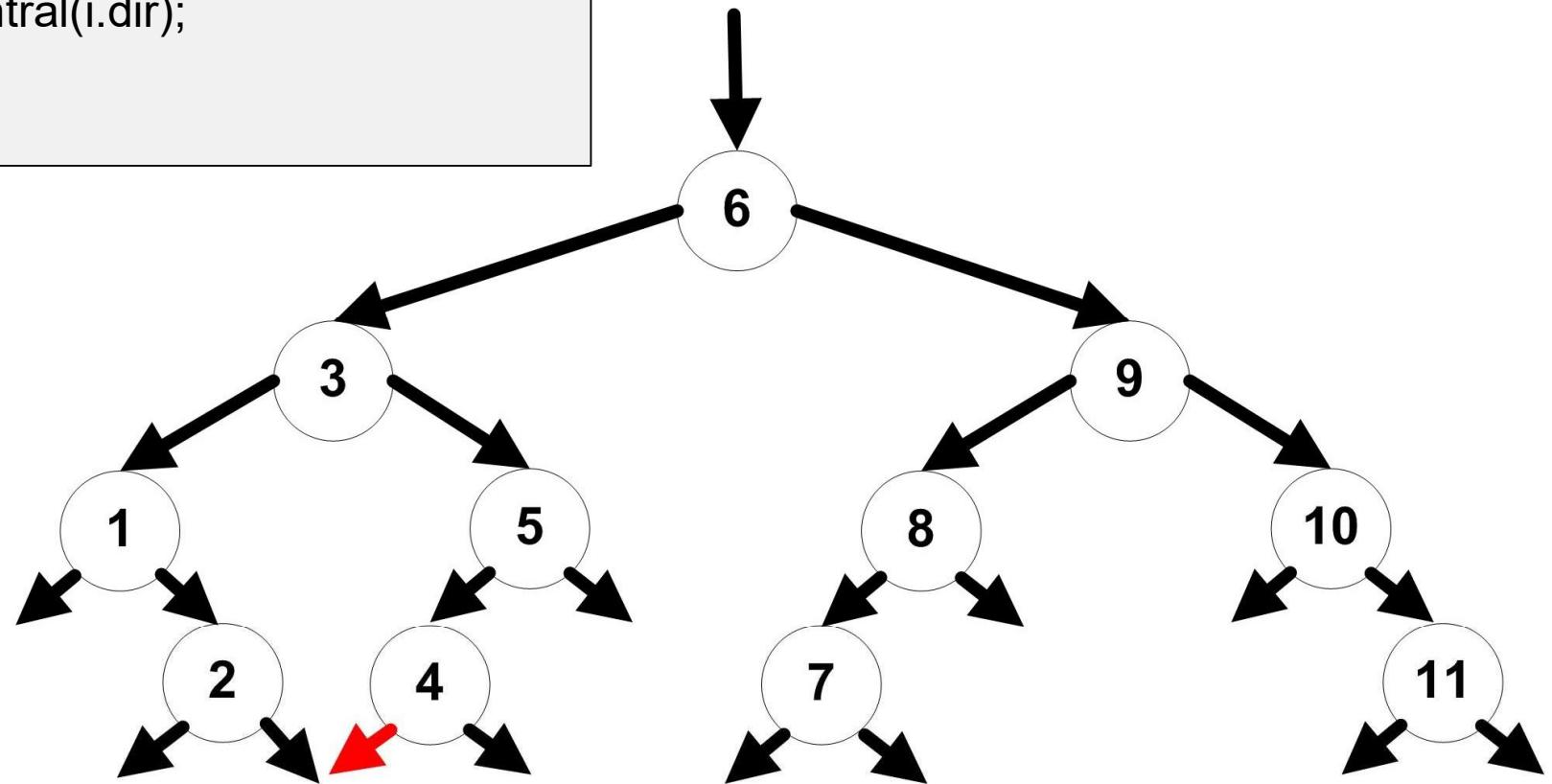


Tela

1 2 3

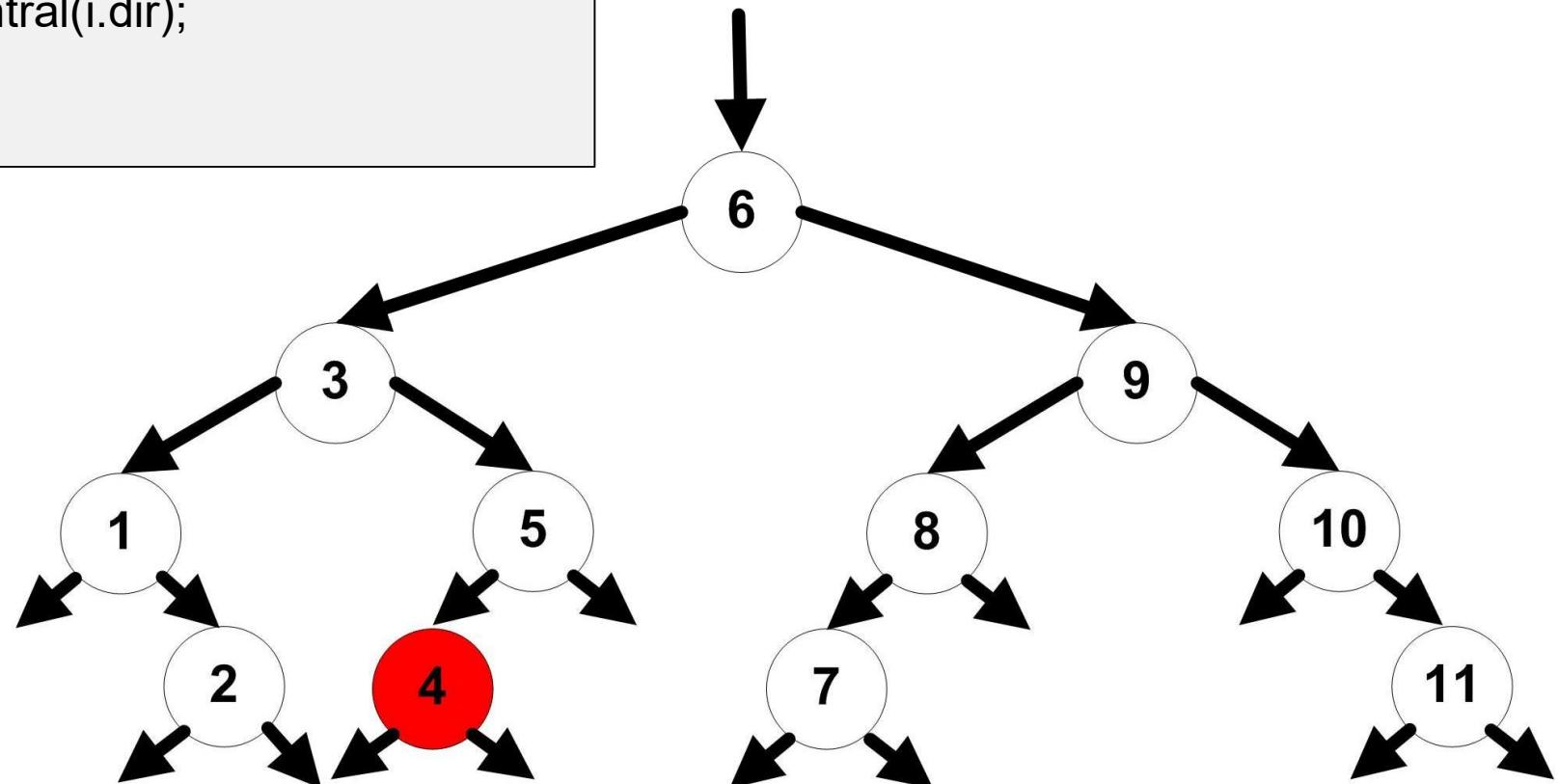
Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

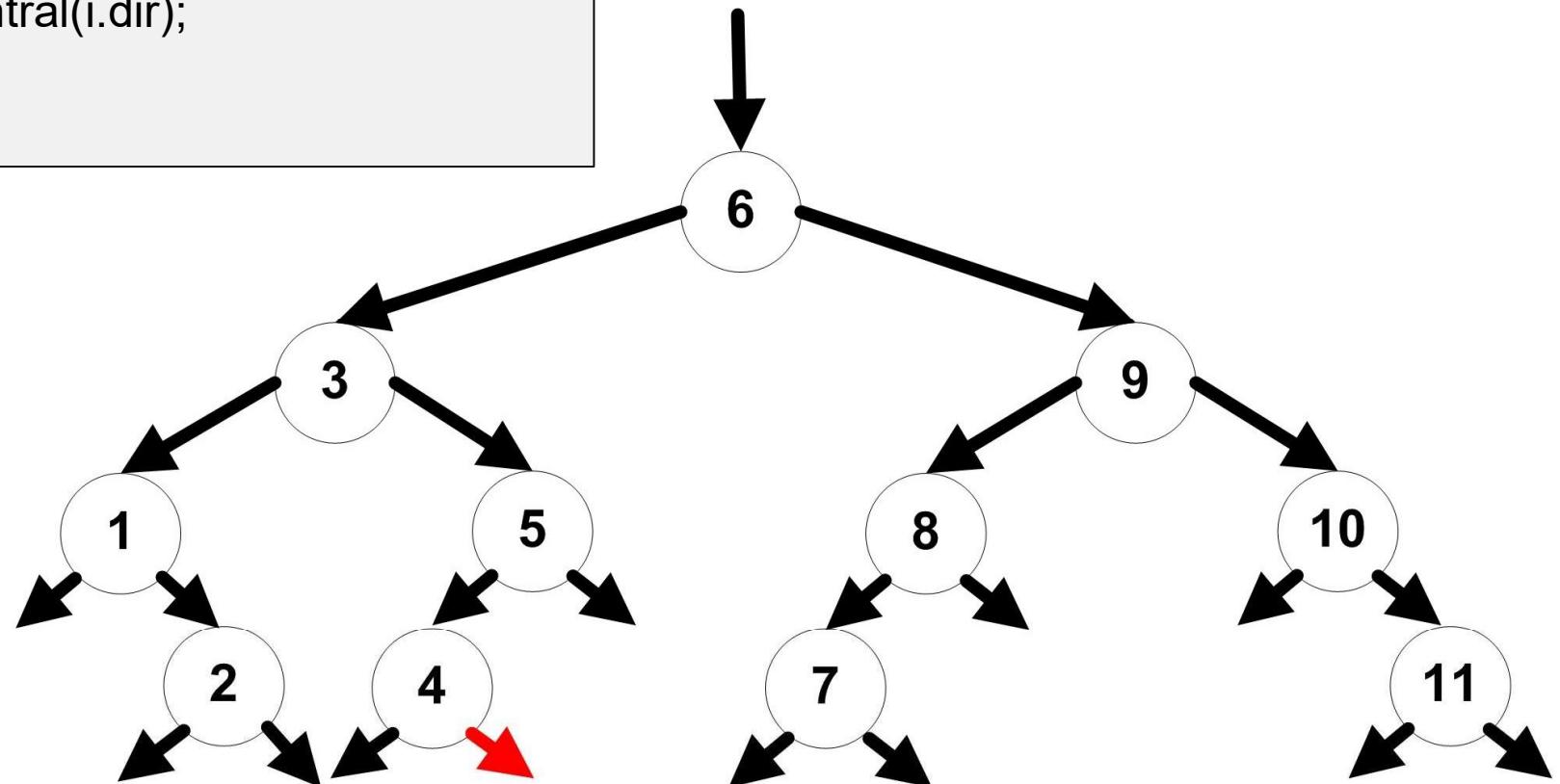


Tela

1 2 3 4

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

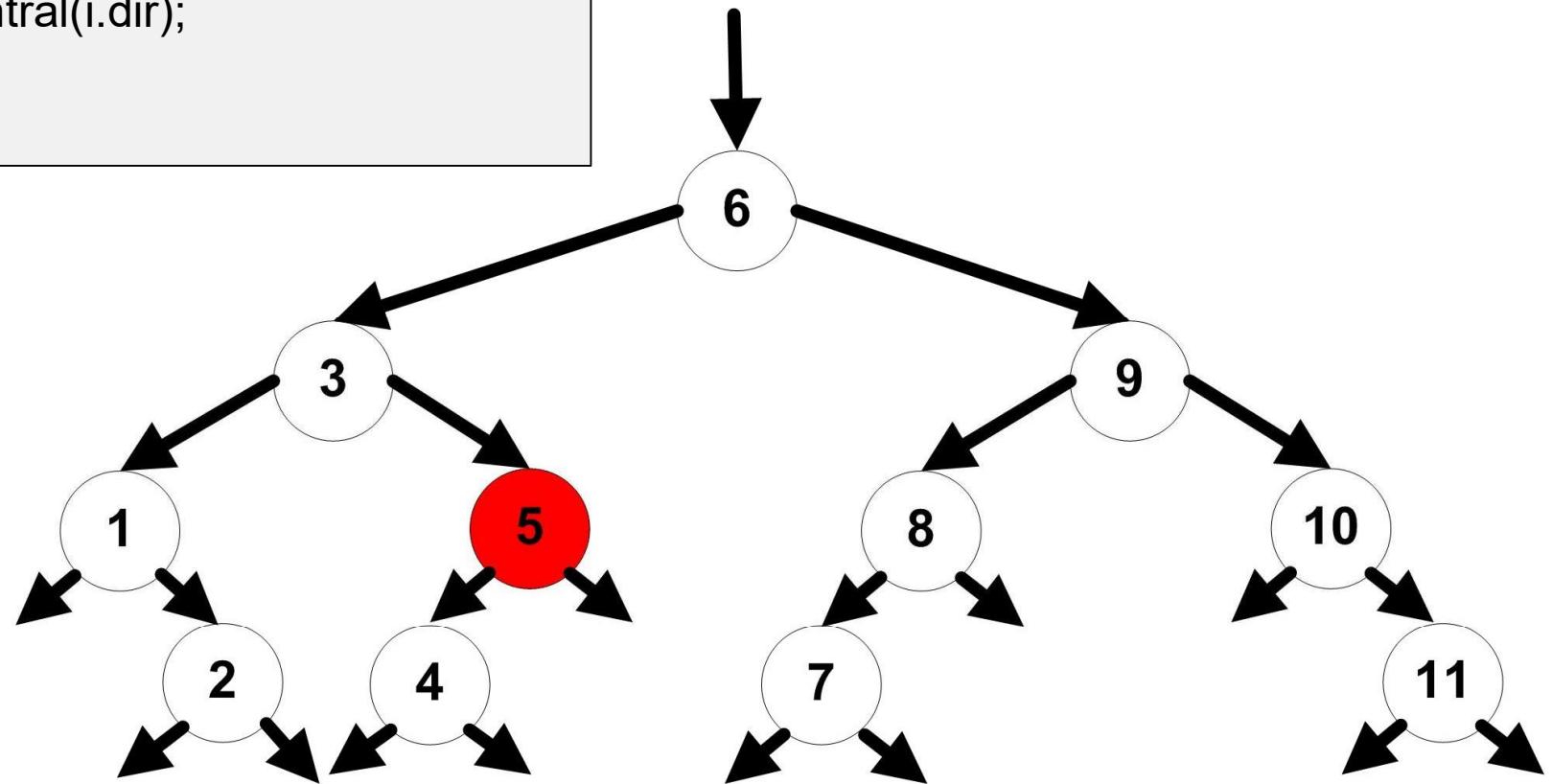


Tela

1 2 3 4

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

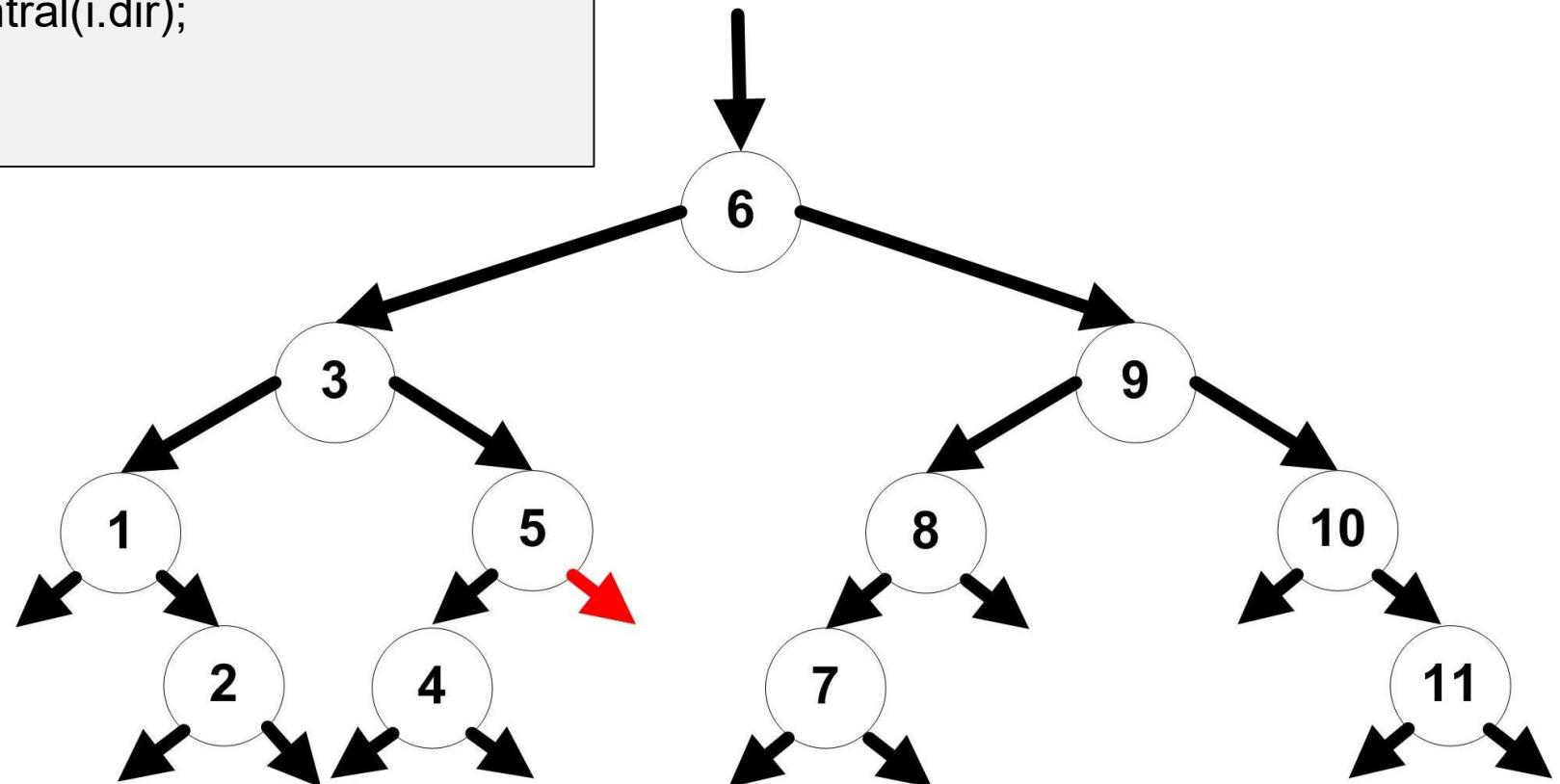


Tela

1 2 3 4 5

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

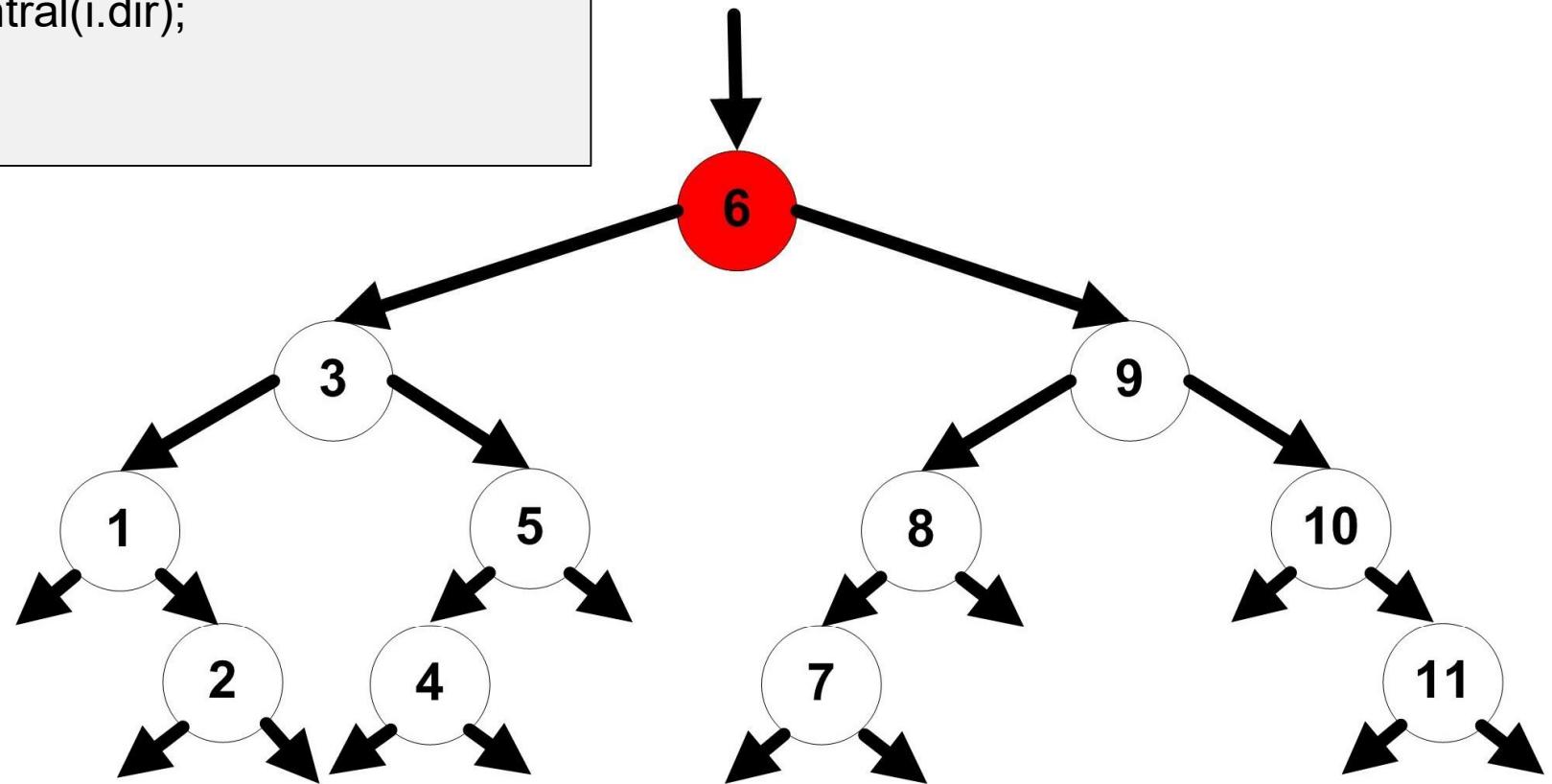


Tela

1 2 3 4 5

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

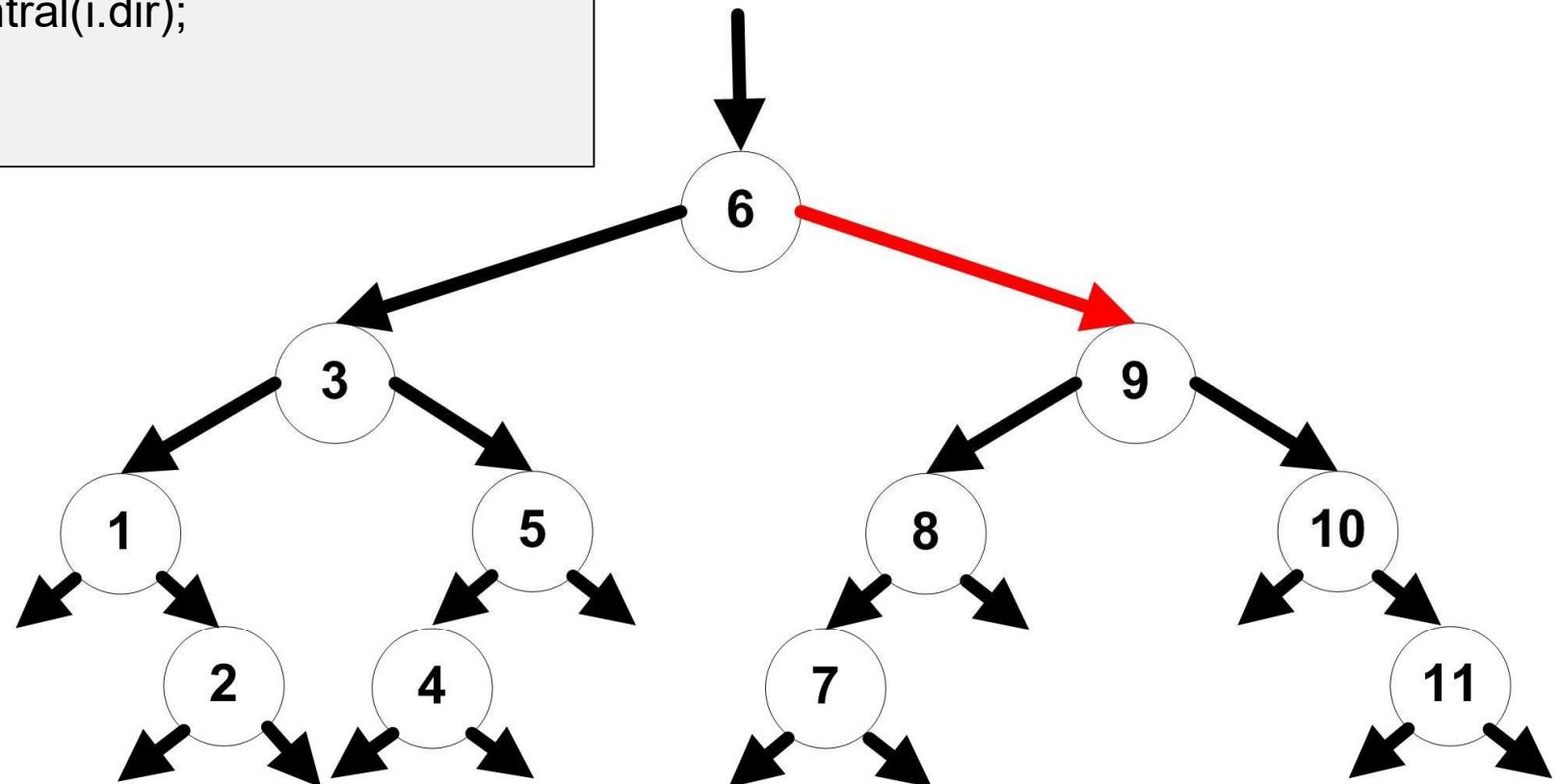


Tela

1 2 3 4 5 6

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

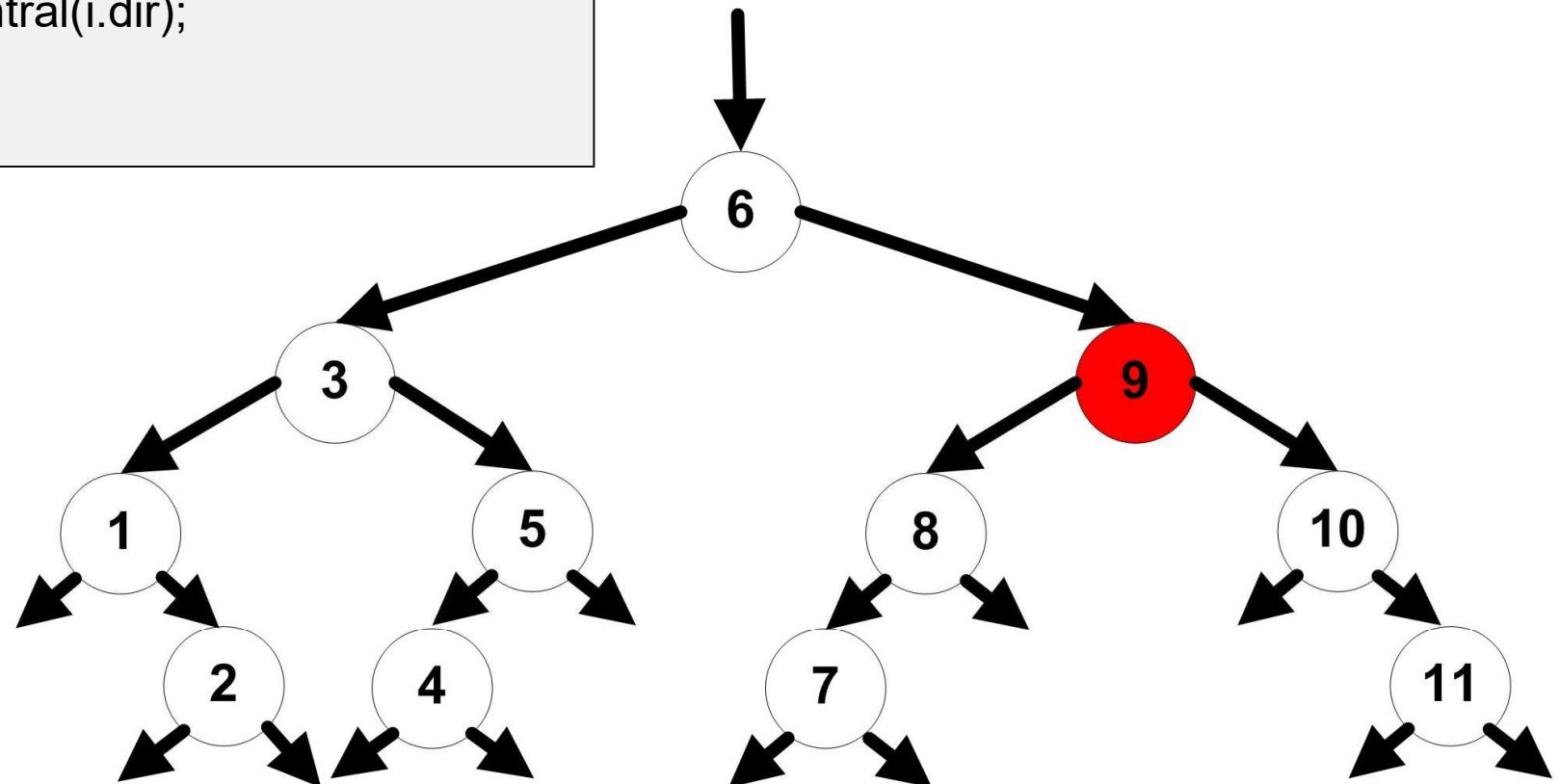


Tela

1 2 3 4 5 6

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

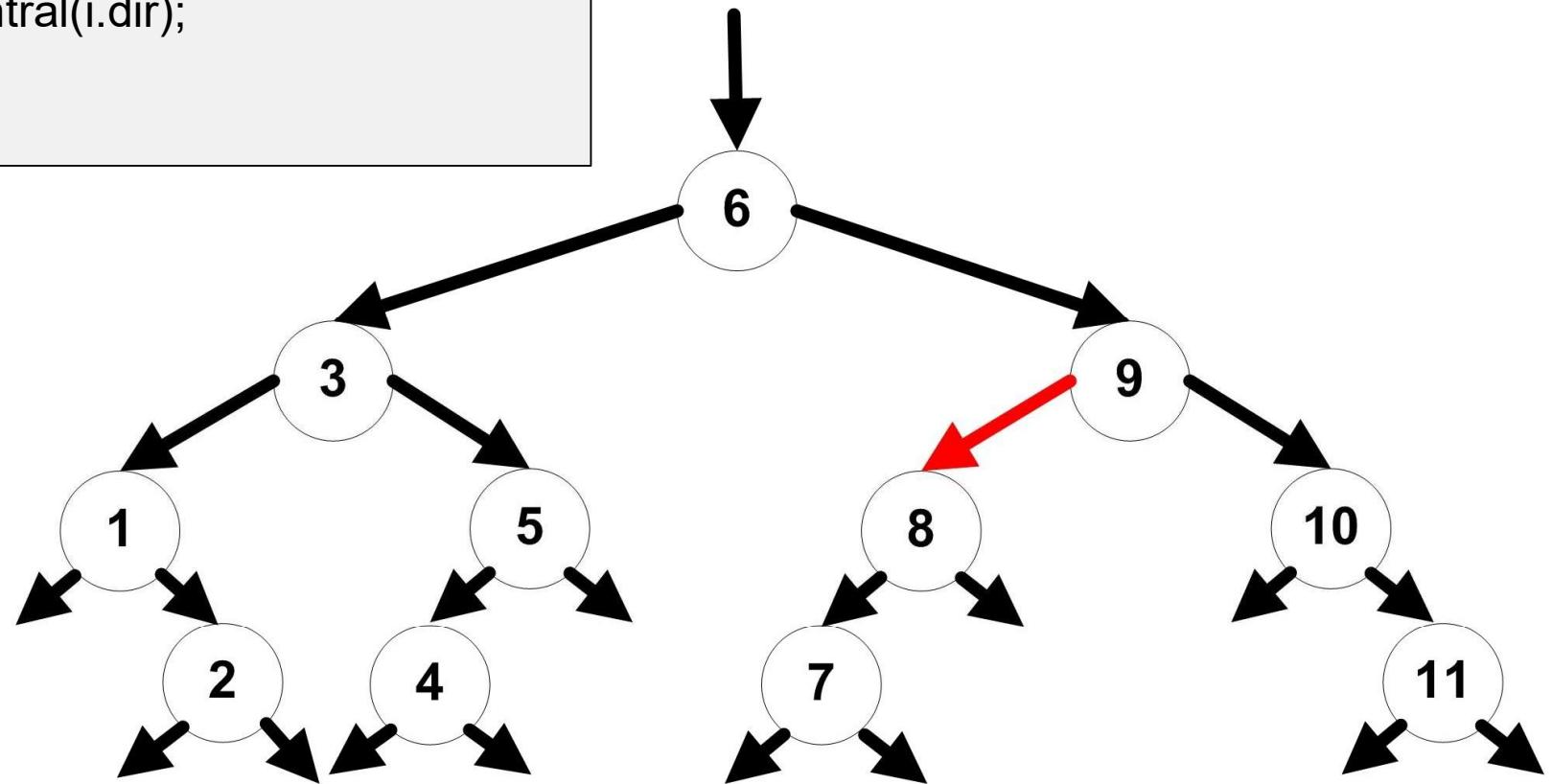


Tela

1 2 3 4 5 6

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

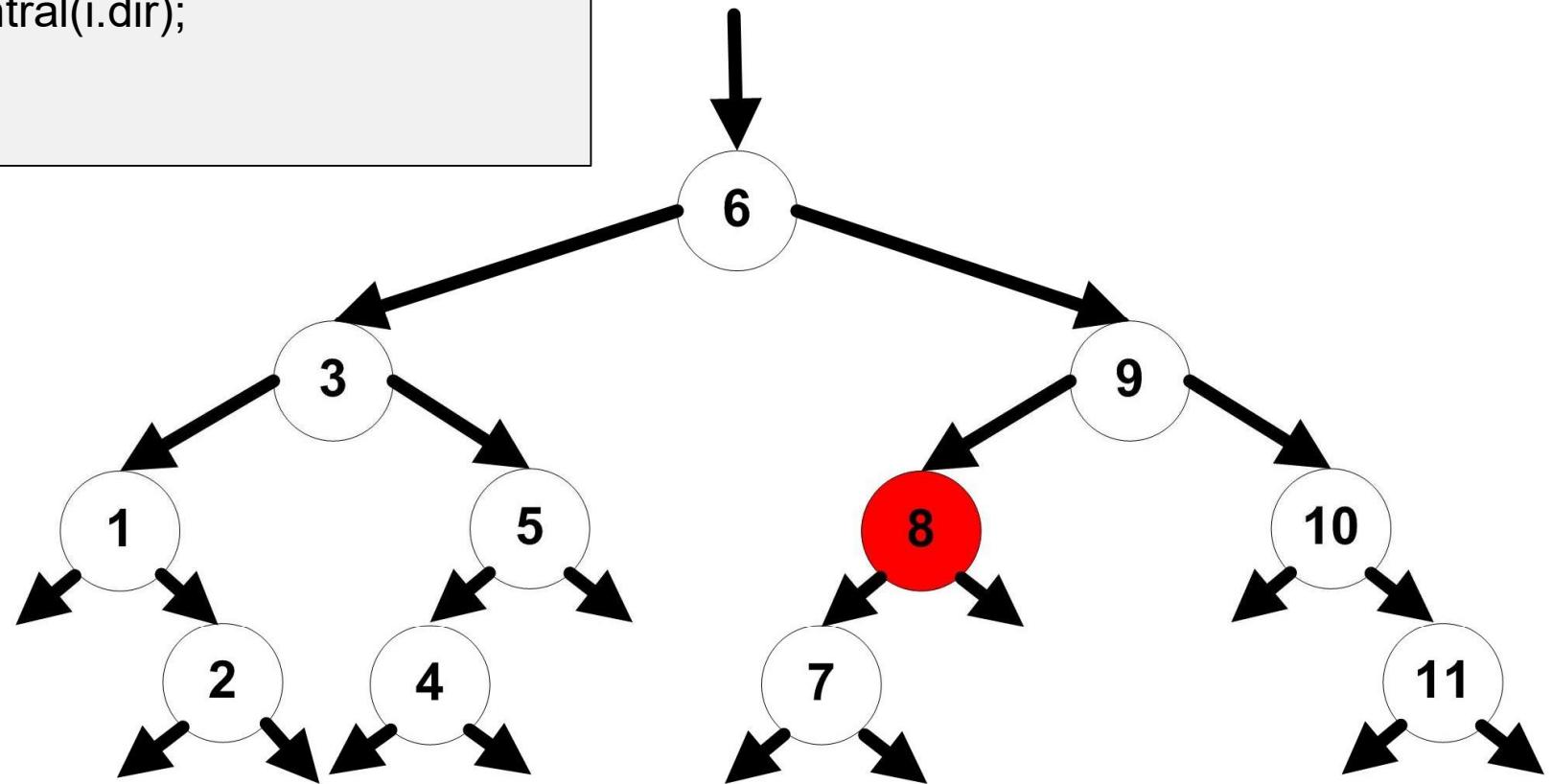


Tela

1 2 3 4 5 6

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

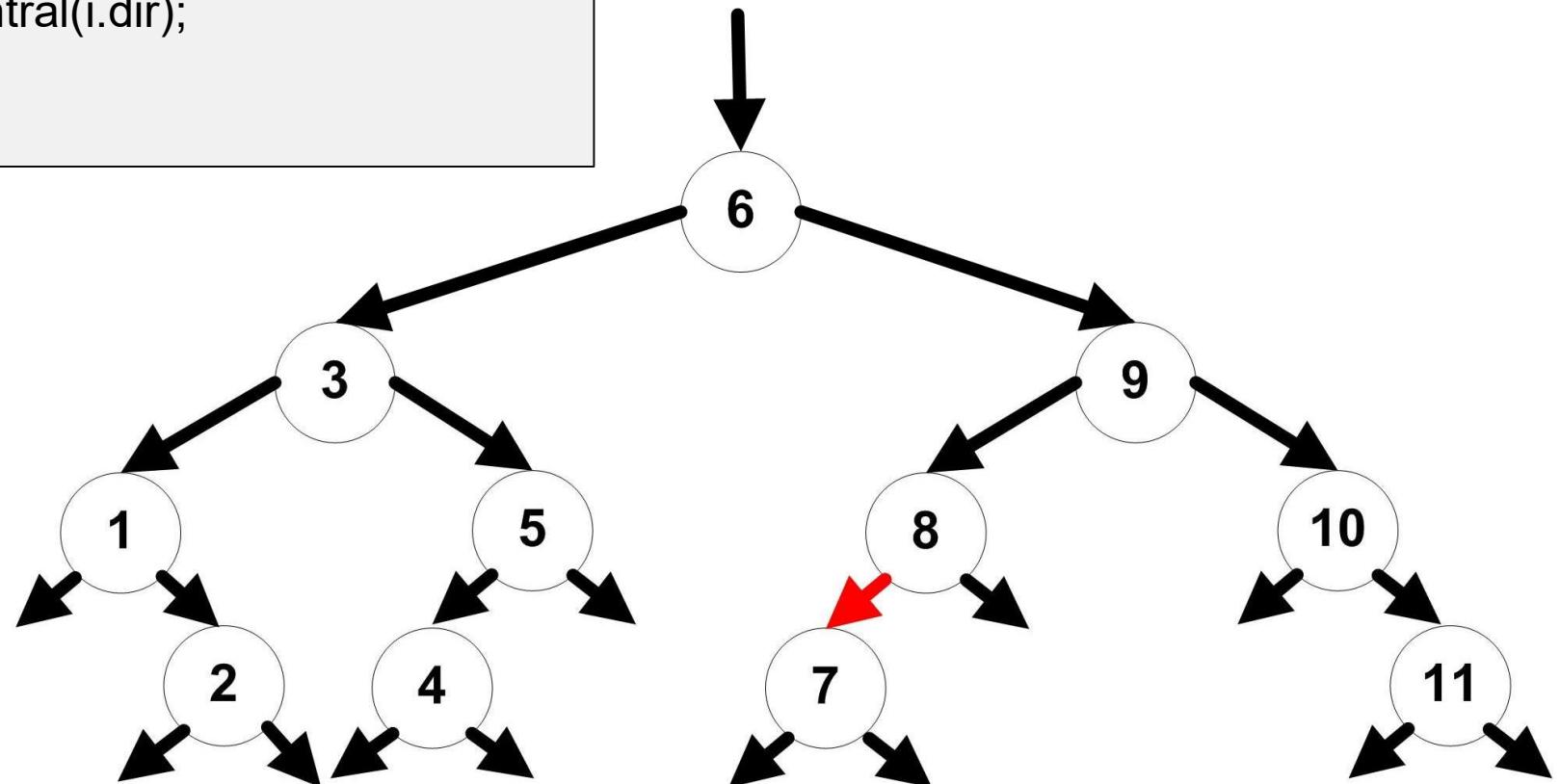


Tela

1 2 3 4 5 6

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

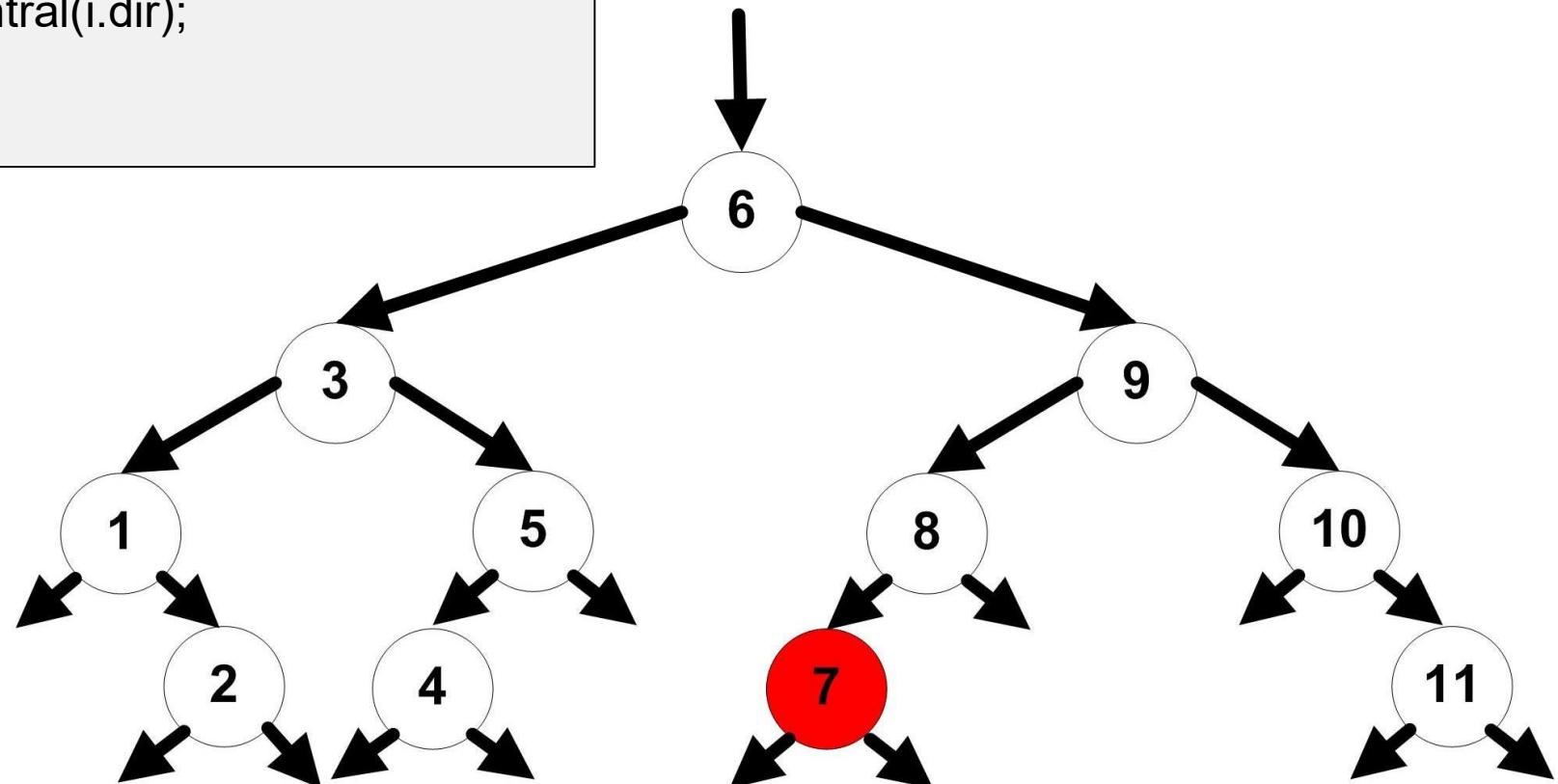


Tela

1 2 3 4 5 6

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

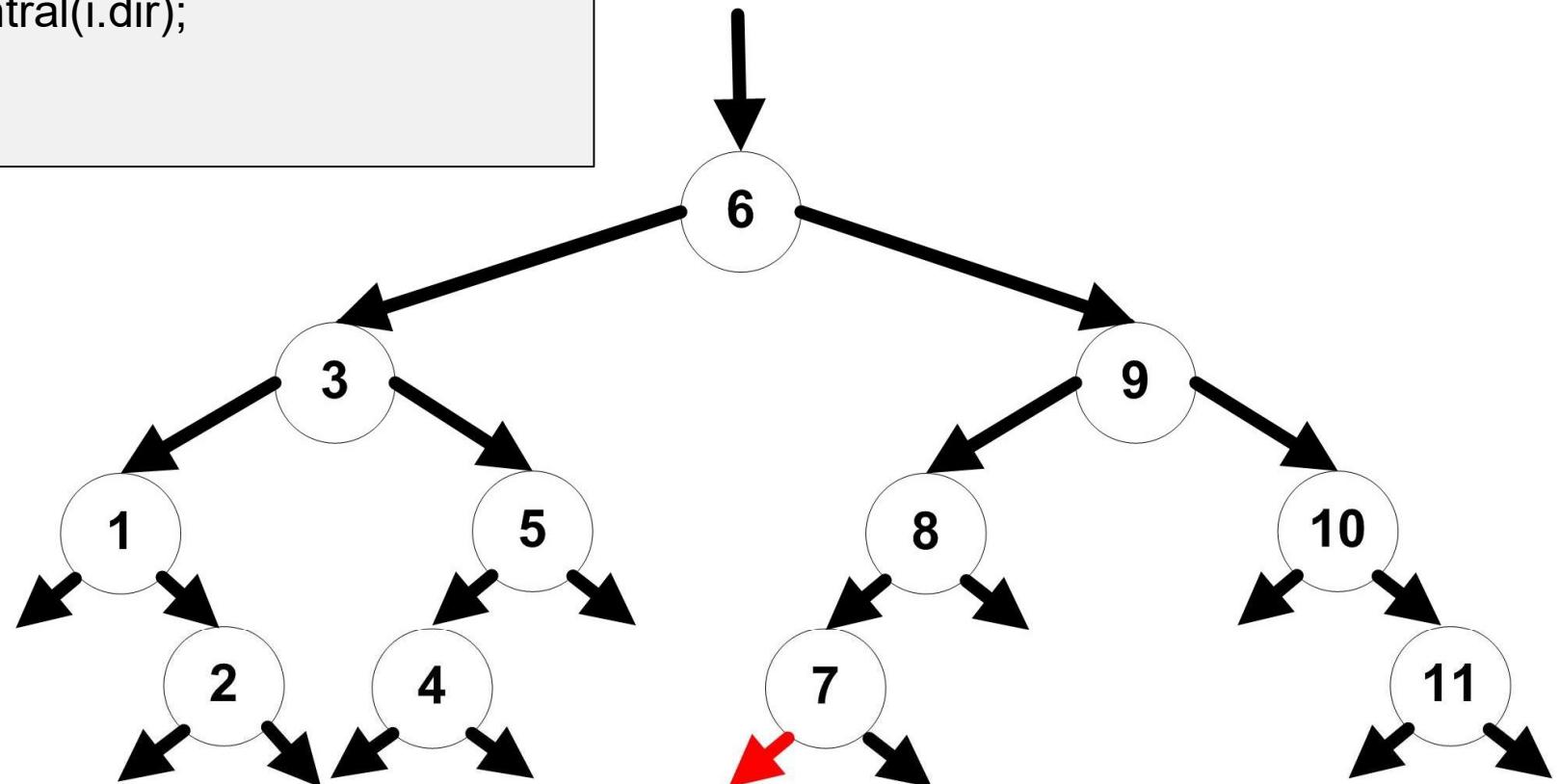


Tela

1 2 3 4 5 6

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

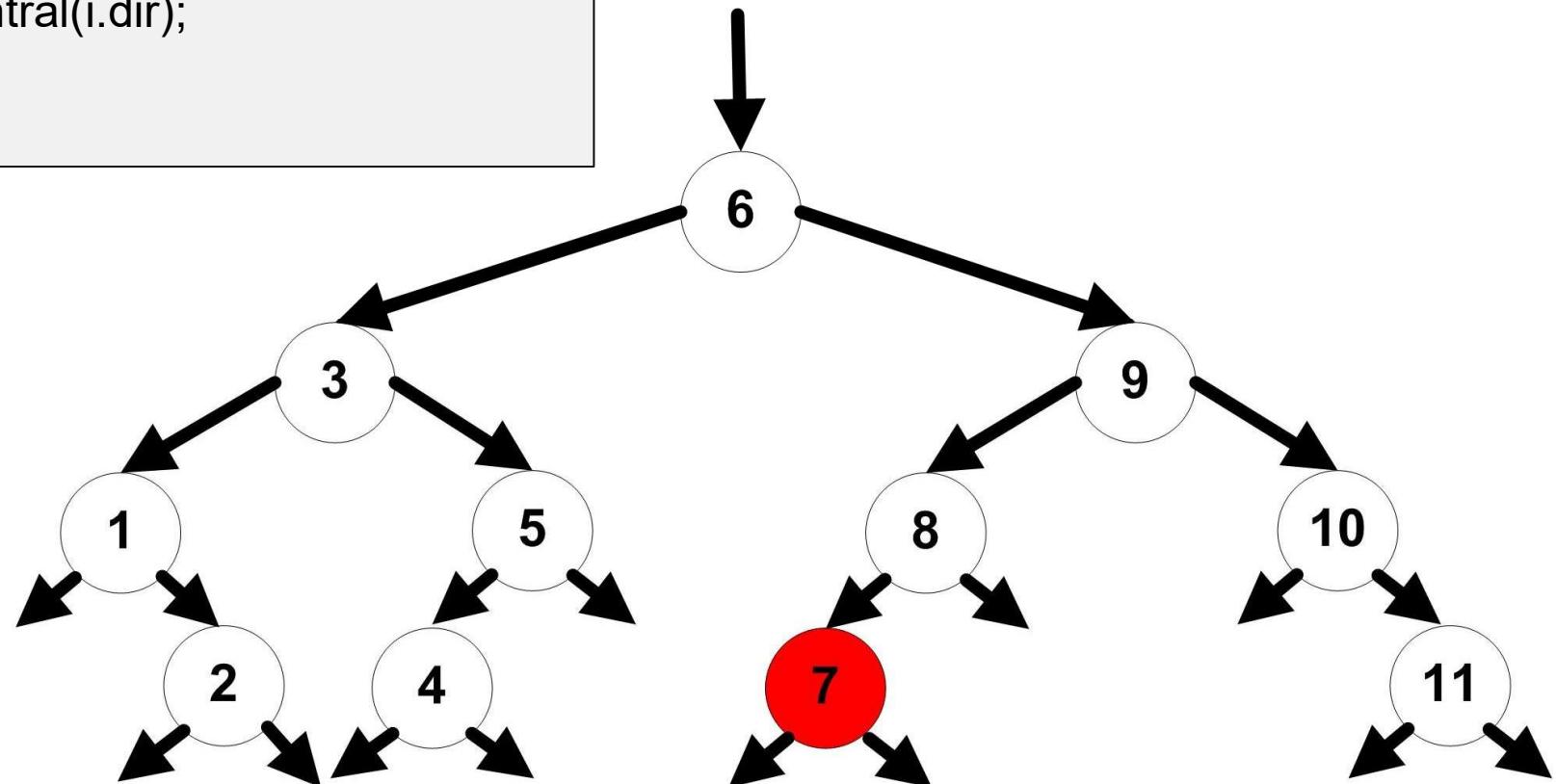


Tela

1 2 3 4 5 6

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

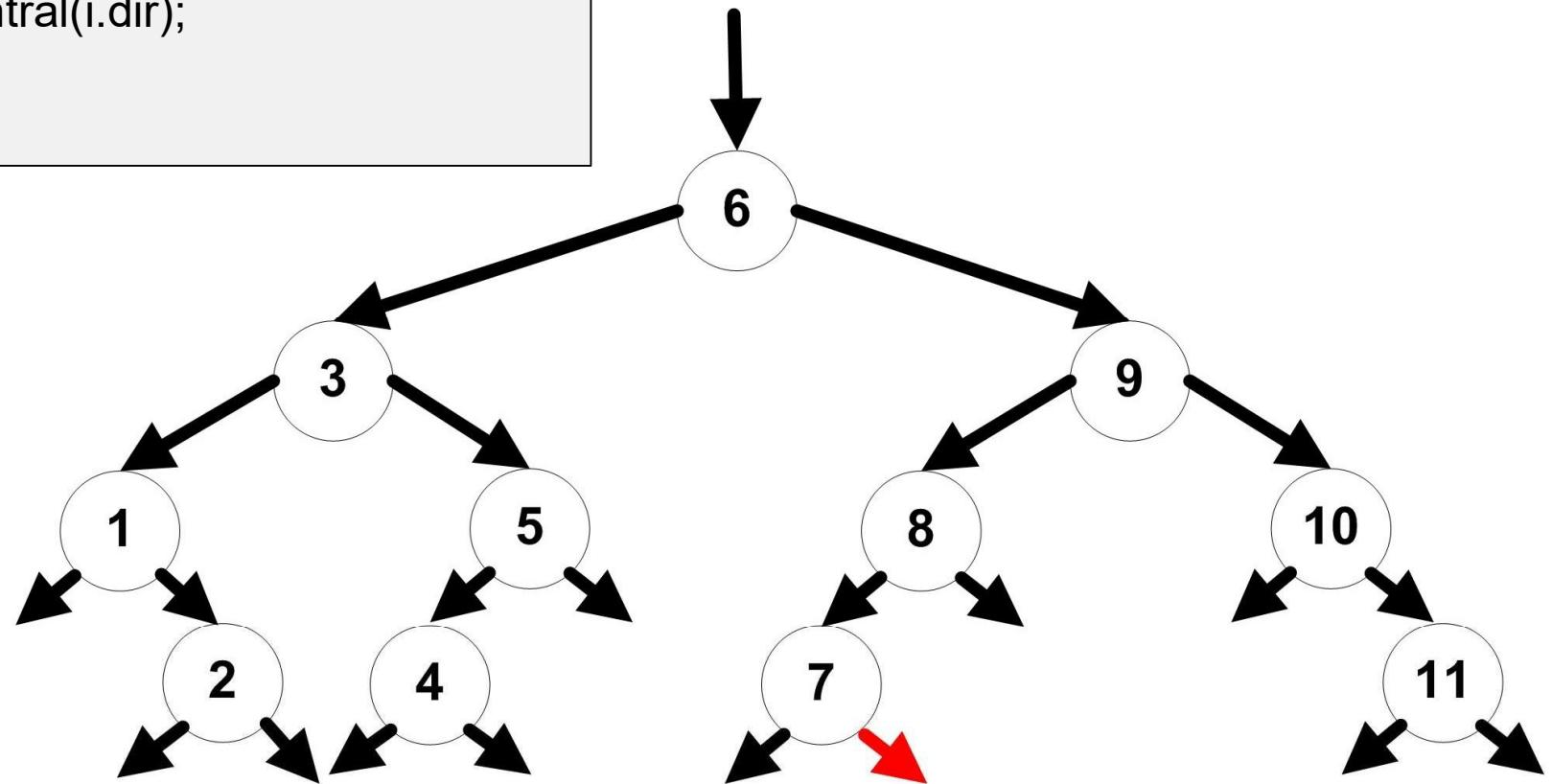


Tela

1 2 3 4 5 6 7

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

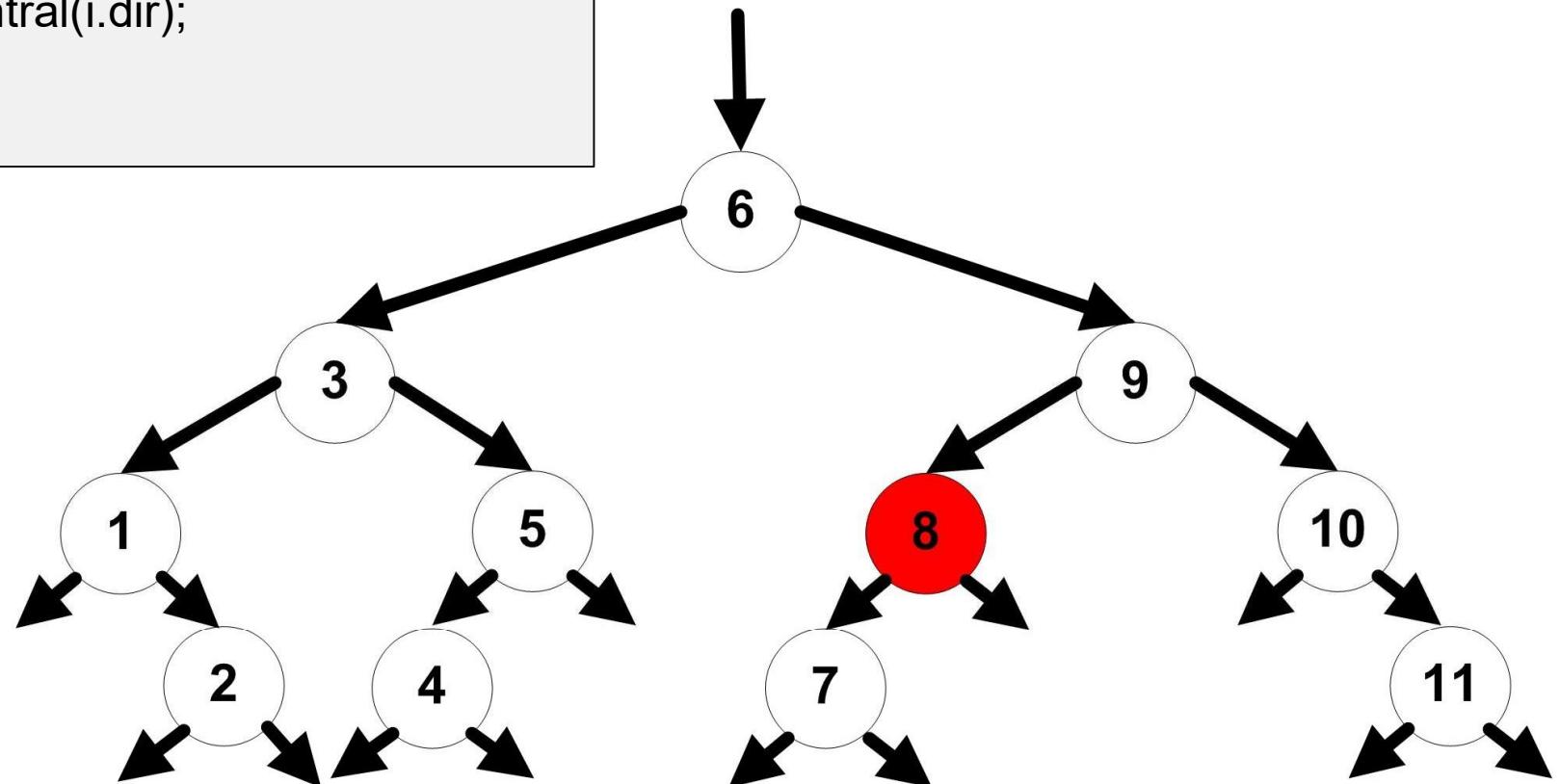


Tela

1 2 3 4 5 6 7

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

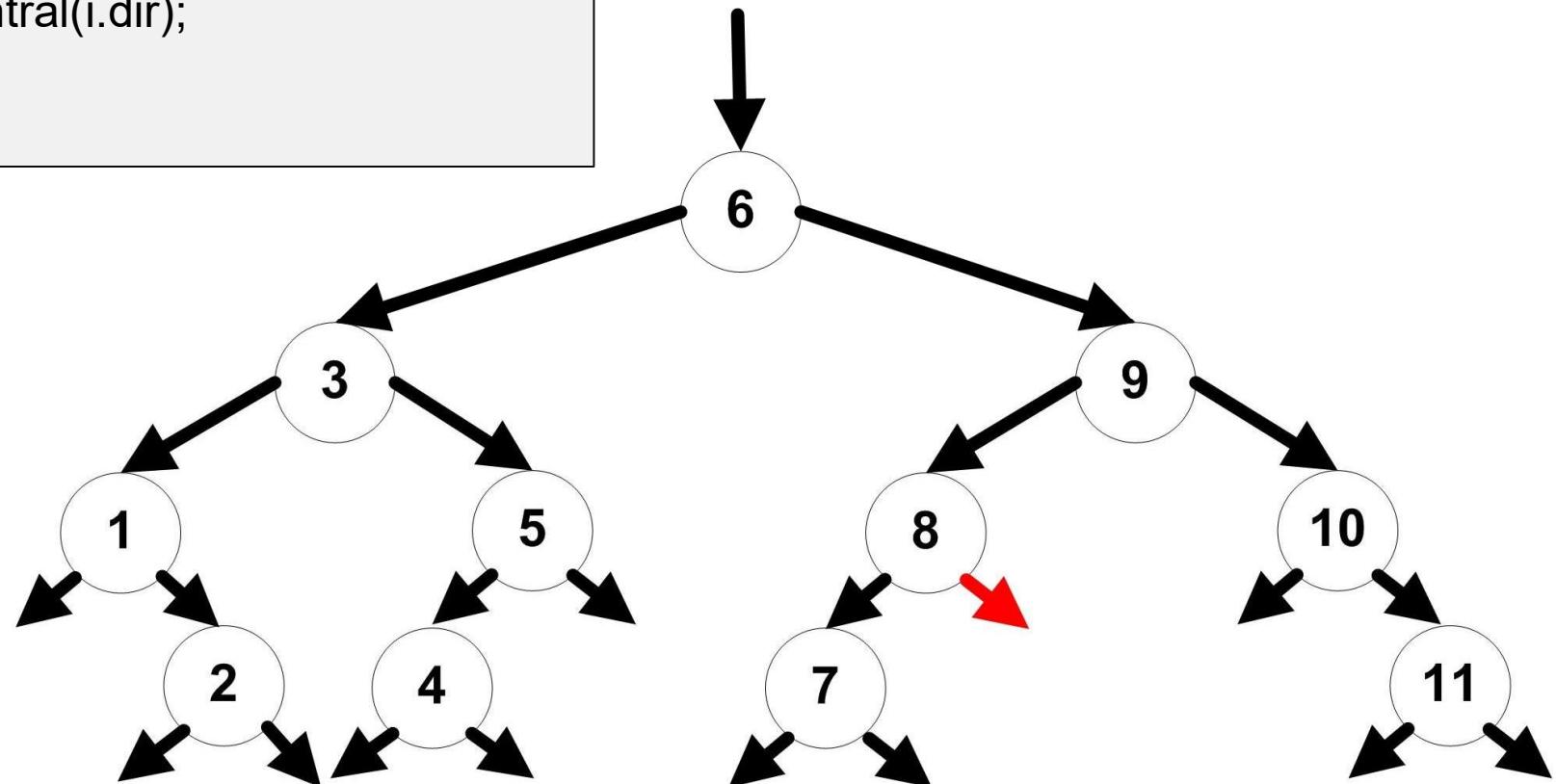


Tela

1 2 3 4 5 6 7 8

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

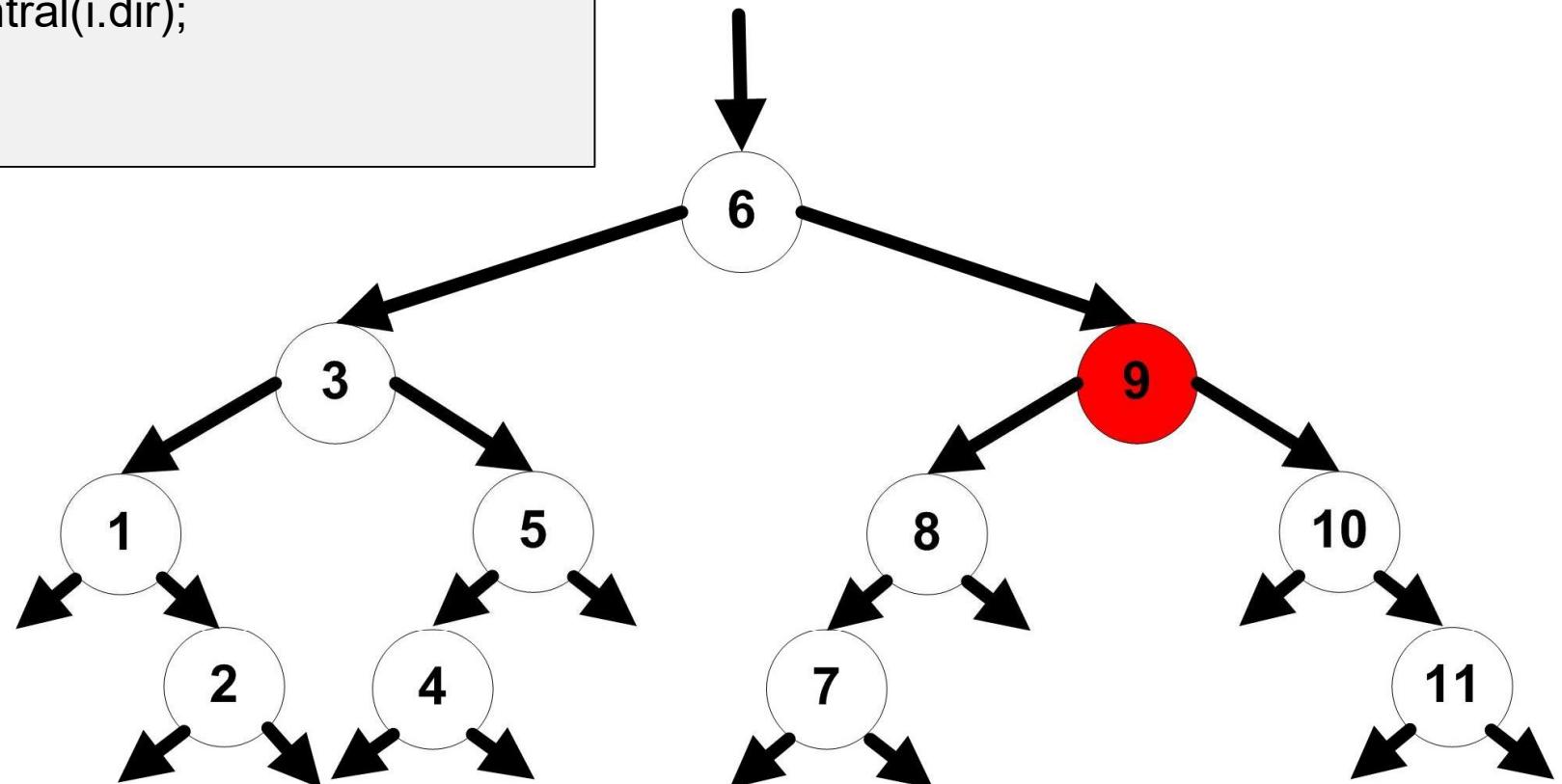


Tela

1 2 3 4 5 6 7 8

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

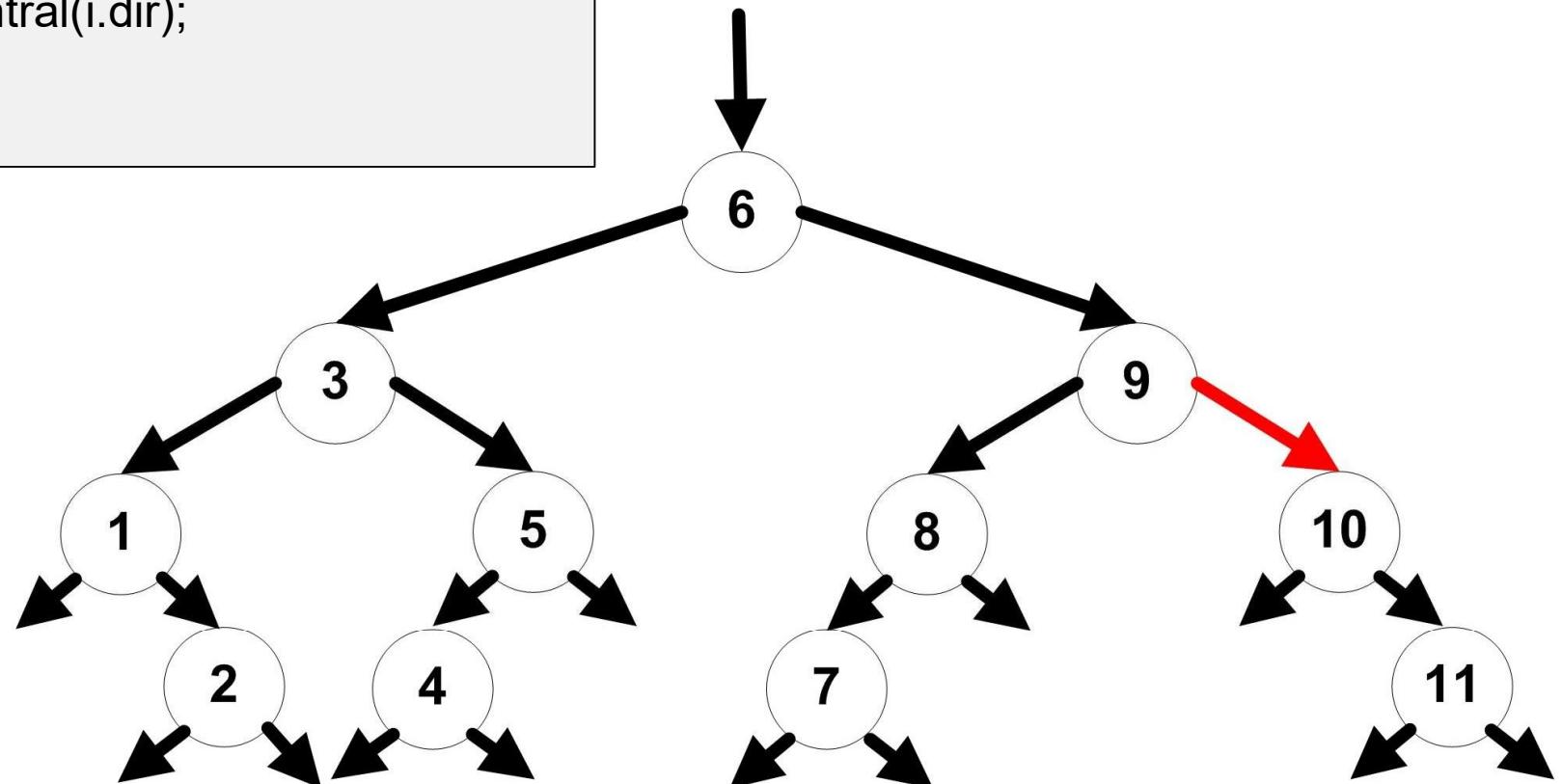


Tela

1 2 3 4 5 6 7 8 9

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

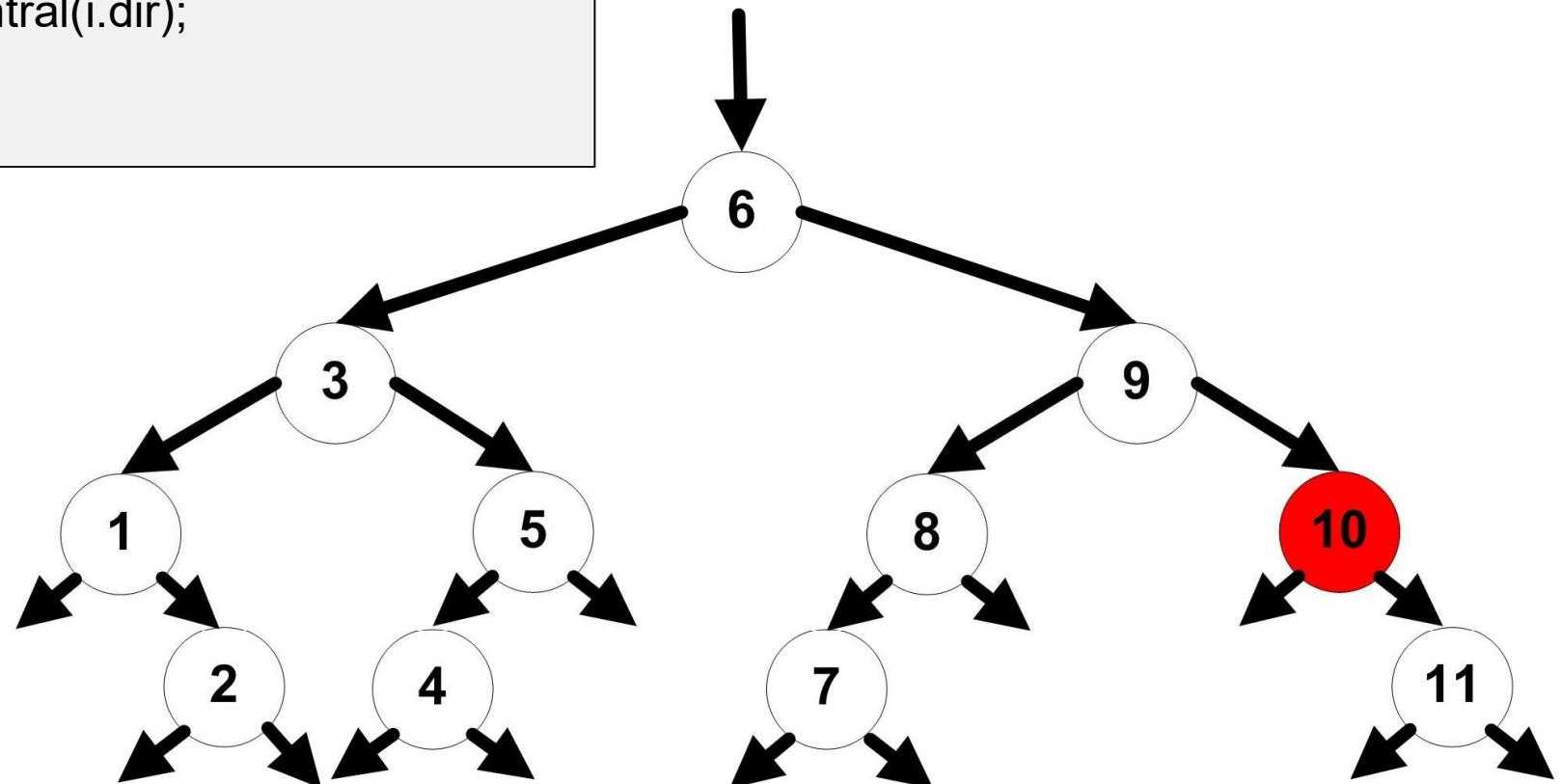


Tela

1 2 3 4 5 6 7 8 9

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Tela

1 2

3

4

5

6

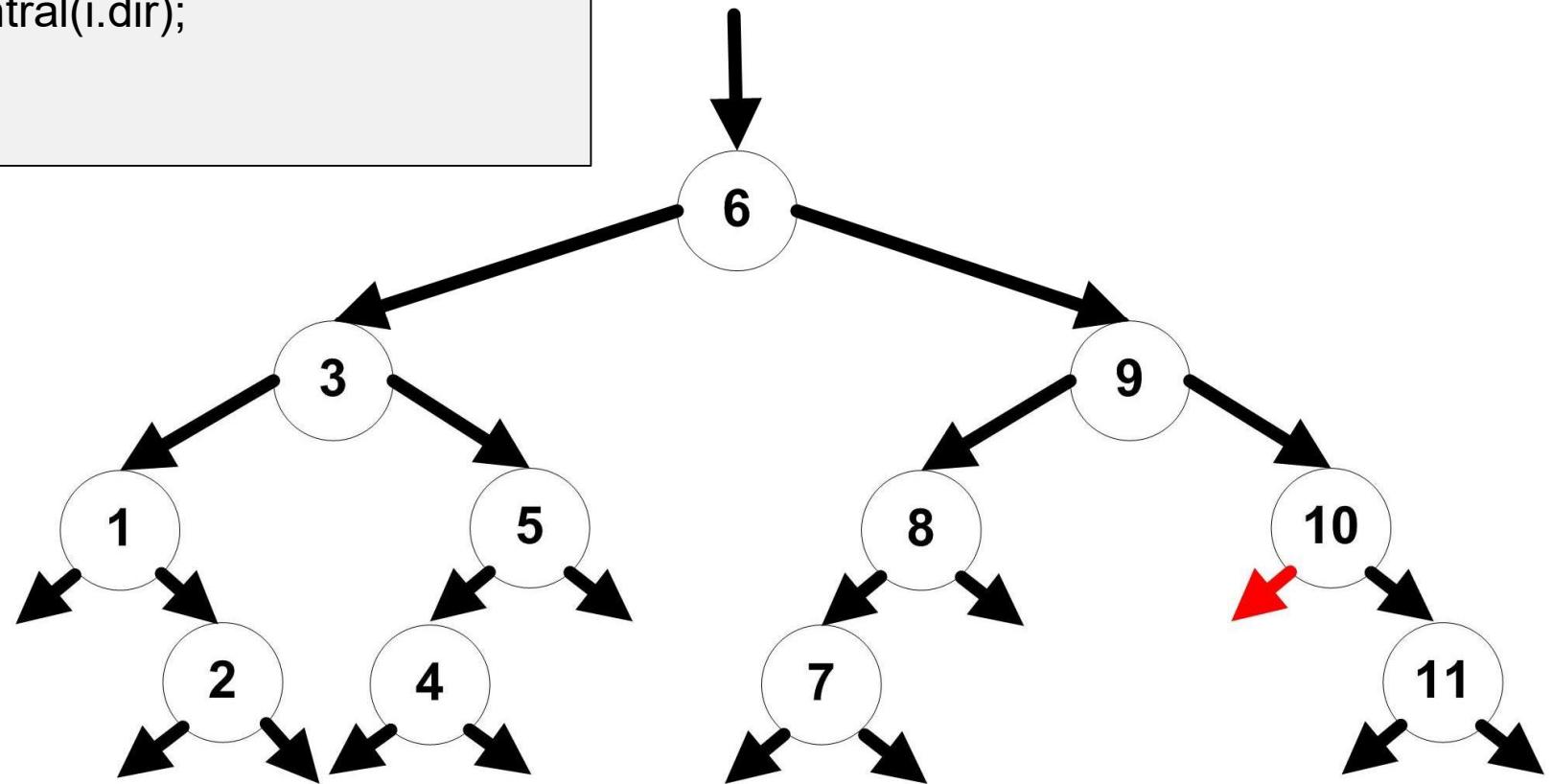
7

8

9

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

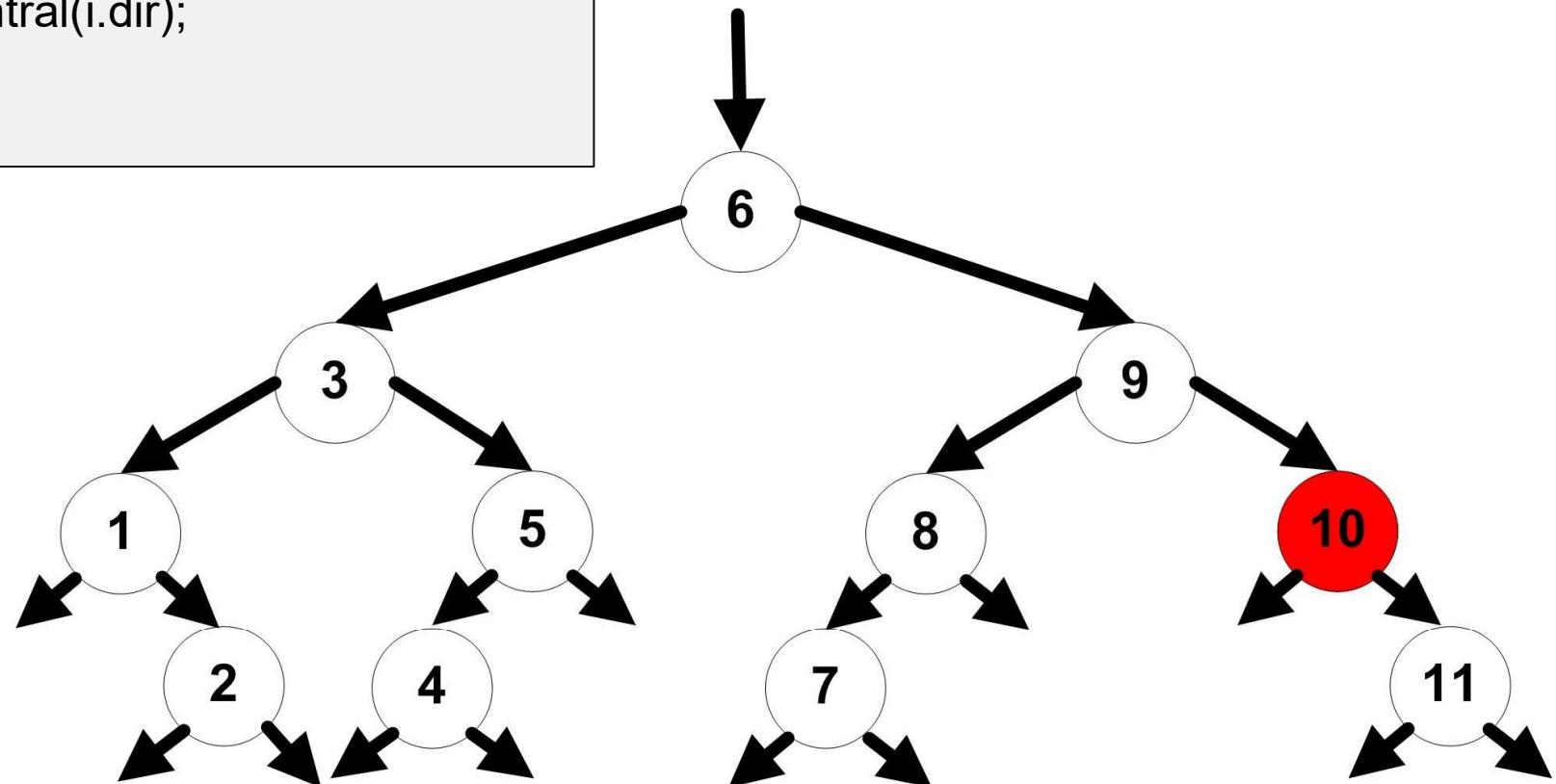


Tela

1 2 3 4 5 6 7 8 9

Caminhamento Central ou Em Ordem

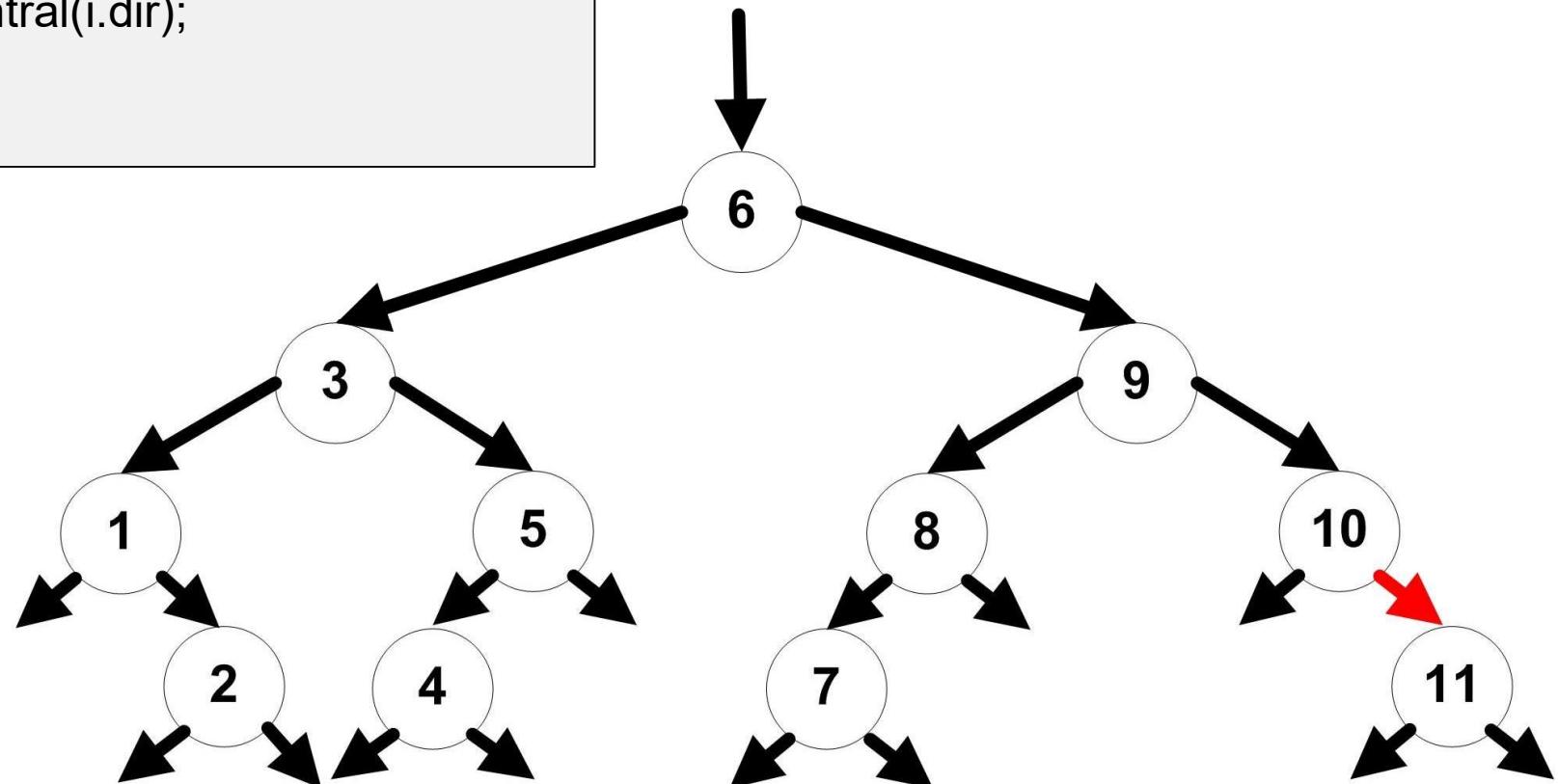
```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Tela	1	2	3	4	5	6	7	8	9	10
------	---	---	---	---	---	---	---	---	---	----

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

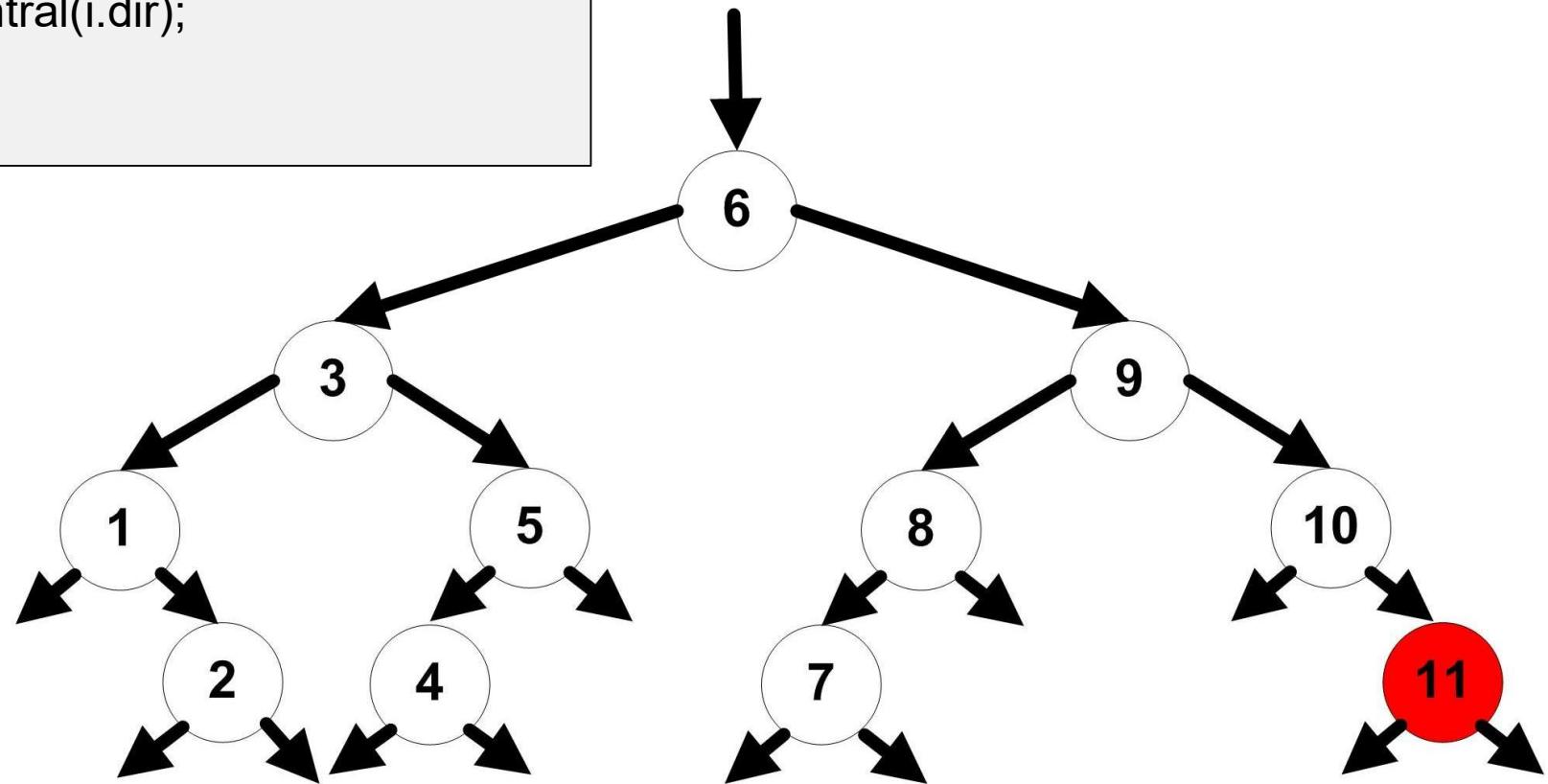


Tela

1 2 3 4 5 6 7 8 9 10

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

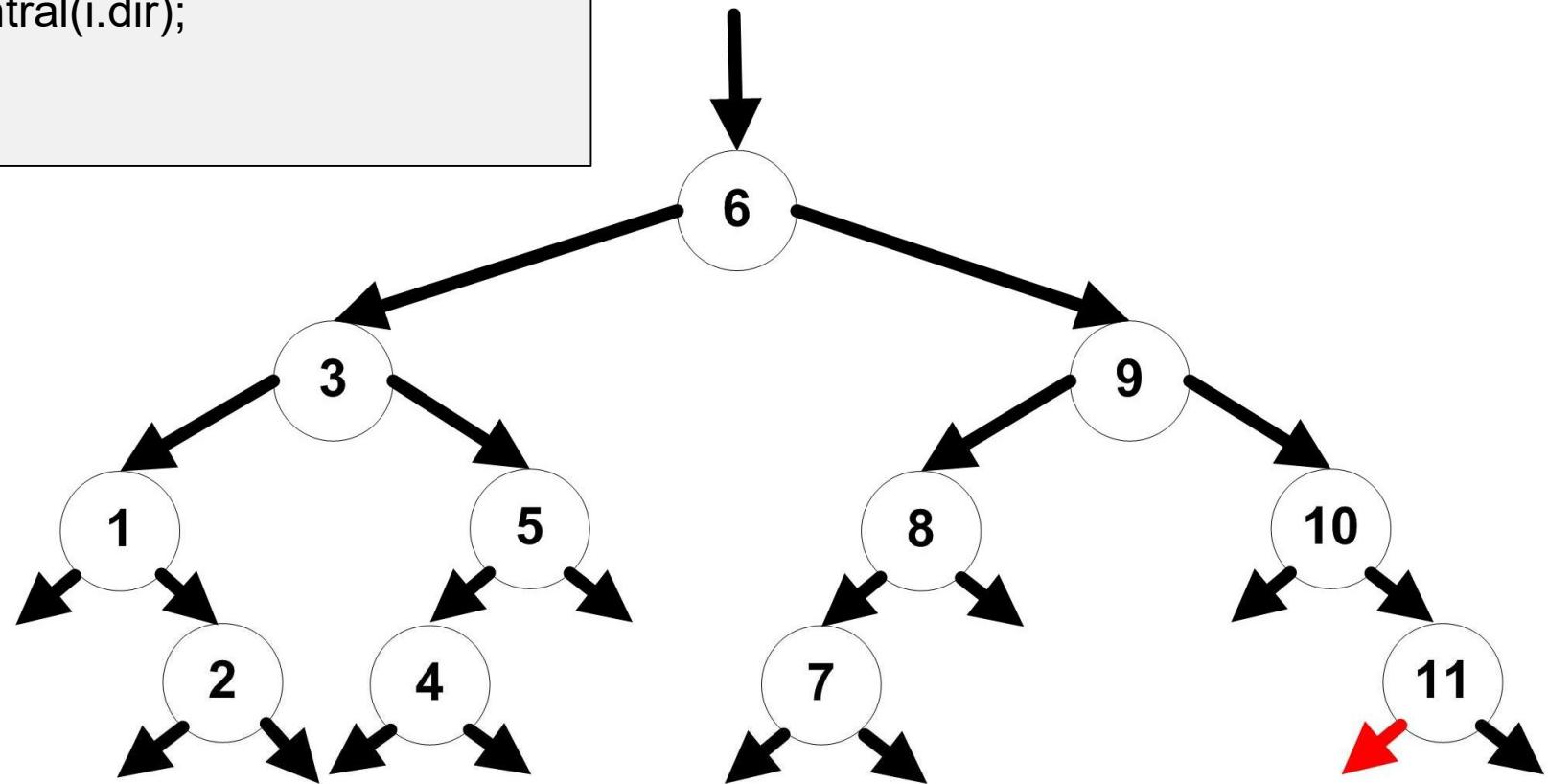


Tela

1 2 3 4 5 6 7 8 9 10

Caminhamento Central ou Em Ordem

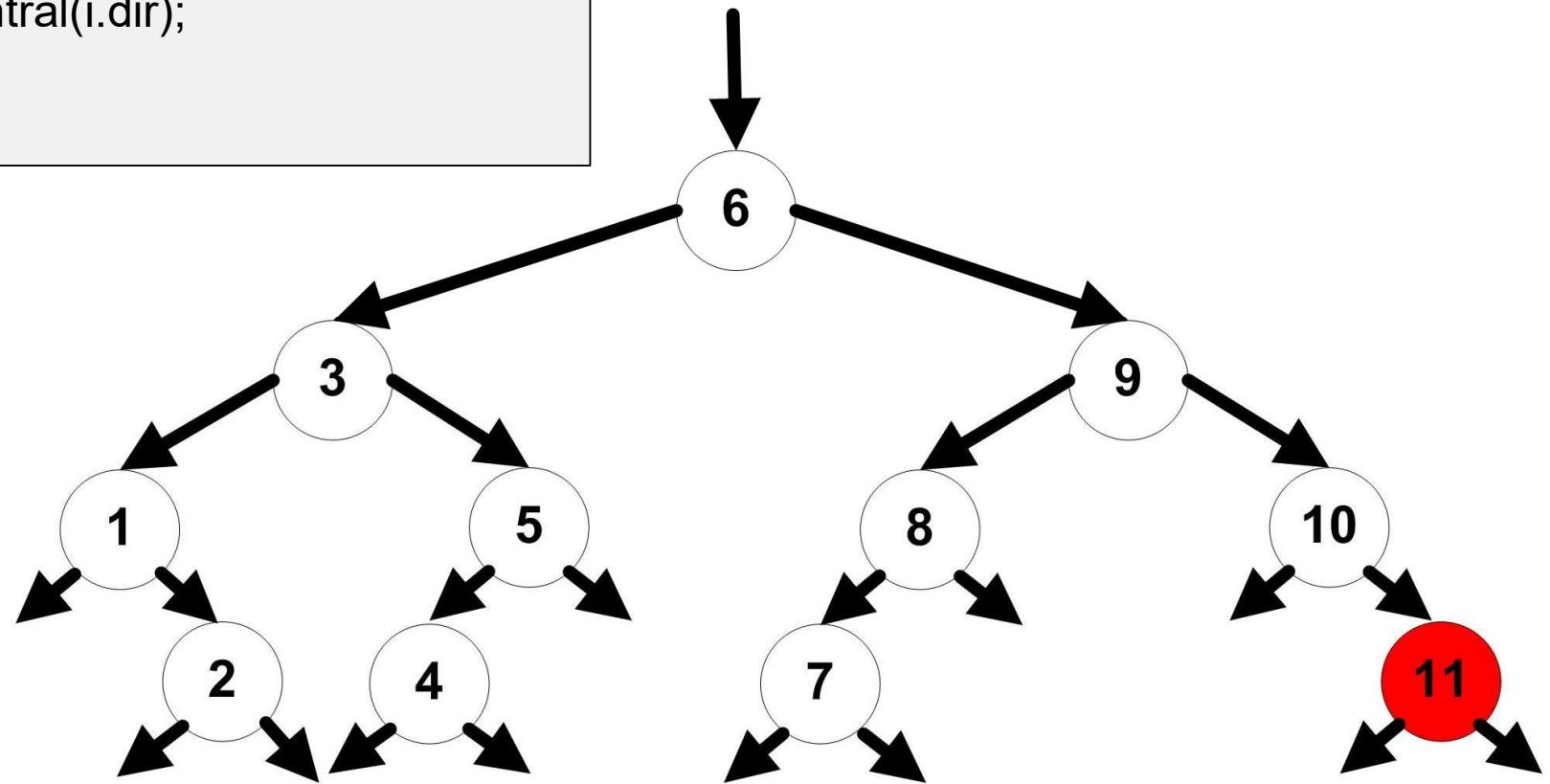
```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Tela	1	2	3	4	5	6	7	8	9	10

Caminhamento Central ou Em Ordem

```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```

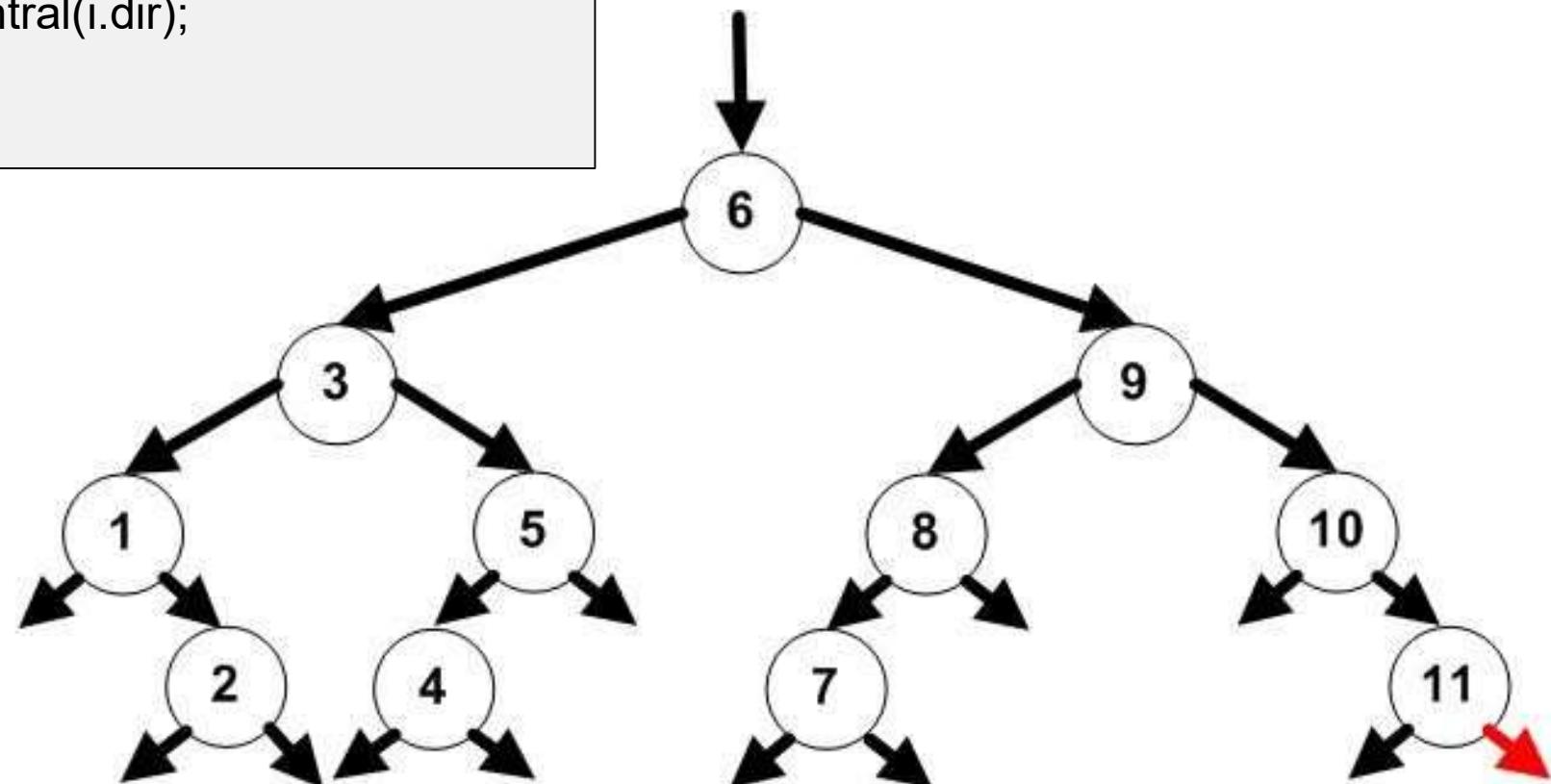


Tela

1 2 3 4 5 6 7 8 9 10 11

Caminhamento Central ou Em Ordem

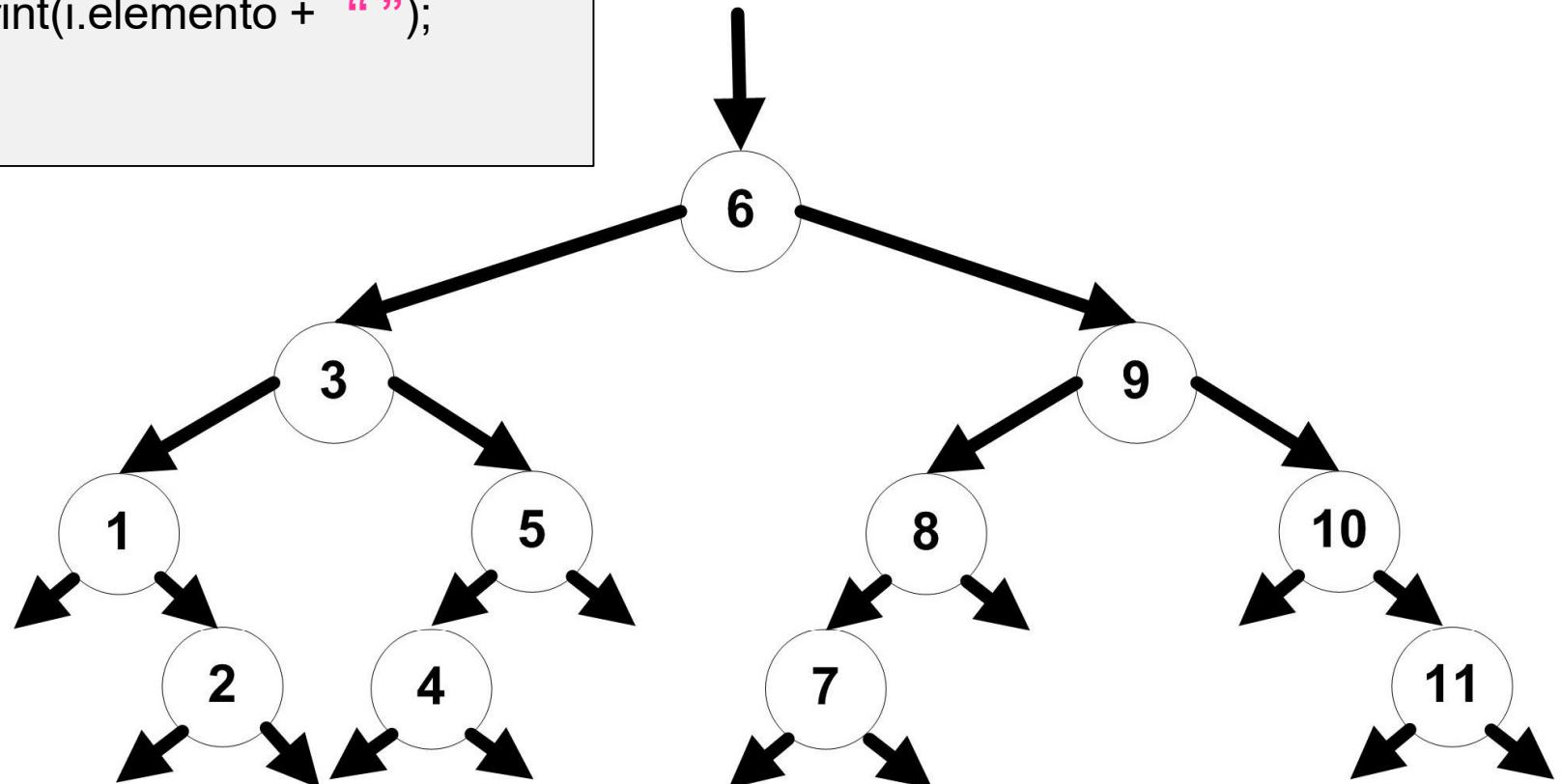
```
void caminharCentral(No i) {  
    if (i != null) {  
        caminharCentral(i.esq);  
        System.out.print(i.elemento + " ");  
        caminharCentral(i.dir);  
    }  
}
```



Tela	1	2	3	4	5	6	7	8	9	10	11

Caminhamento Pós-fixado ou Pós-ordem

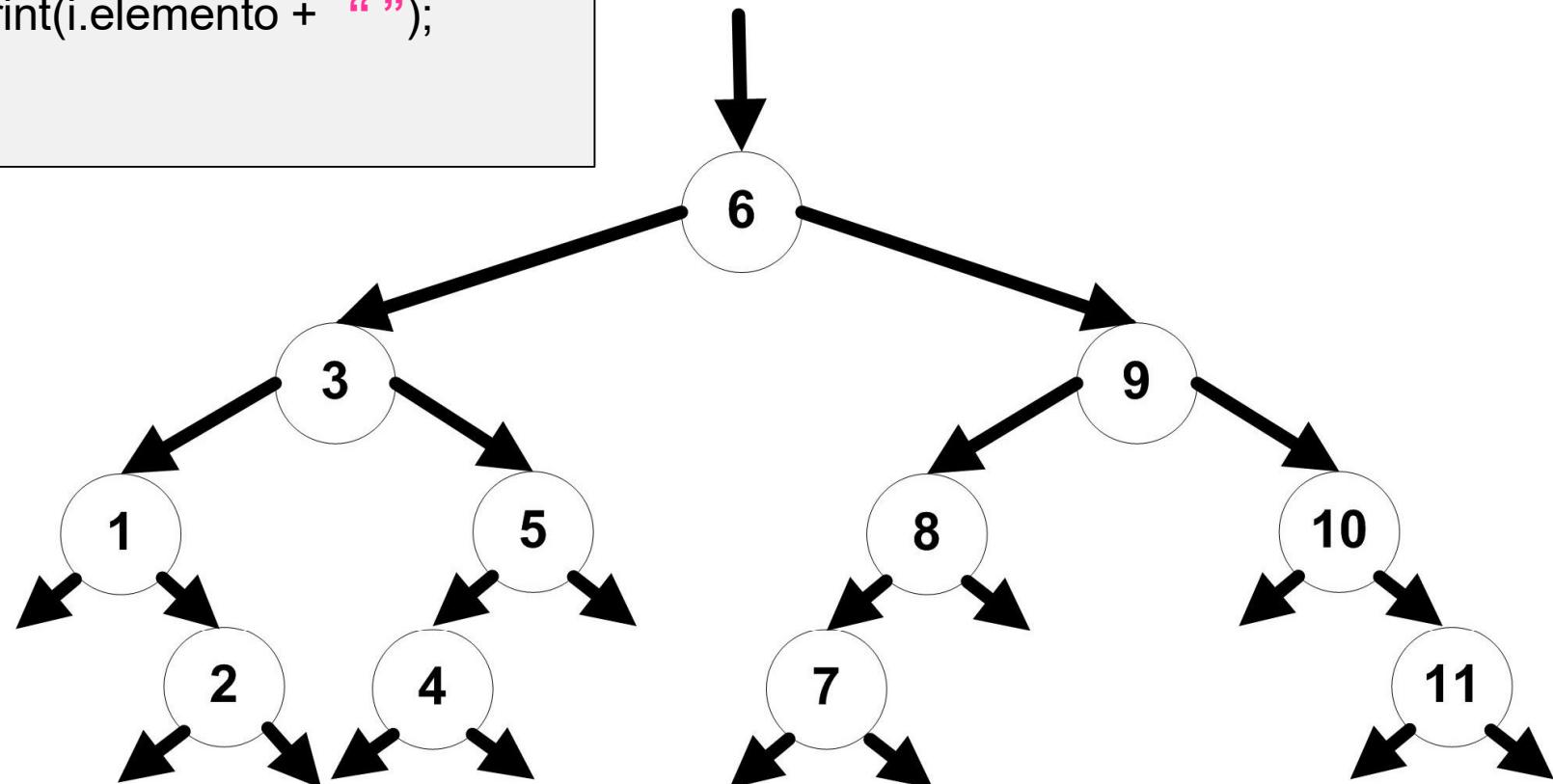
```
void caminharPos(No i) {  
    if (i != null) {  
        caminharPos(i.esq);  
        caminharPos(i.dir);  
        System.out.print(i.elemento + " ");  
    }  
}
```



Tela

Caminhamento Pós-fixado ou Pós-ordem

```
void caminharPos(No i) {  
    if (i != null) {  
        caminharPos(i.esq);  
        caminharPos(i.dir);  
        System.out.print(i.elemento + " ");  
    }  
}
```

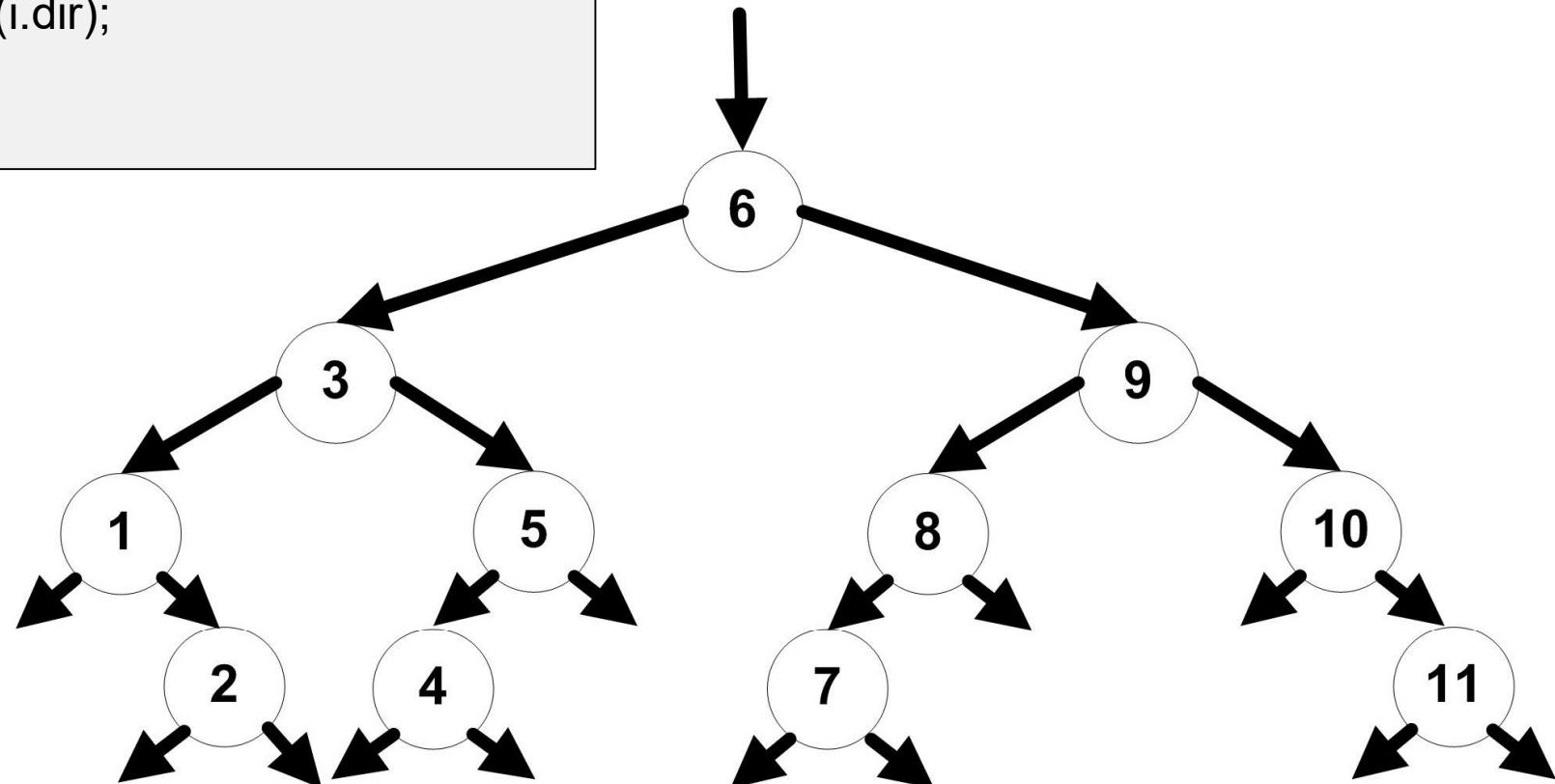


Tela

2 1 4 5 3 7 8 11 10 9 6

Caminhamento Pré-fixado ou Pré-ordem

```
void caminharPre(No i) {  
    if (i != null) {  
        System.out.print(i.elemento + " ");  
        caminharPre(i.esq);  
        caminharPre(i.dir);  
    }  
}
```

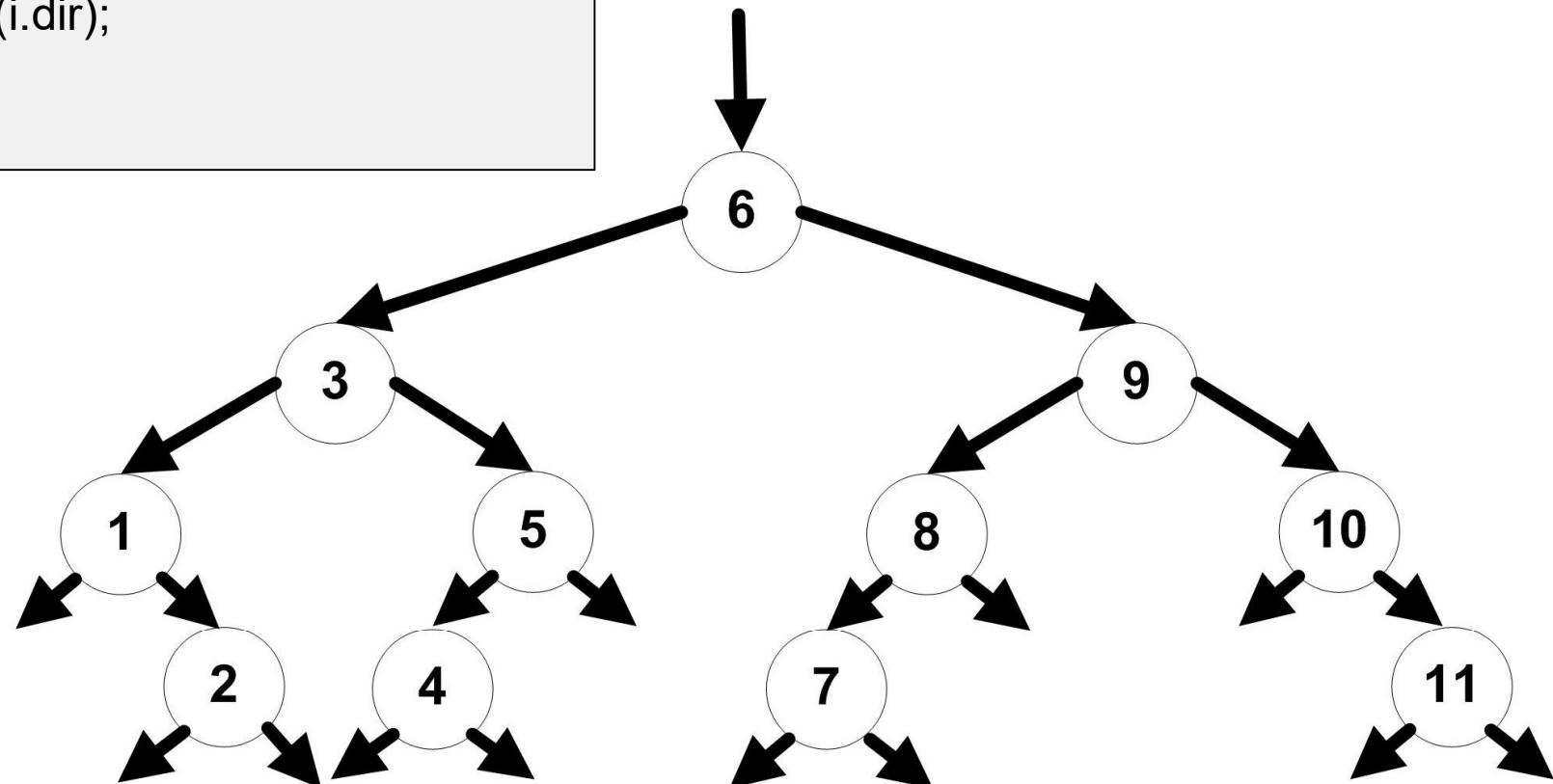


Tela

6 3 1 2 5 4 9 8 7 10 11

Caminhamento Pré-fixado ou Pré-ordem

```
void caminharPre(No i) {  
    if (i != null) {  
        System.out.print(i.elemento + " ");  
        caminharPre(i.esq);  
        caminharPre(i.dir);  
    }  
}
```



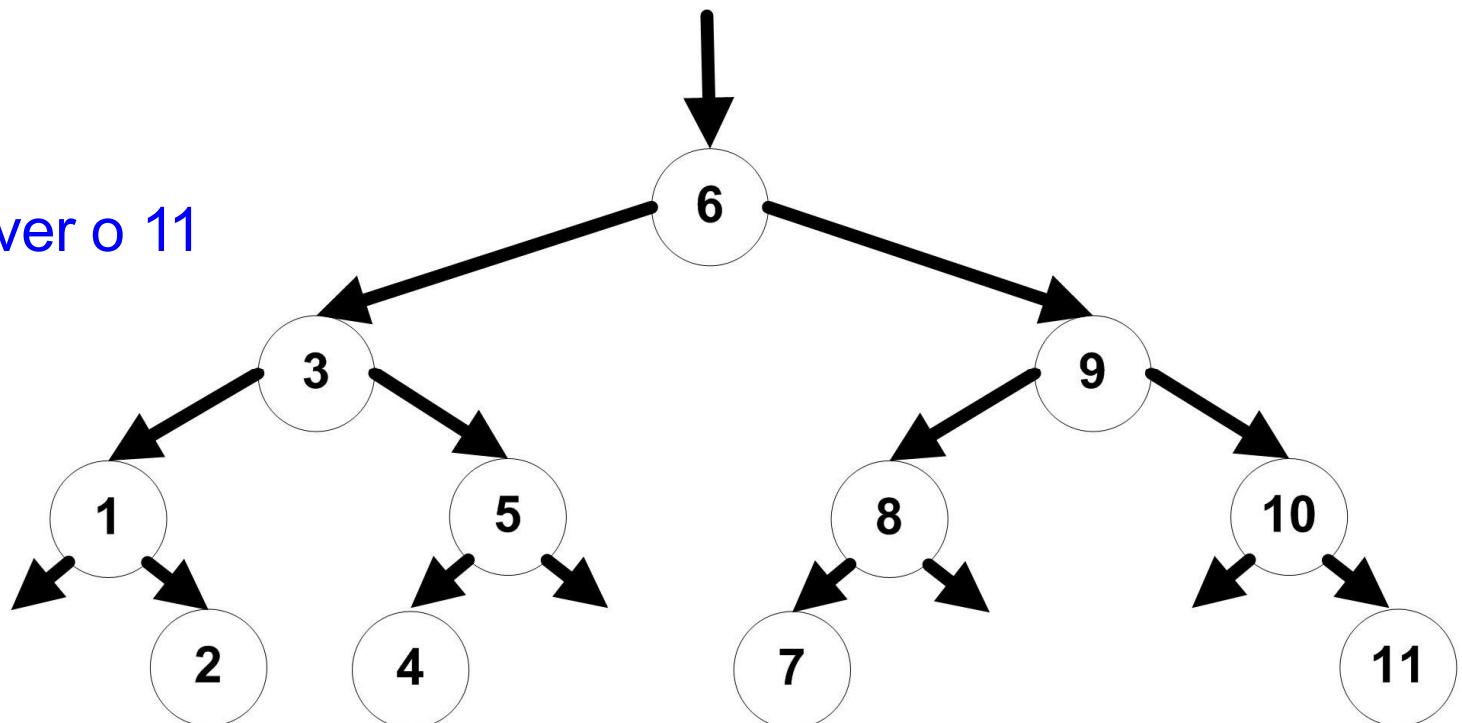
Tela

6 3 1 2 5 4 9 8 7 10 11

Funcionamento Básico da Remoção

(1) Se o elemento estiver em uma **folha**, removê-la

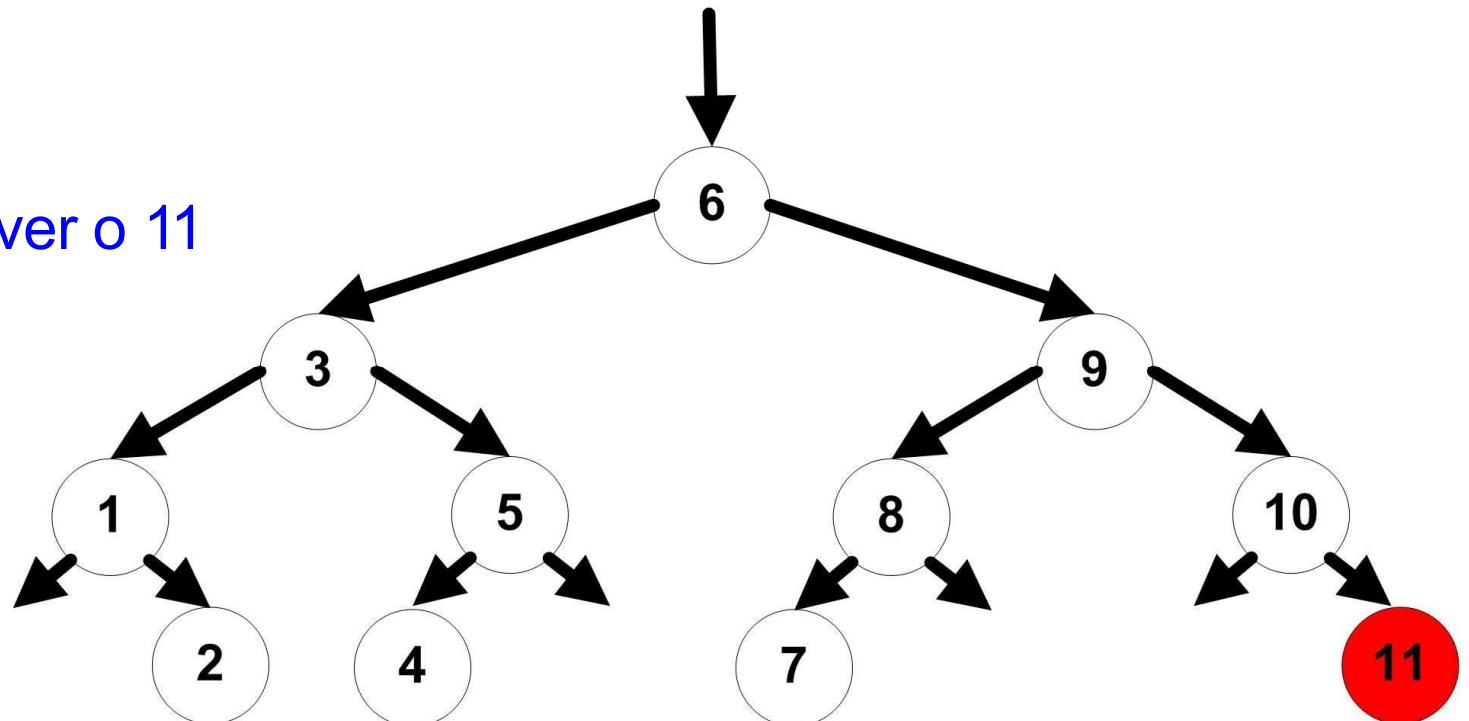
Exemplo: Remover o 11



Funcionamento Básico da Remoção

(1) Se o elemento estiver em uma **folha**, removê-la

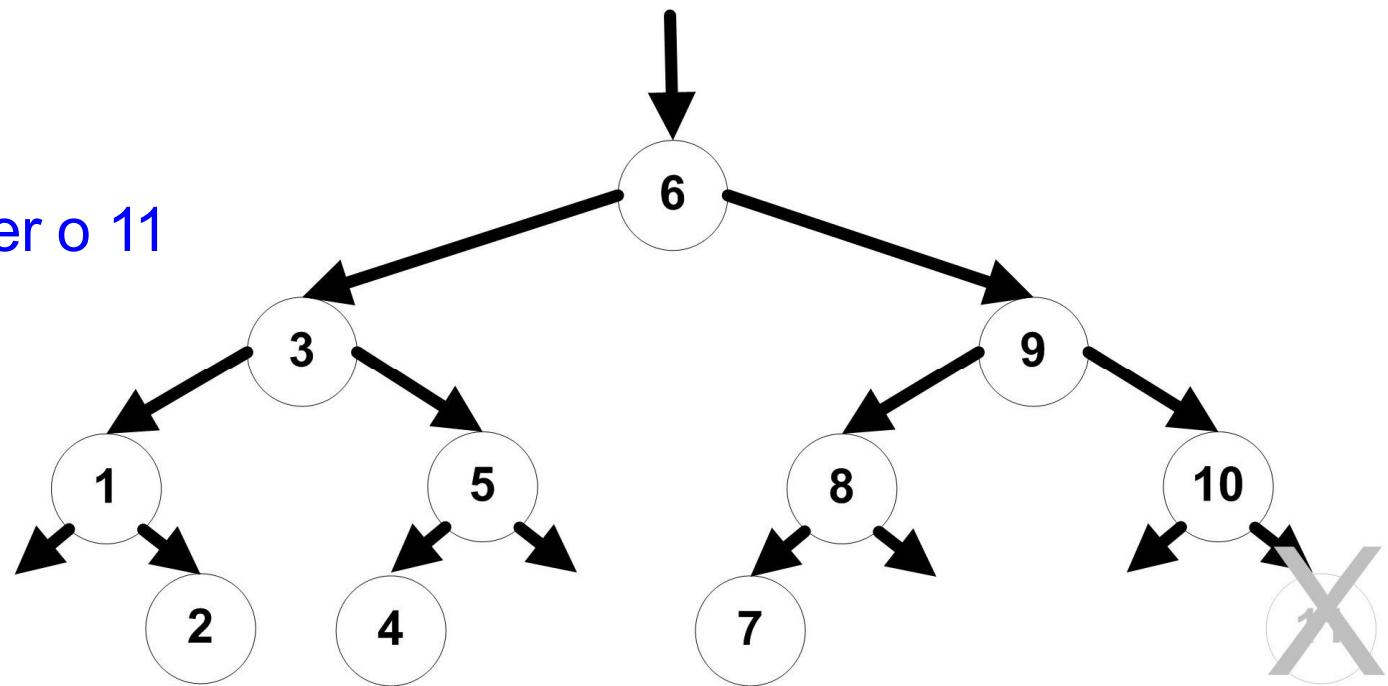
Exemplo: Remover o 11



Funcionamento Básico da Remoção

(1) Se o elemento estiver em uma **folha**, removê-la

Exemplo: Remover o 11



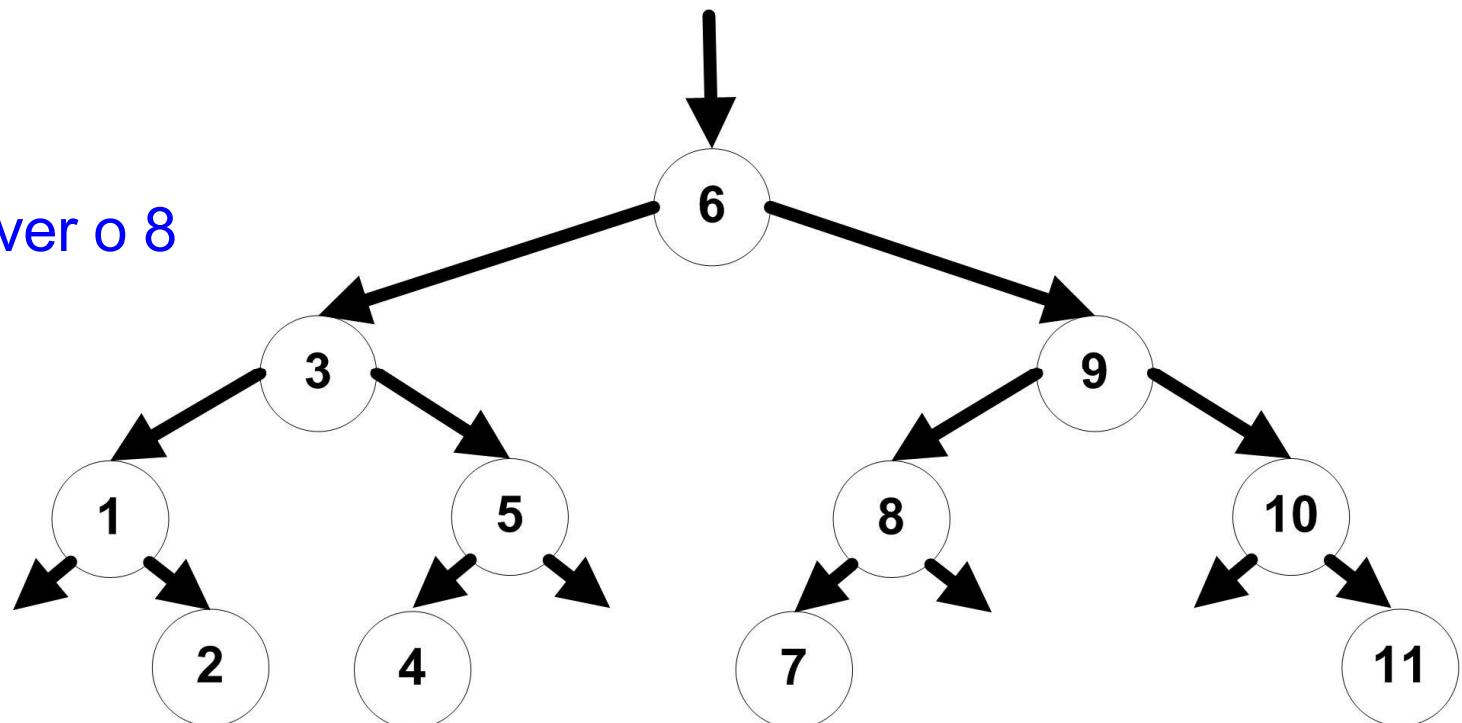
Funcionamento Básico da Remoção

(2) Senão, se o elemento estiver em um **nó intermediário com um único filho**, remover o nó e fazer com que seu pai aponte para seu filho

Funcionamento Básico da Remoção

(2) Senão, se o elemento estiver em um **nó intermediário com um único filho**, remover o nó e fazer com que seu pai aponte para seu filho

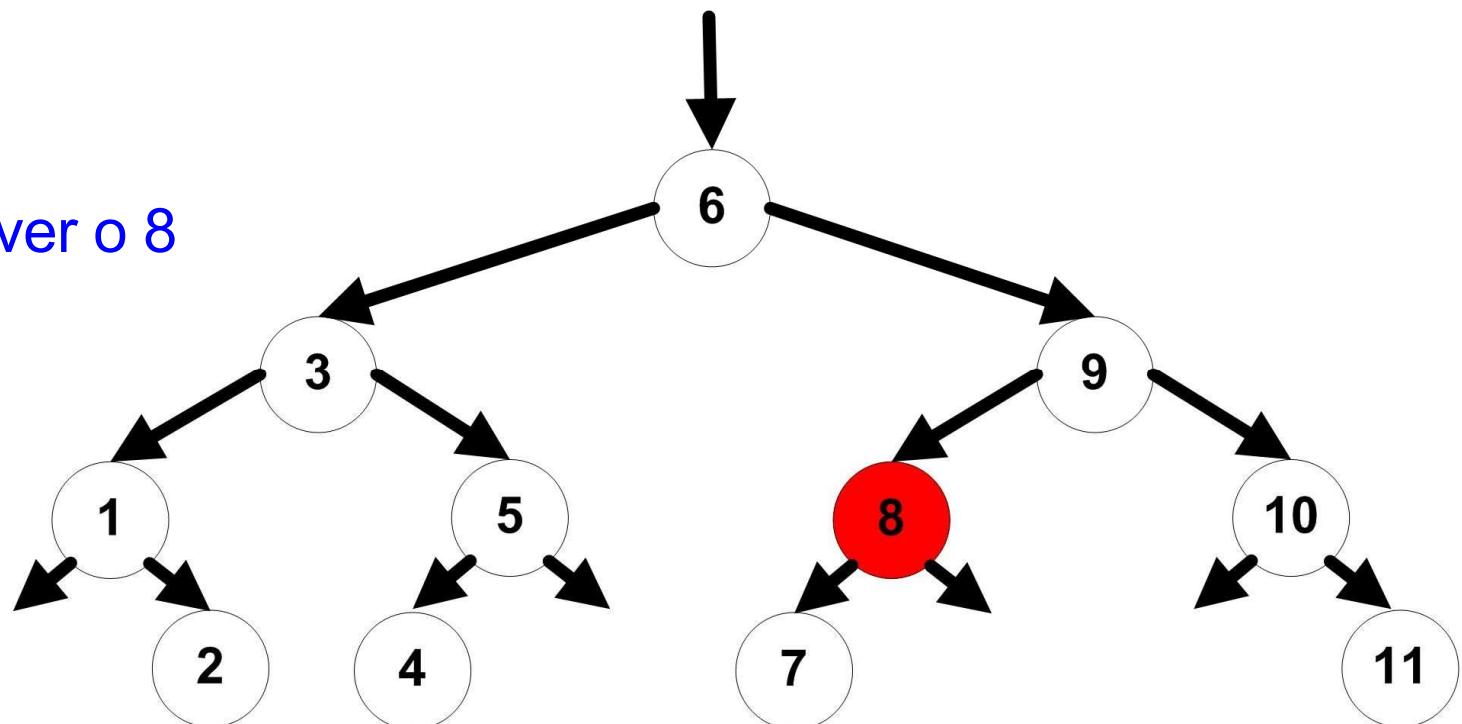
Exemplo: Remover o 8



Funcionamento Básico da Remoção

(2) Senão, se o elemento estiver em um **nó intermediário com um único filho**, remover o nó e fazer com que seu pai aponte para seu filho

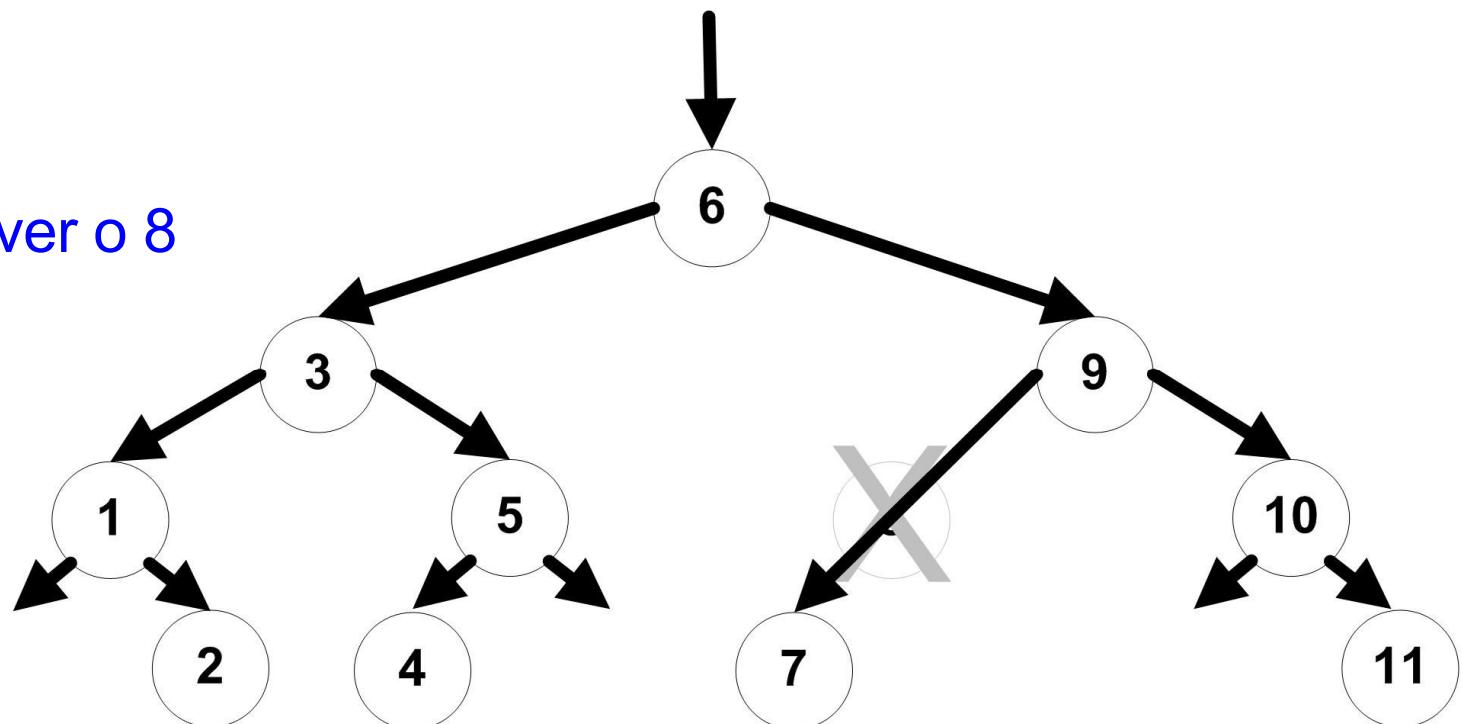
Exemplo: Remover o 8



Funcionamento Básico da Remoção

(2) Senão, se o elemento estiver em um **nó intermediário com um único filho**, remover o nó e fazer com que seu pai aponte para seu filho

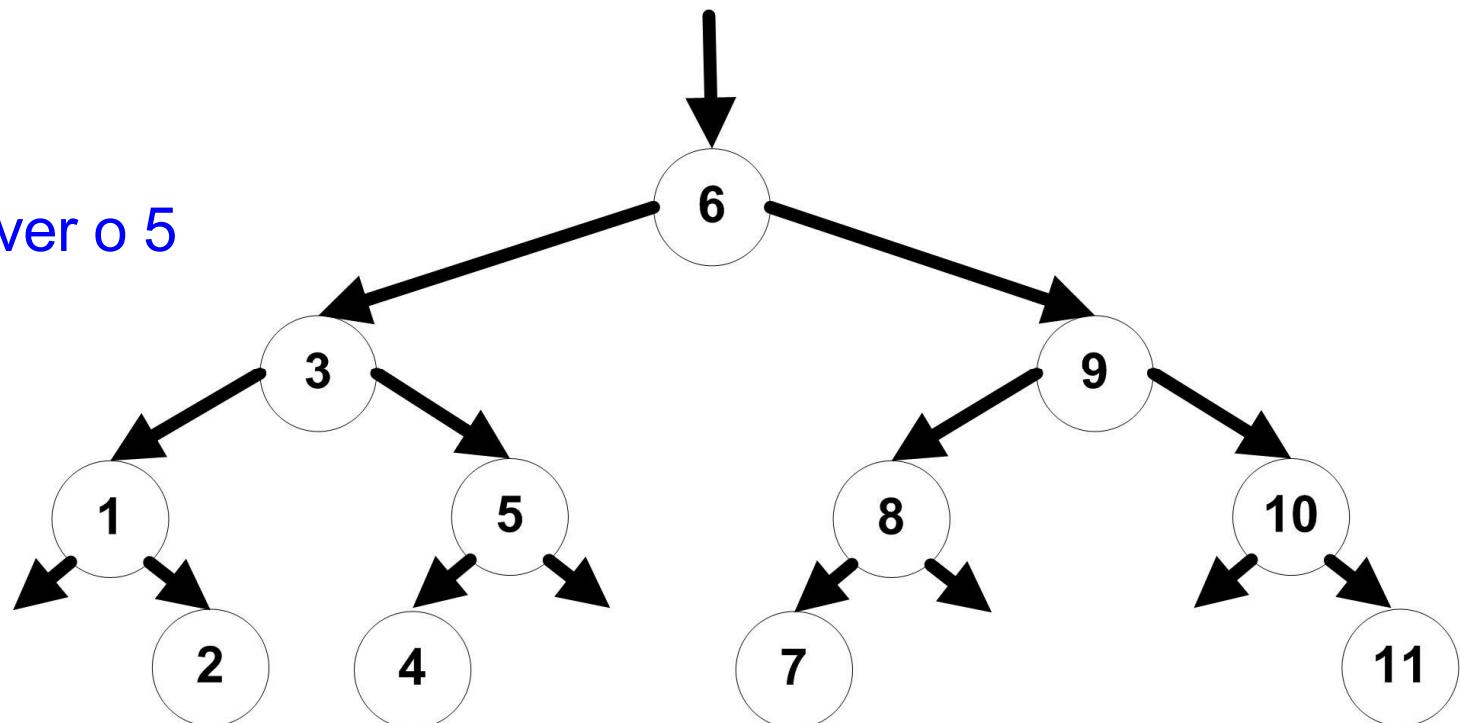
Exemplo: Remover o 8



Funcionamento Básico da Remoção

(2) Senão, se o elemento estiver em um **nó intermediário com um único filho**, remover o nó e fazer com que seu pai aponte para seu filho

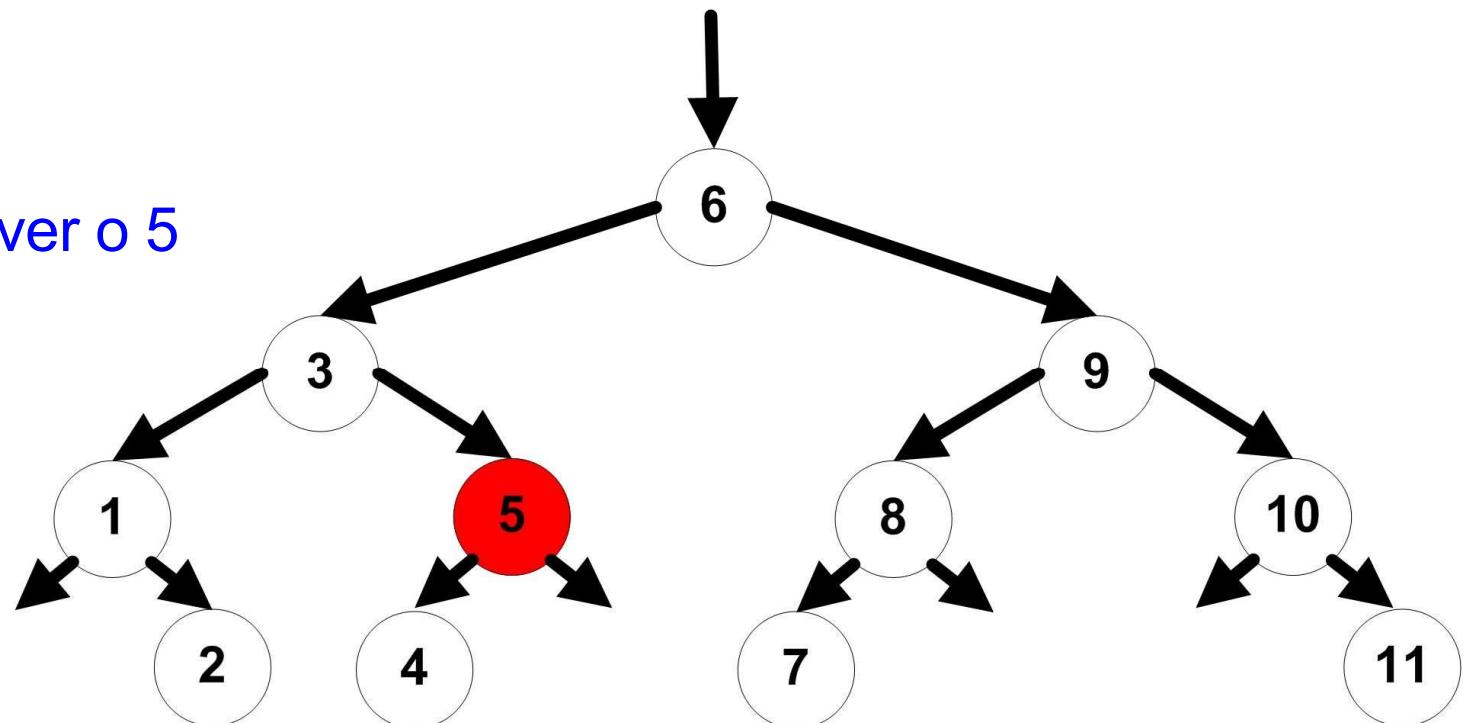
Exemplo: Remover o 5



Funcionamento Básico da Remoção

(2) Senão, se o elemento estiver em um **nó intermediário com um único filho**, remover o nó e fazer com que seu pai aponte para seu filho

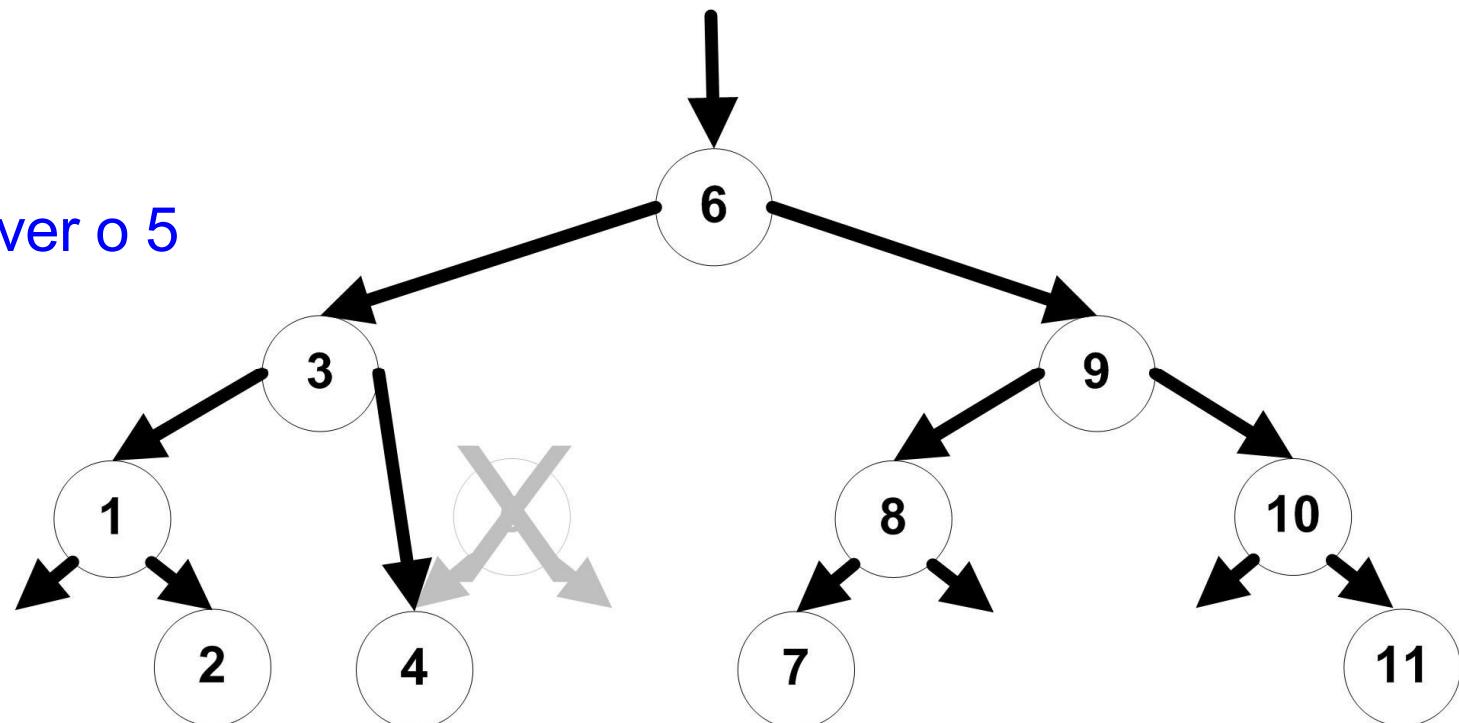
Exemplo: Remover o 5



Funcionamento Básico da Remoção

(2) Senão, se o elemento estiver em um **nó intermediário com um único filho**, remover o nó e fazer com que seu pai aponte para seu filho

Exemplo: Remover o 5



Funcionamento Básico da Remoção

(3) Senão, se o elemento estiver em um **nó intermediário com dois filhos**, o elemento a ser removido deve ser substituído ou pelo **maior nó da subárvore à esquerda** ou **menor nó da subárvore à direita**

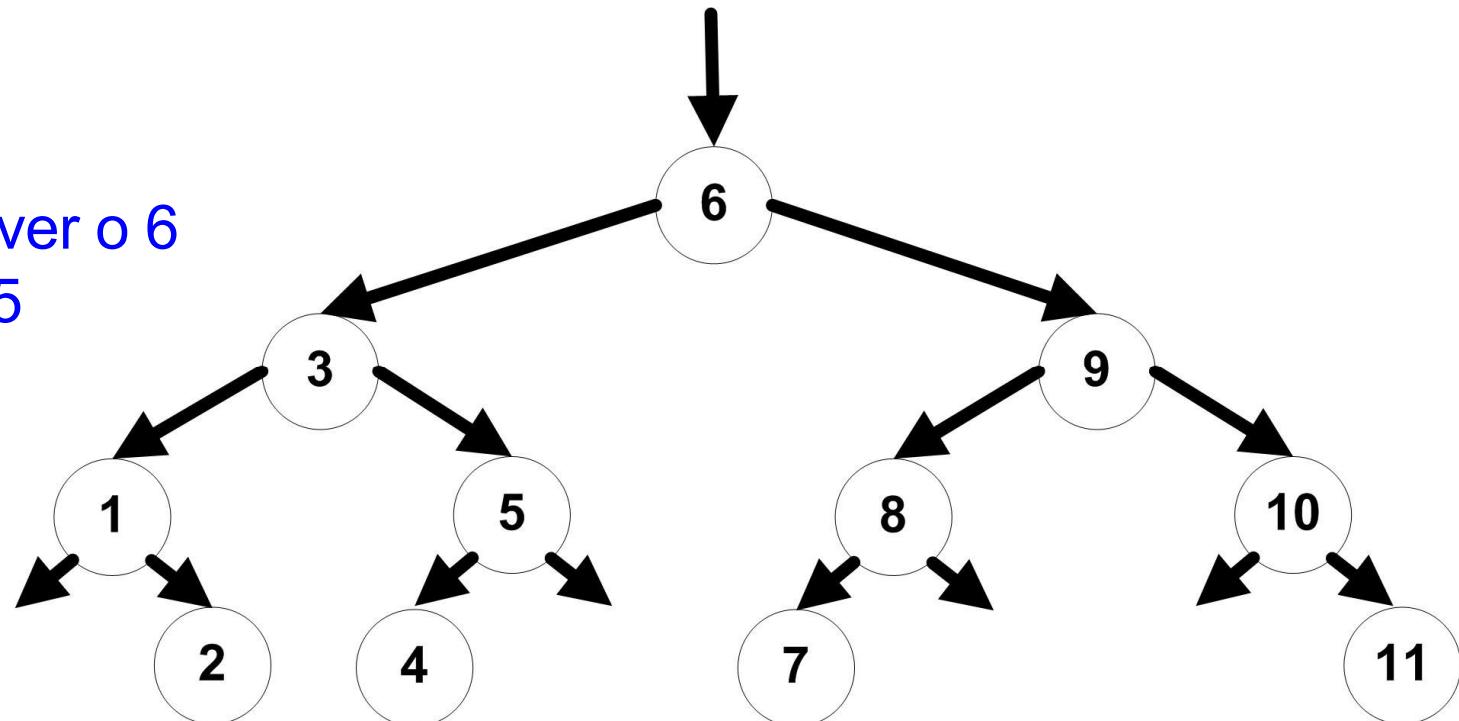
Funcionamento Básico da Remoção

(3) Senão, se o elemento estiver em um **nó intermediário com dois filhos**, o elemento a ser removido deve ser substituído ou pelo **maior nó da subárvore à esquerda** ou **menor nó da subárvore à direita**

Funcionamento Básico da Remoção

(3) Senão, se o elemento estiver em um **nó intermediário com dois filhos**, o elemento a ser removido deve ser substituído ou pelo **maior nó da subárvore à esquerda** ou **menor nó da subárvore à direita**

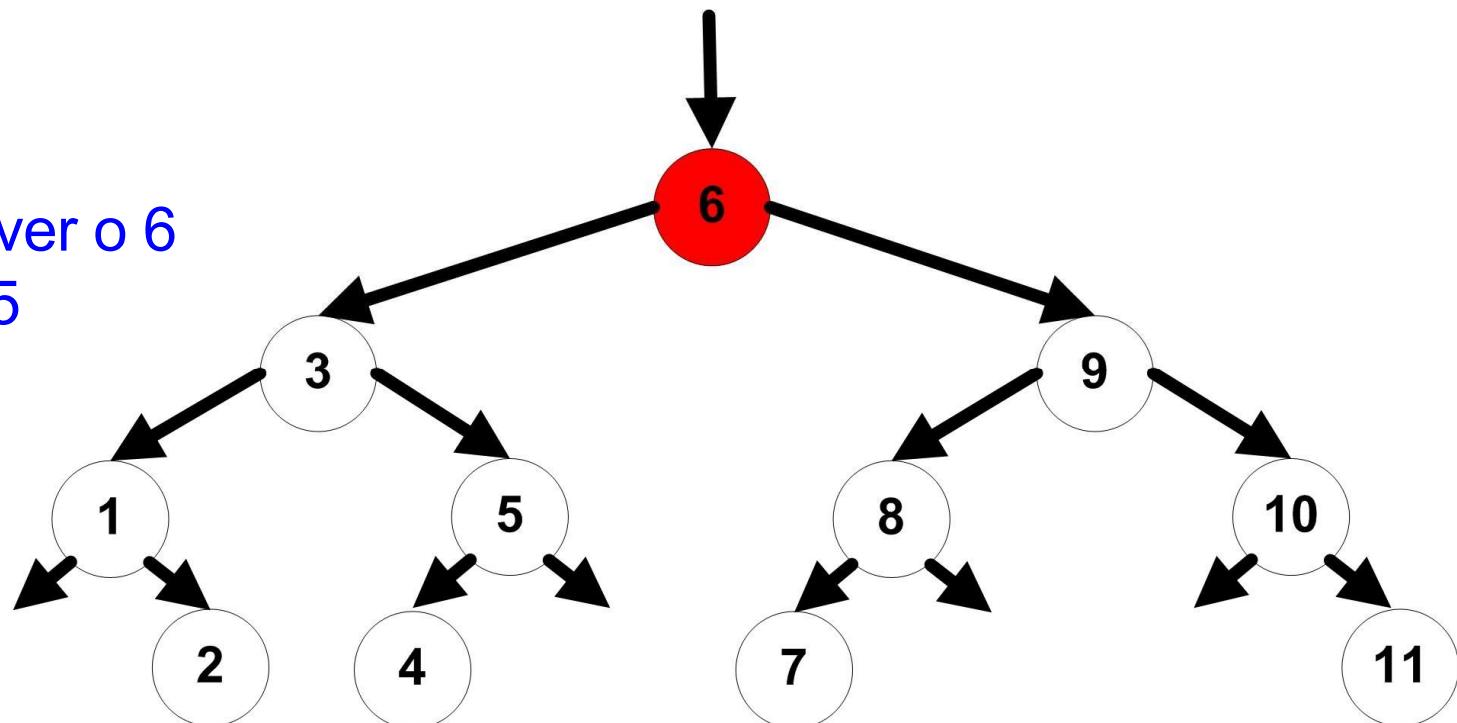
Exemplo: Remover o 6
e substituir pelo 5



Funcionamento Básico da Remoção

(3) Senão, se o elemento estiver em um **nó intermediário com dois filhos**, o elemento a ser removido deve ser substituído ou pelo **maior nó da subárvore à esquerda** ou **menor nó da subárvore à direita**

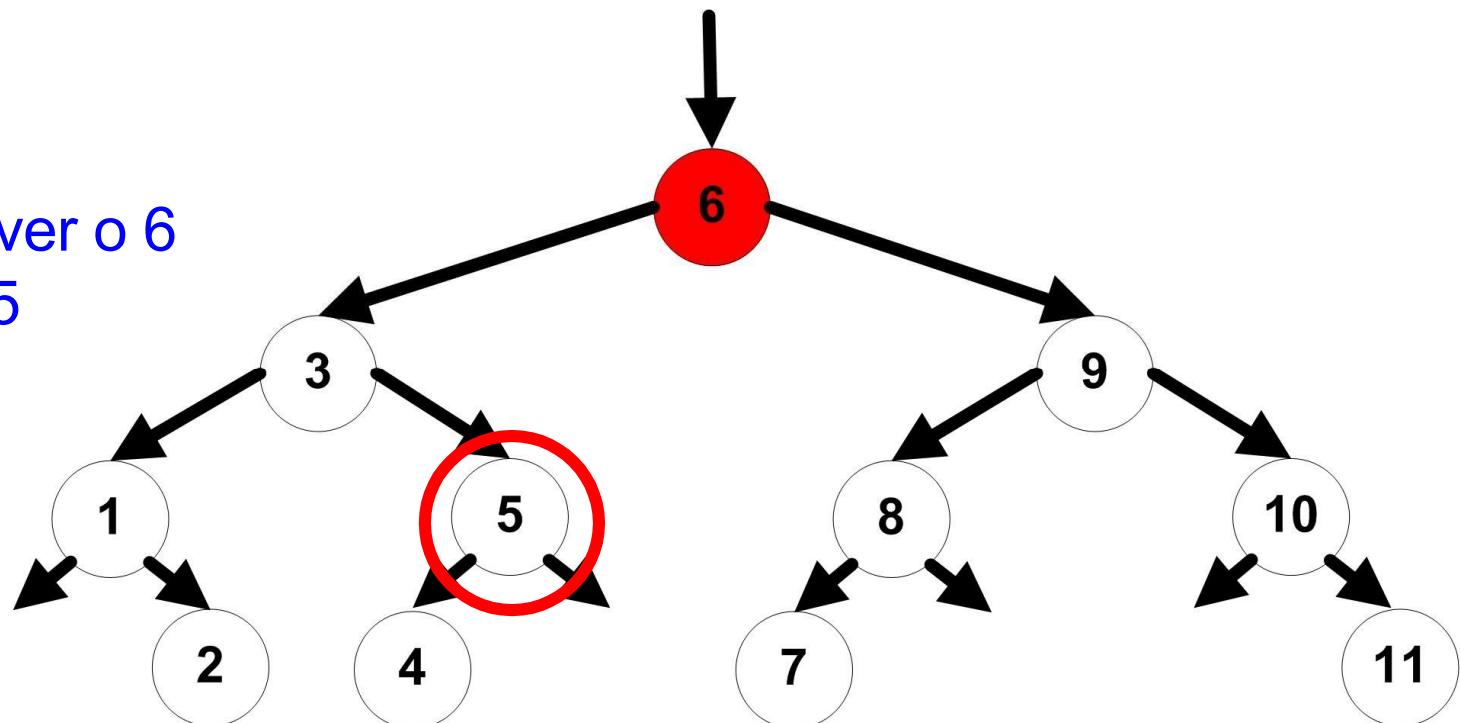
Exemplo: Remover o 6
e substituir pelo 5



Funcionamento Básico da Remoção

(3) Senão, se o elemento estiver em um **nó intermediário com dois filhos**, o elemento a ser removido deve ser substituído ou pelo **maior nó da subárvore à esquerda** ou **menor nó da subárvore à direita**

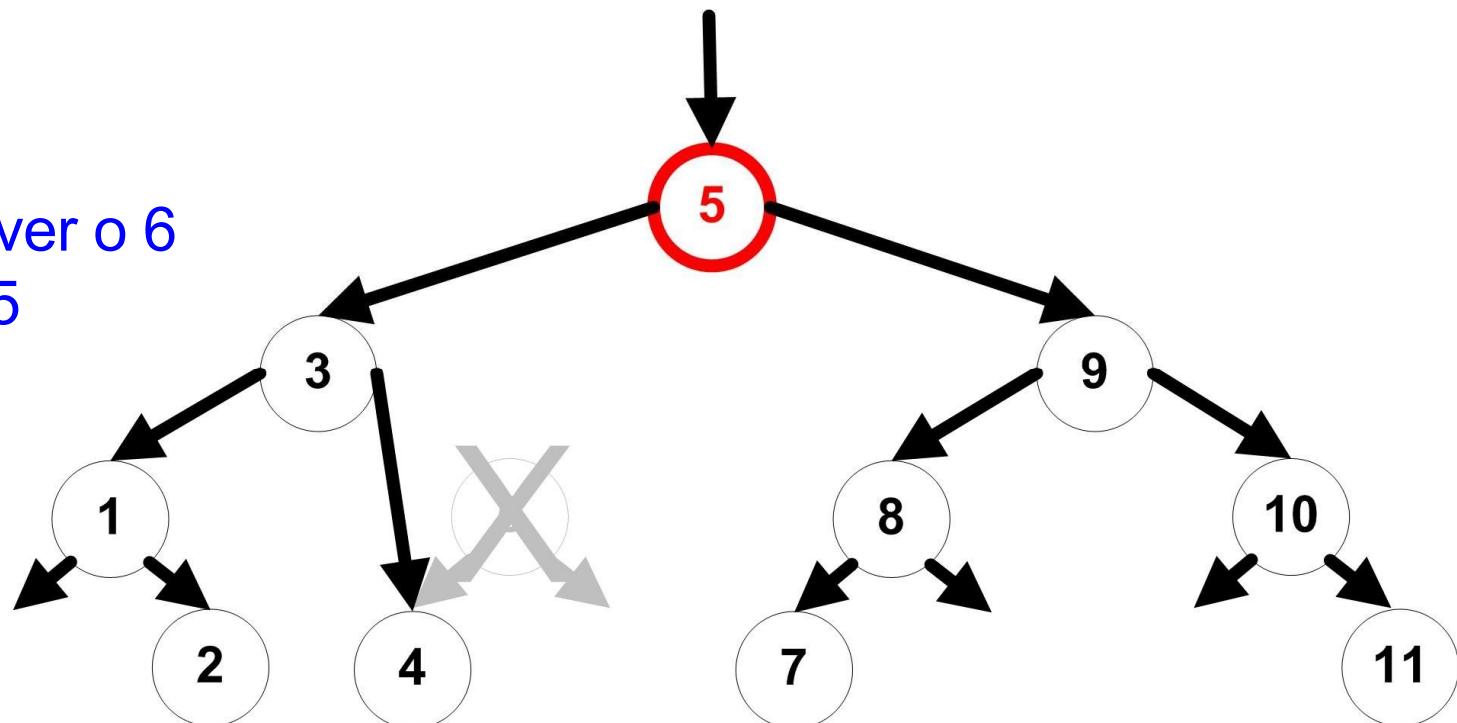
Exemplo: Remover o 6
e substituir pelo 5



Funcionamento Básico da Remoção

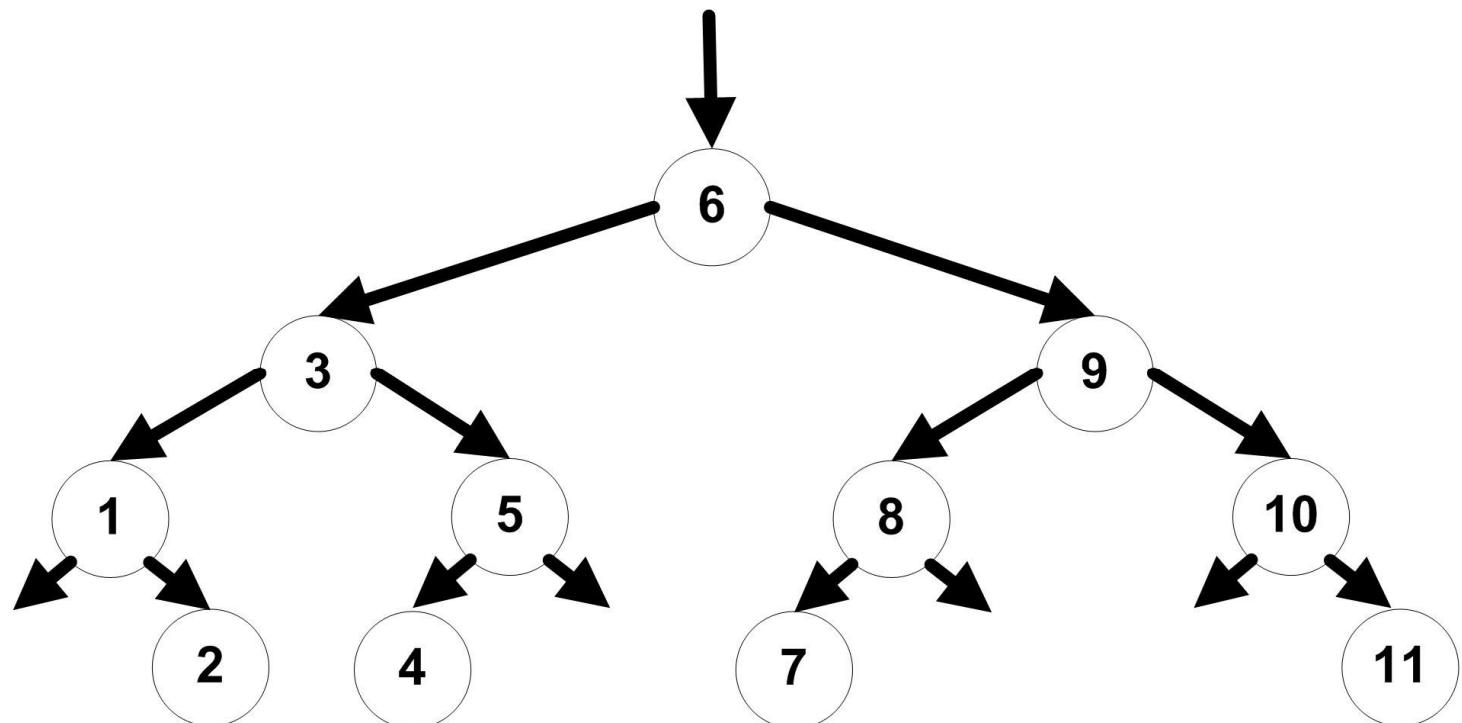
(3) Senão, se o elemento estiver em um **nó intermediário com dois filhos**, o elemento a ser removido deve ser substituído ou pelo **maior nó da subárvore à esquerda** ou **menor nó da subárvore à direita**

Exemplo: Remover o 6 e substituir pelo 5



Funcionamento Básico da Remoção

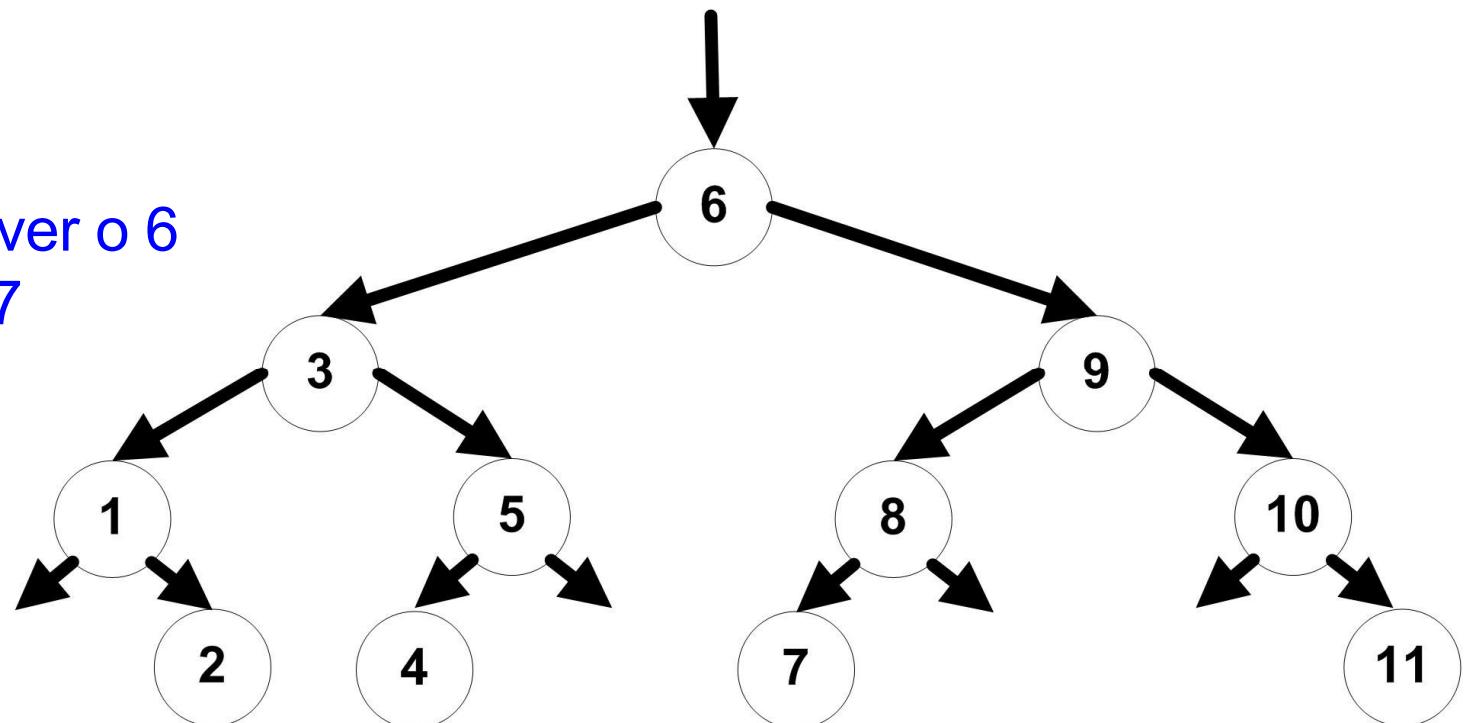
(3) Senão, se o elemento estiver em um **nó intermediário com dois filhos**, o elemento a ser removido deve ser substituído ou pelo **maior nó da subárvore à esquerda** ou **menor nó da subárvore à direita**



Funcionamento Básico da Remoção

(3) Senão, se o elemento estiver em um **nó intermediário com dois filhos**, o elemento a ser removido deve ser substituído ou pelo **maior nó da subárvore à esquerda** ou **menor nó da subárvore à direita**

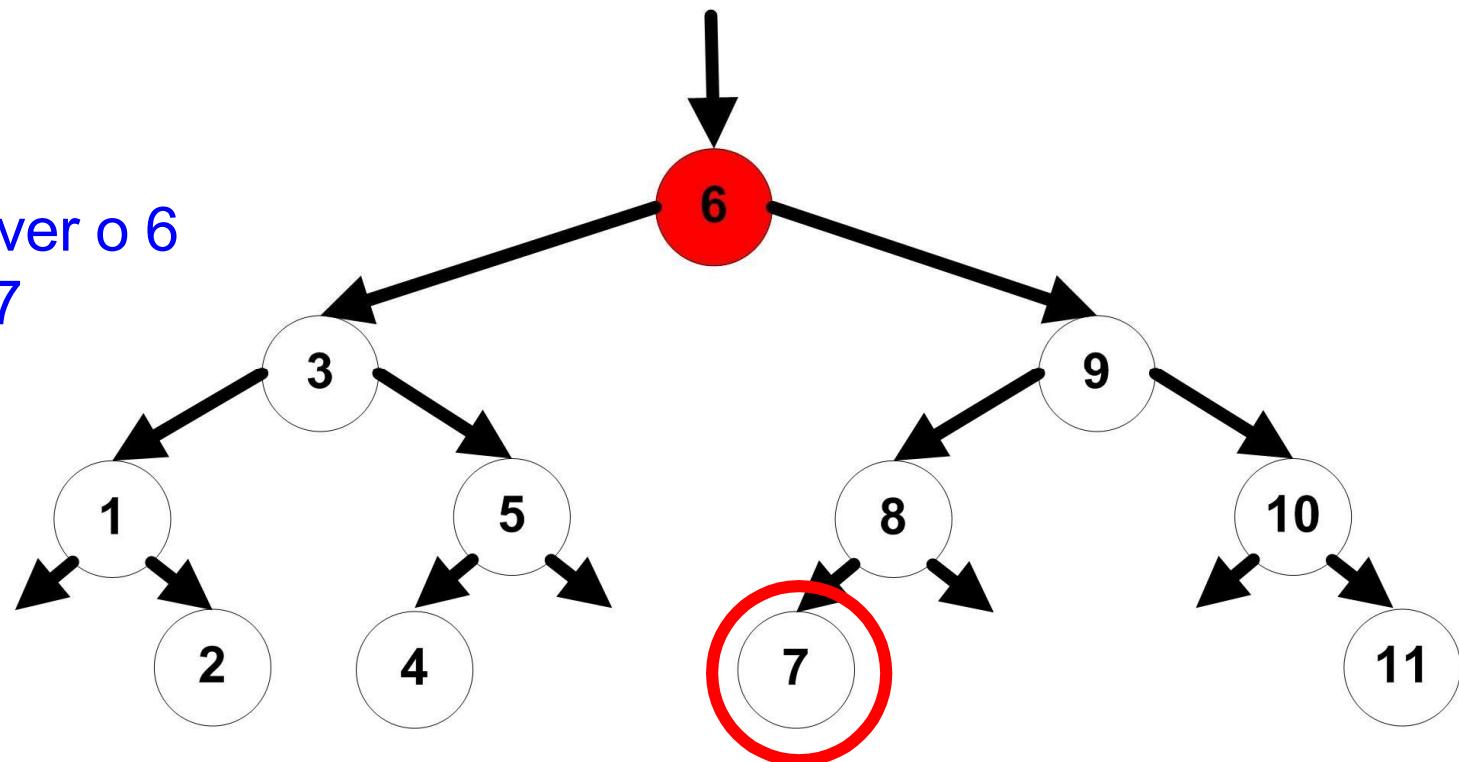
Exemplo: Remover o 6 e substituir pelo 7



Funcionamento Básico da Remoção

(3) Senão, se o elemento estiver em um **nó intermediário com dois filhos**, o elemento a ser removido deve ser substituído ou pelo **maior nó da subárvore à esquerda** ou **menor nó da subárvore à direita**

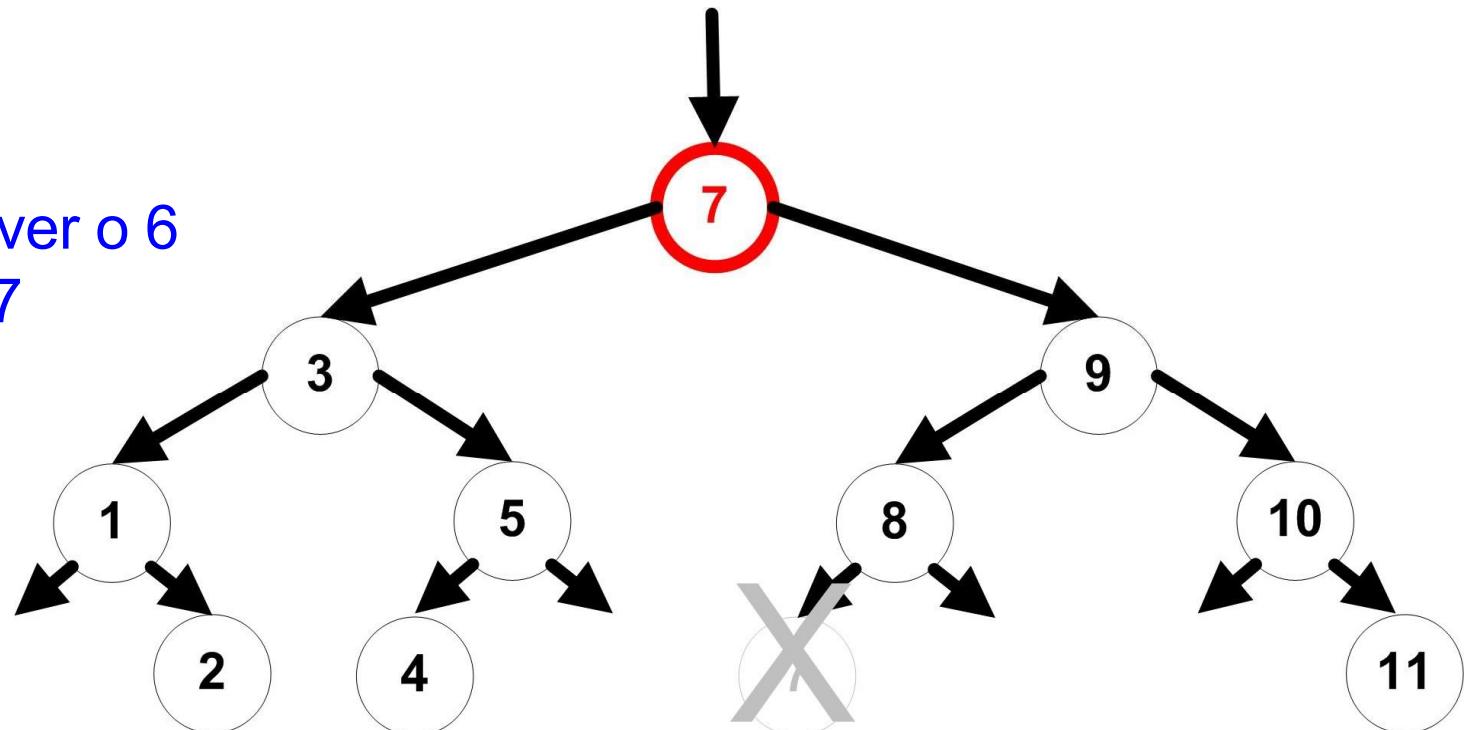
Exemplo: Remover o 6
e substituir pelo 7



Funcionamento Básico da Remoção

(3) Senão, se o elemento estiver em um **nó intermediário com dois filhos**, o elemento a ser removido deve ser substituído ou pelo **maior nó da subárvore à esquerda** ou **menor nó da subárvore à direita**

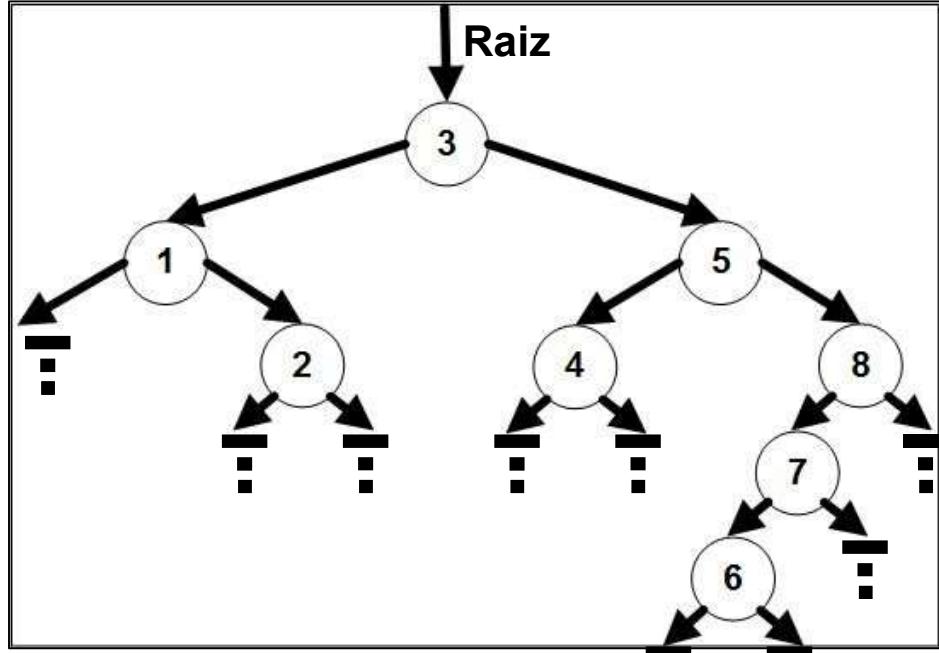
Exemplo: Remover o 6
e substituir pelo 7



Classe Árvore Binária: Remoção em Java

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

raiz
n(3)

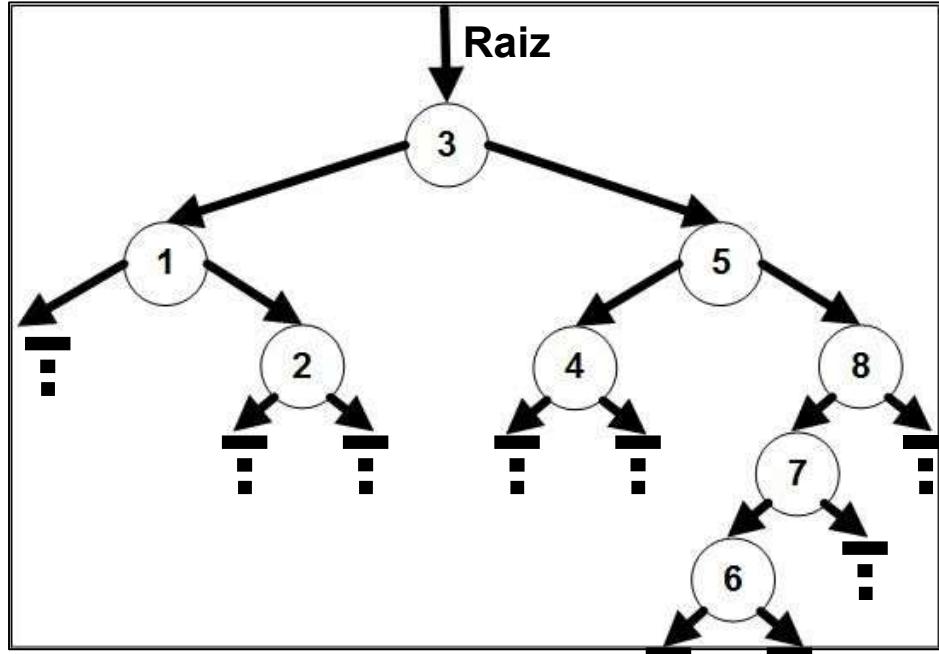


Algoritmo de Remoção em Java

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

raiz
n(3)

Vamos remover o 2 (uma folha) de nossa árvore



Algoritmo de Remoção em Java

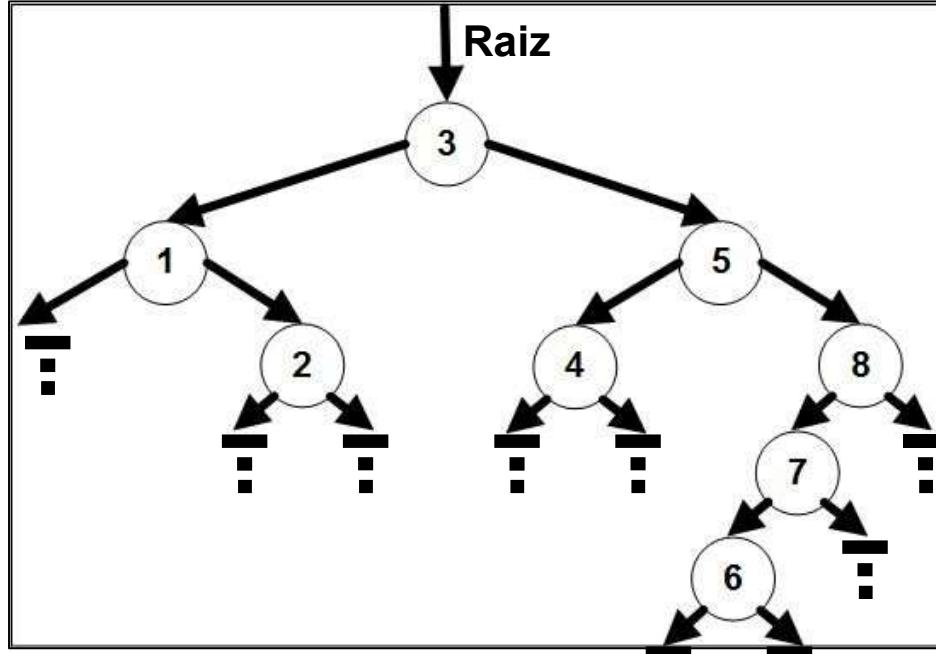
```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    else if(i.dir == null) { i = i.esq; }
    else if(i.esq == null) { i = i.dir; }
    else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2



Algoritmo de Remoção em Java

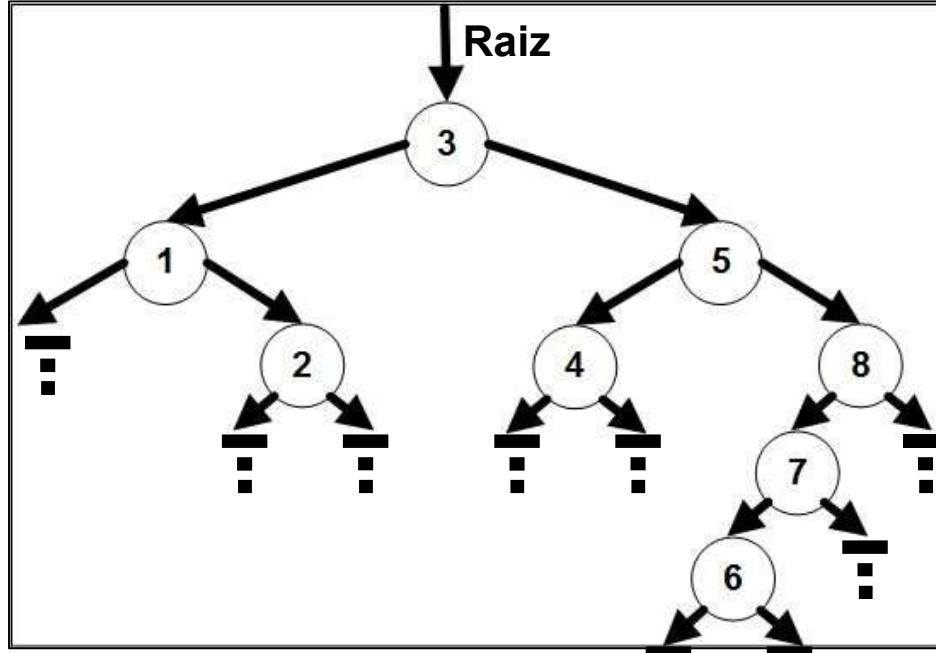
```
//remover(2), folha
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2



Algoritmo de Remoção em Java

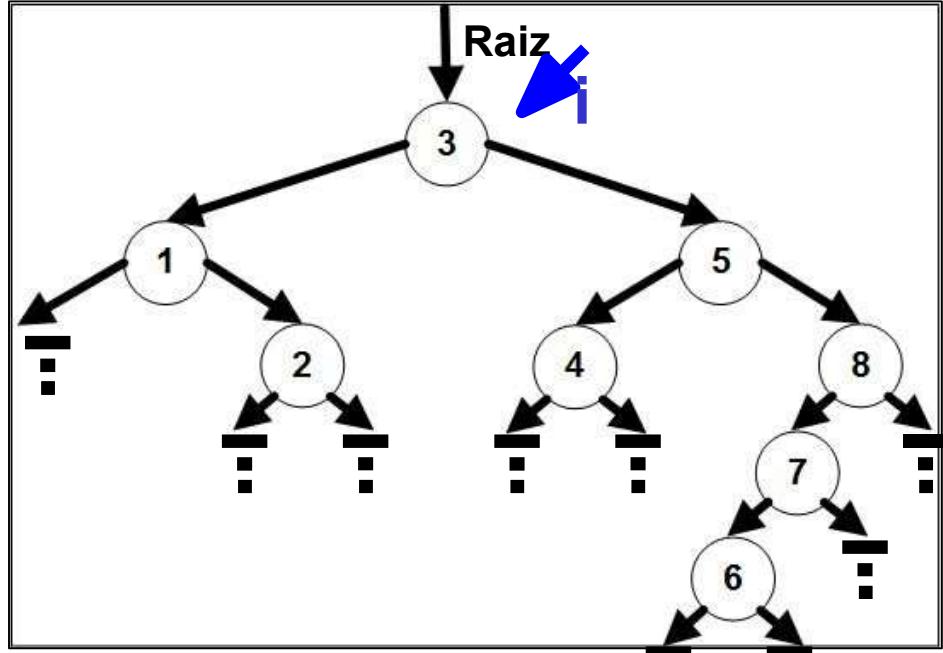
```
//remover(2), folha
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2 x 2 i n(3)



Algoritmo de Remoção em Java

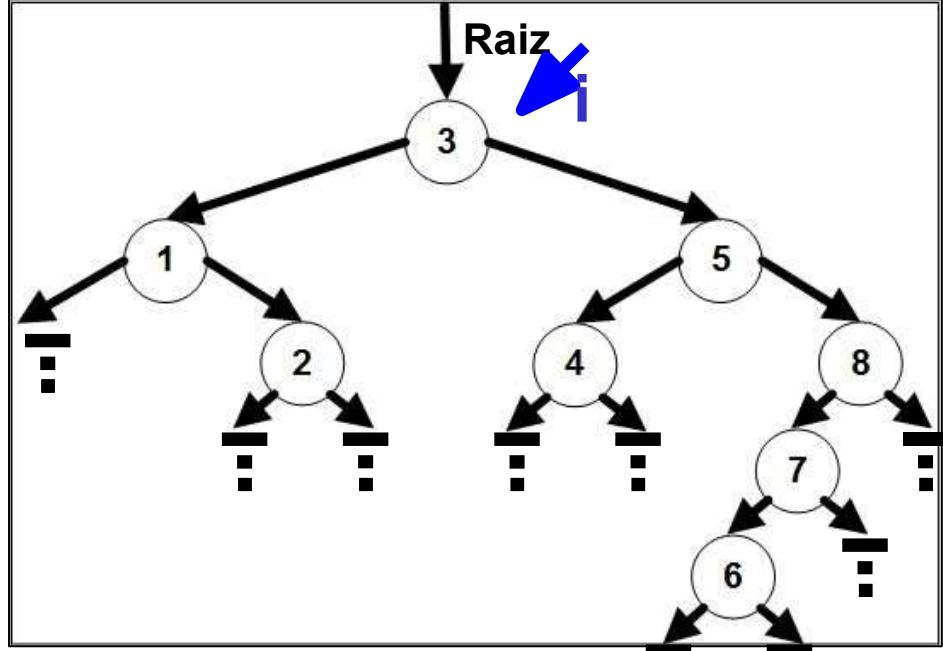
```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
false: n(3) == null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2 x 2 i n(3)



Algoritmo de Remoção em Java

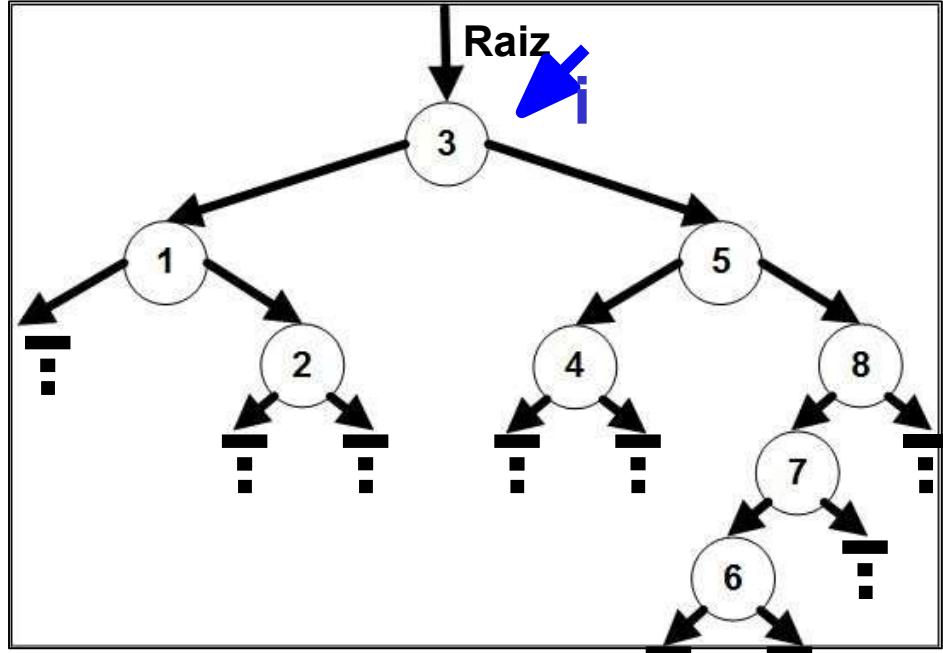
```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
true: 2 < 3

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2 x 2 i n(3)



Algoritmo de Remoção em Java

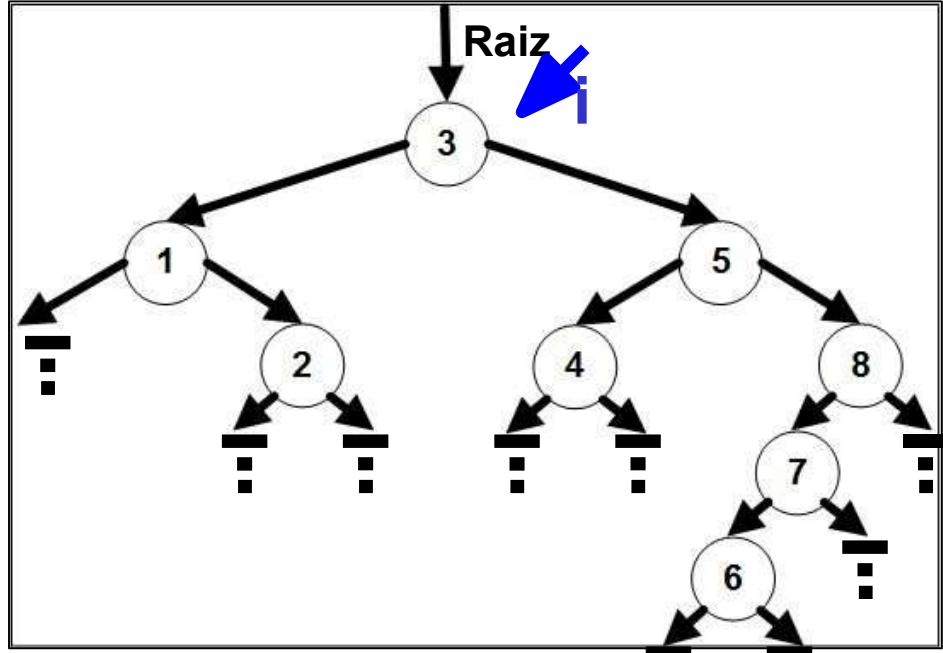
```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq=remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2 x 2 i n(3)



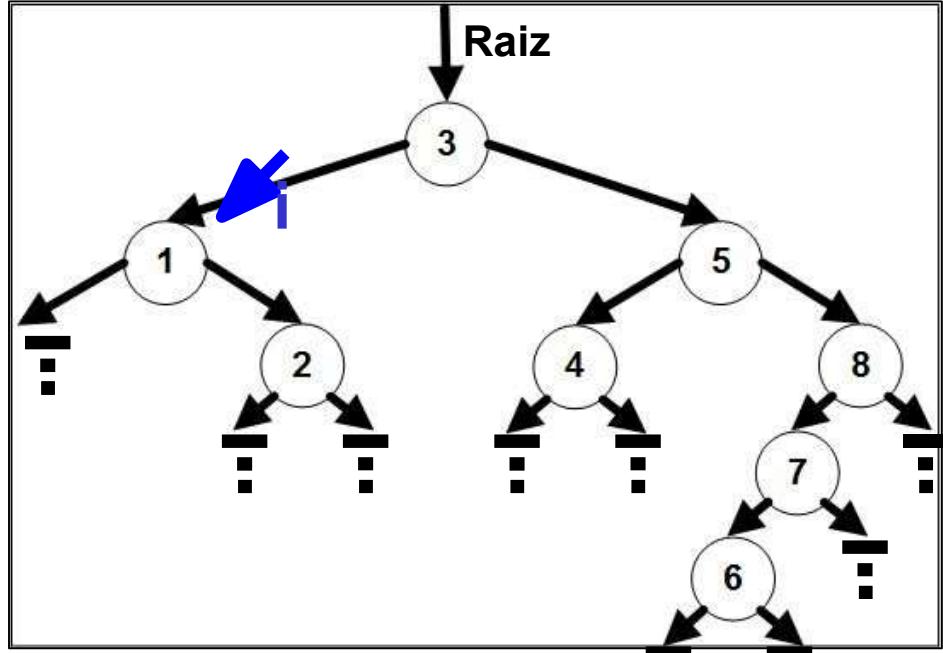
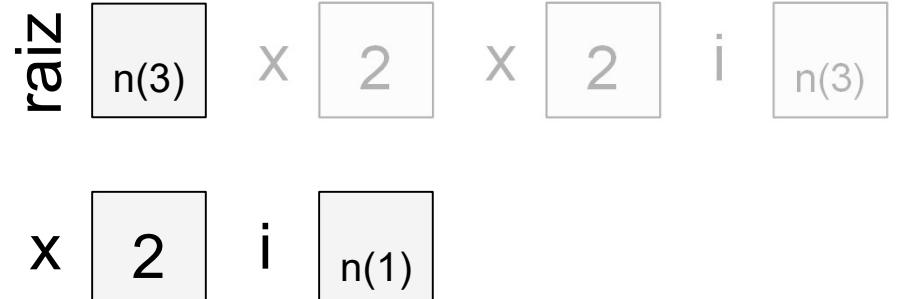
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



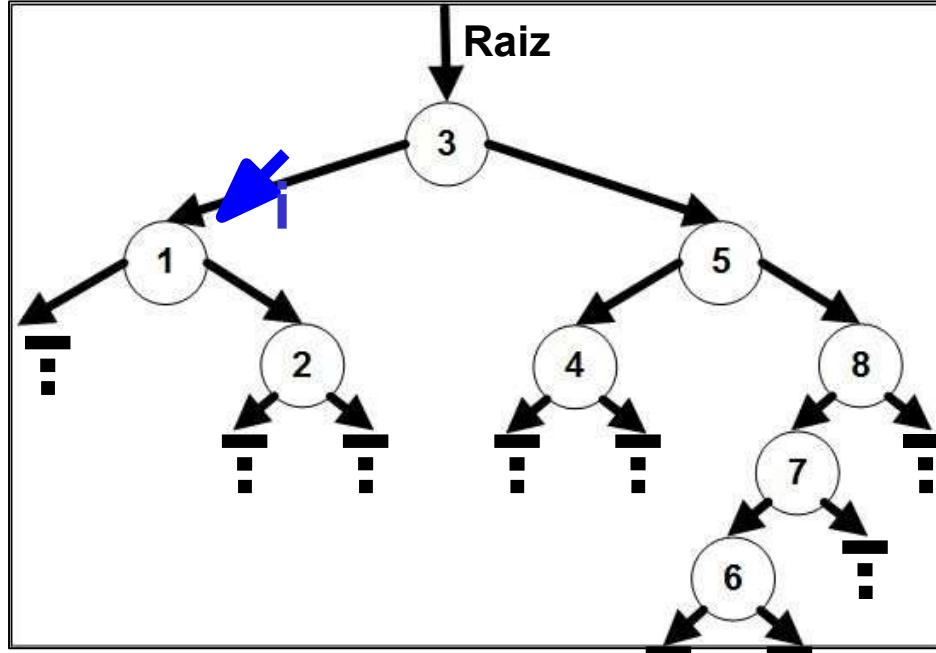
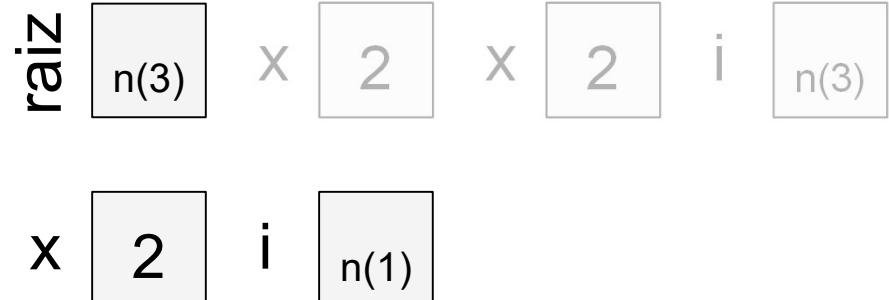
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
false: n(1) == null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



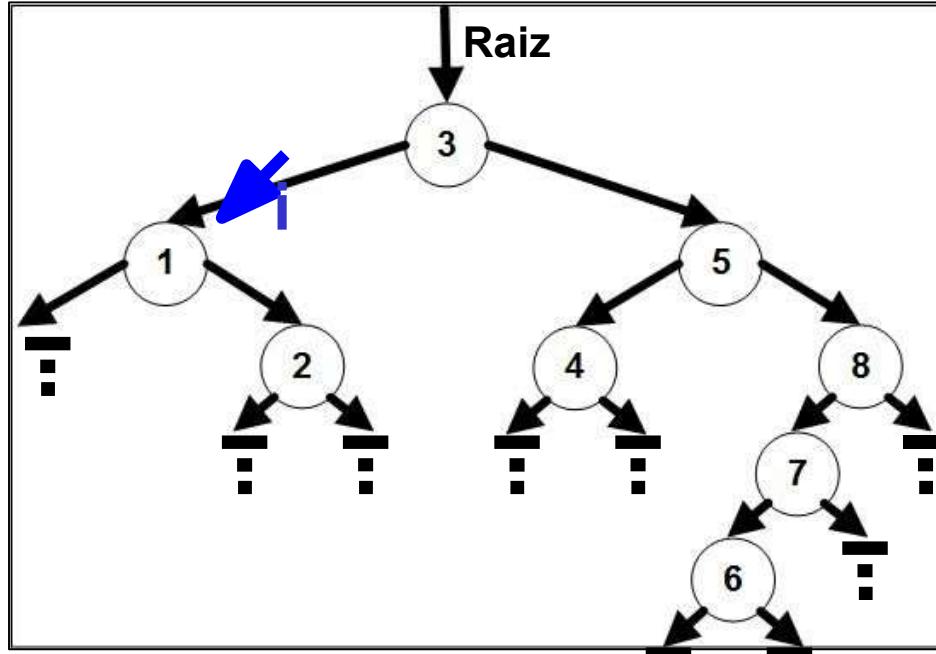
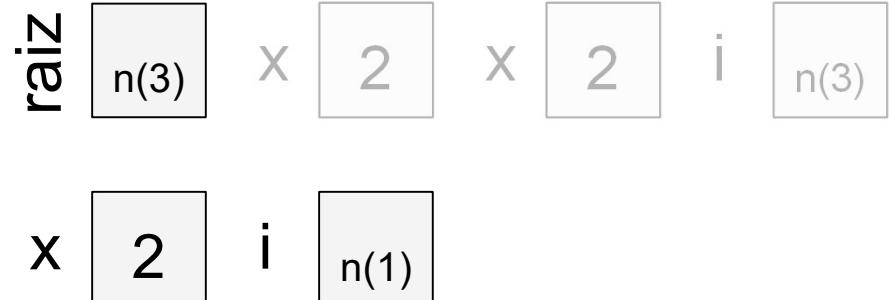
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
false: 2 < 1

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



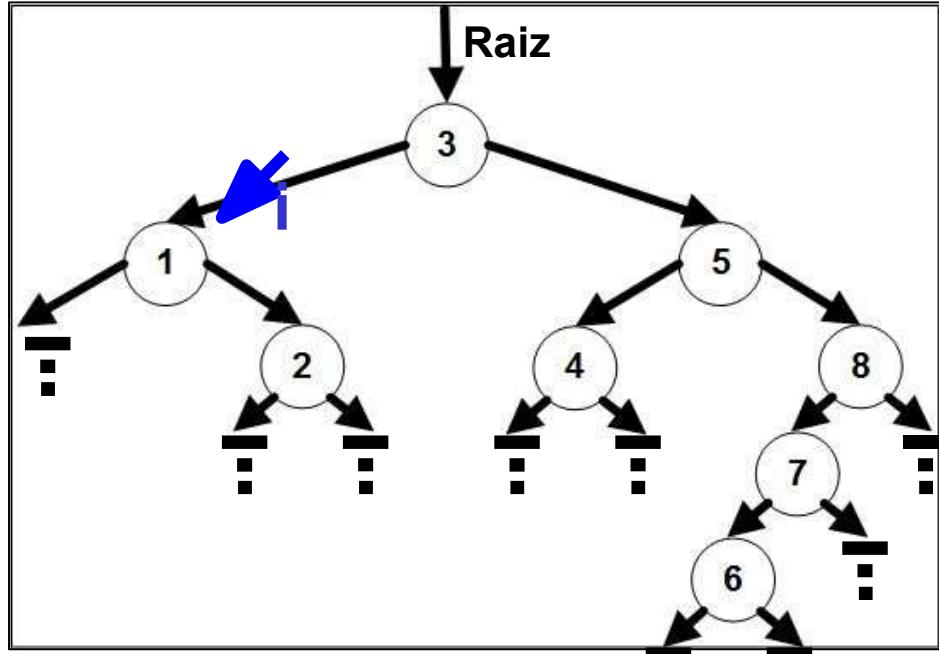
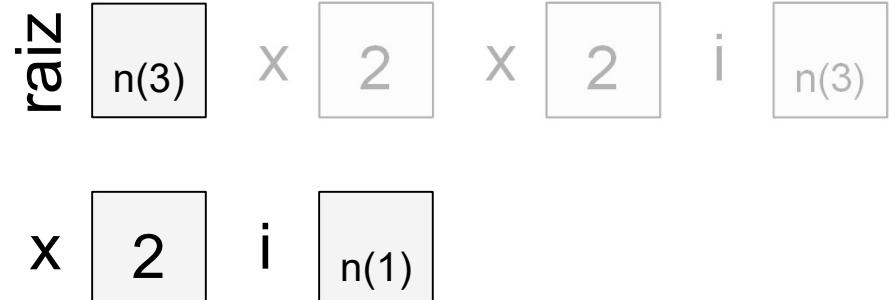
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento){ i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
true: 2 > 1

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



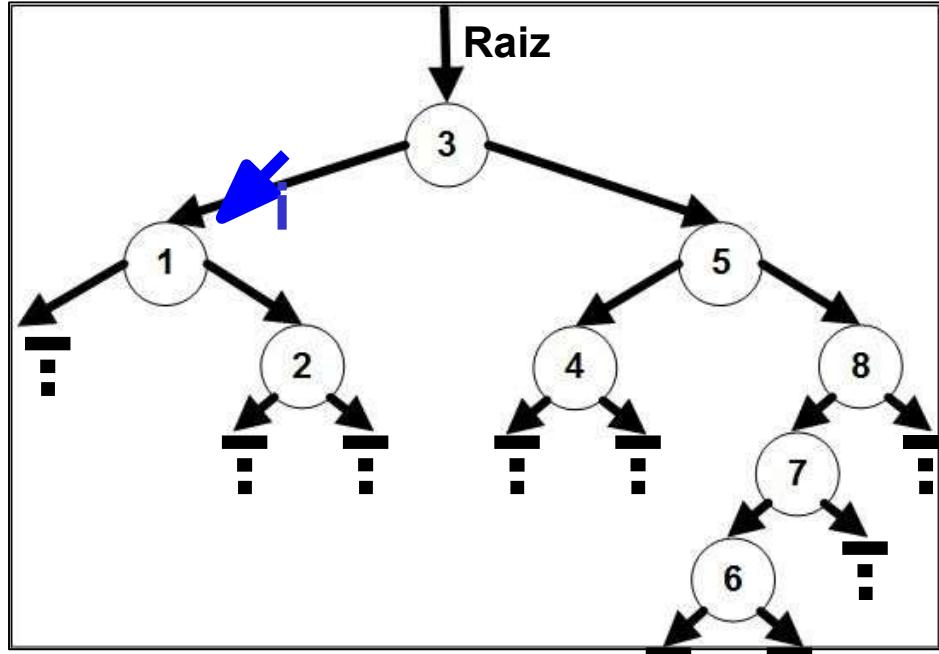
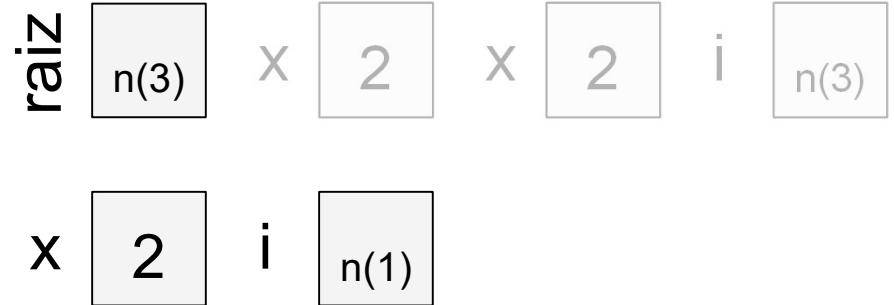
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



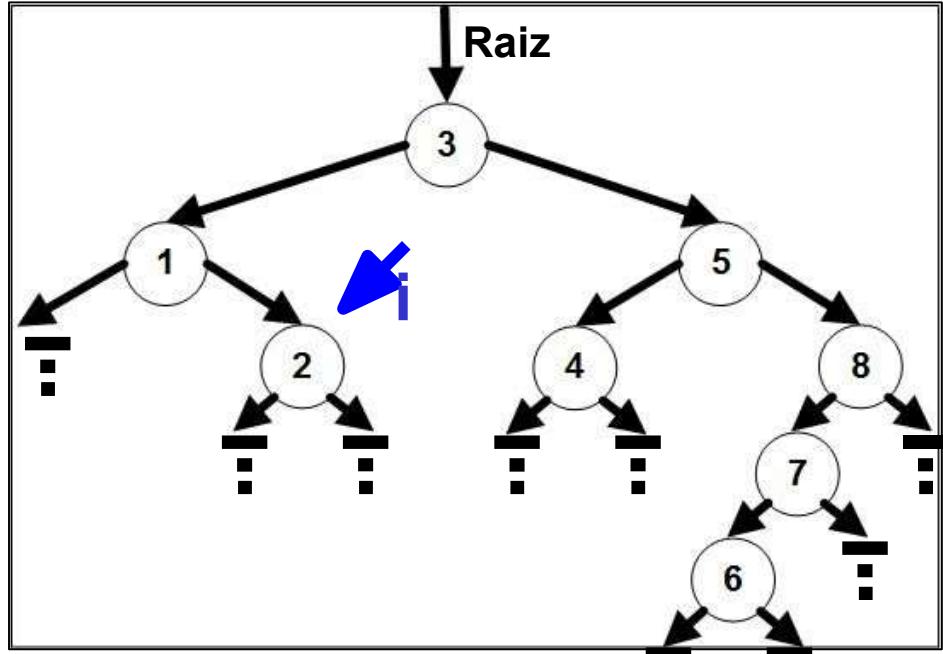
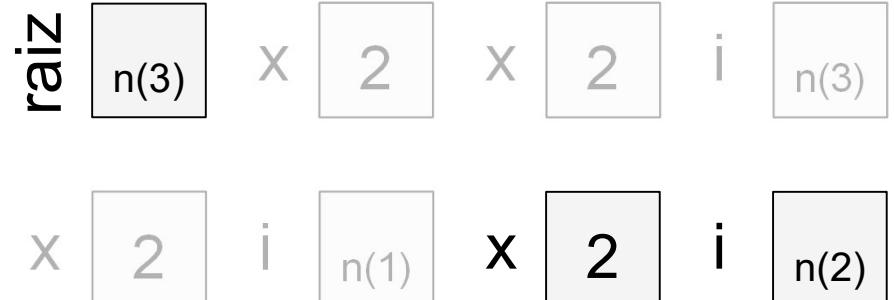
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



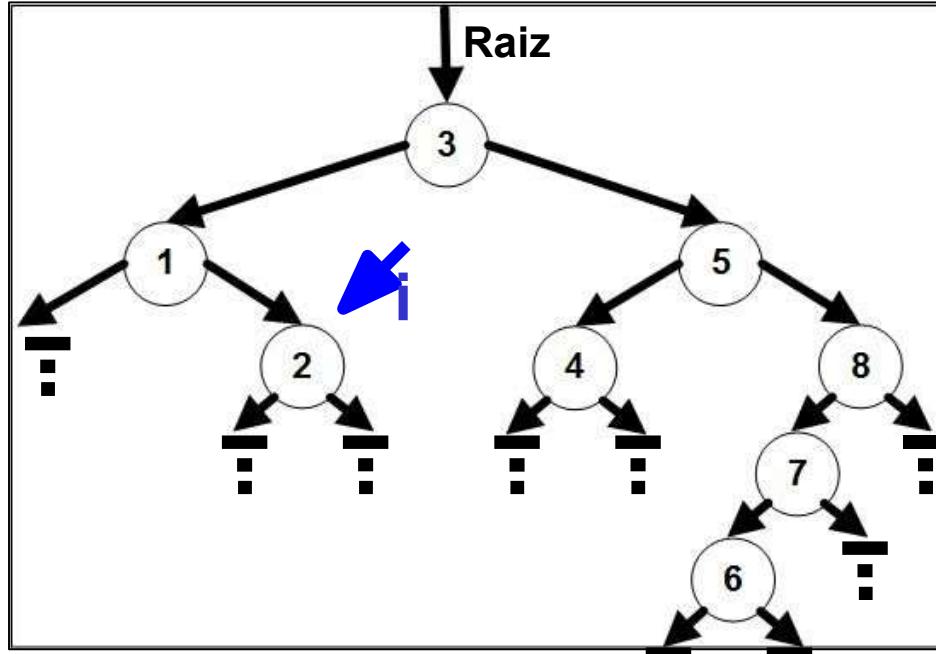
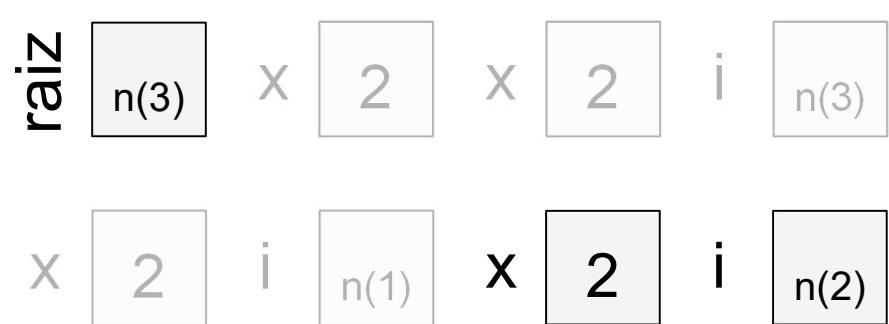
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
false: n(2) == null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



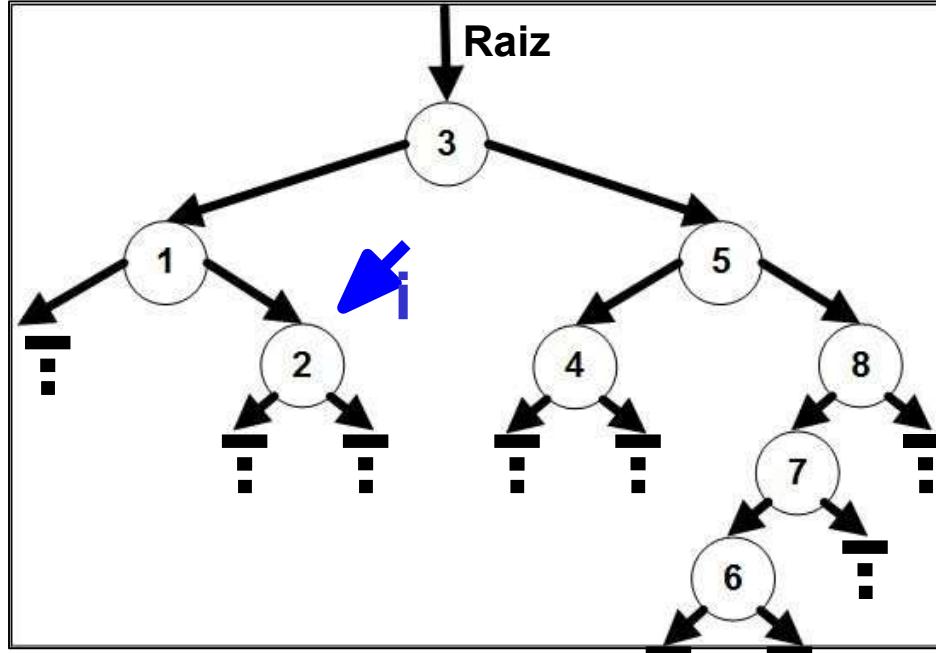
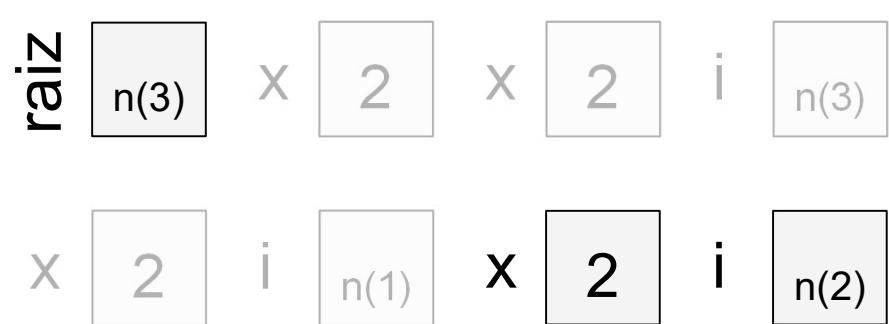
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
false: 2 < 2

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



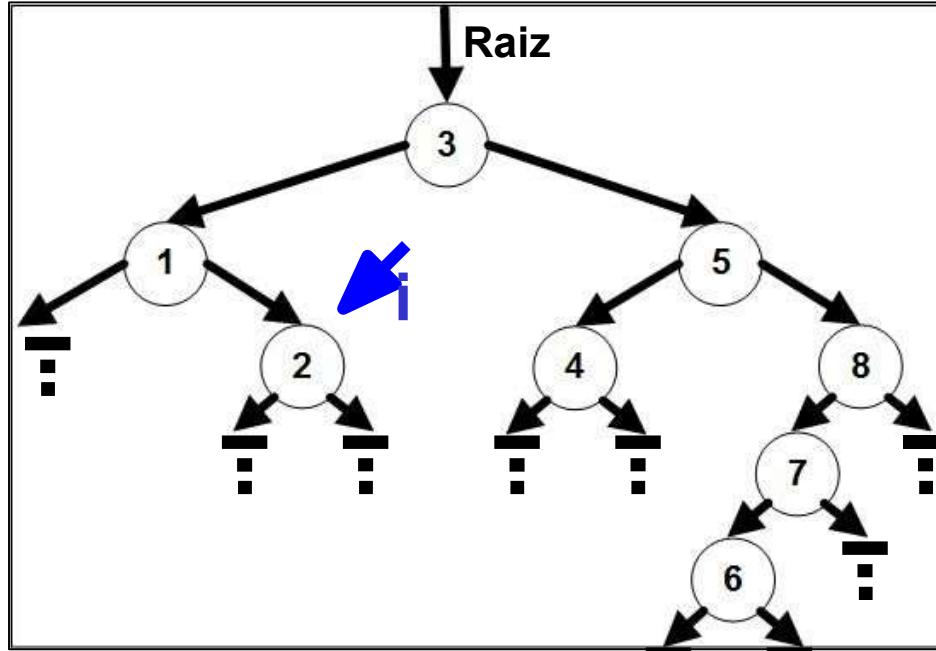
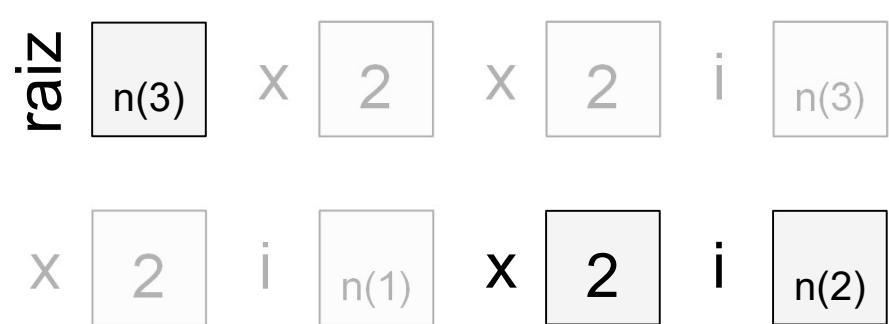
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento){ i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
false: 2 > 2

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



Algoritmo de Remoção em Java

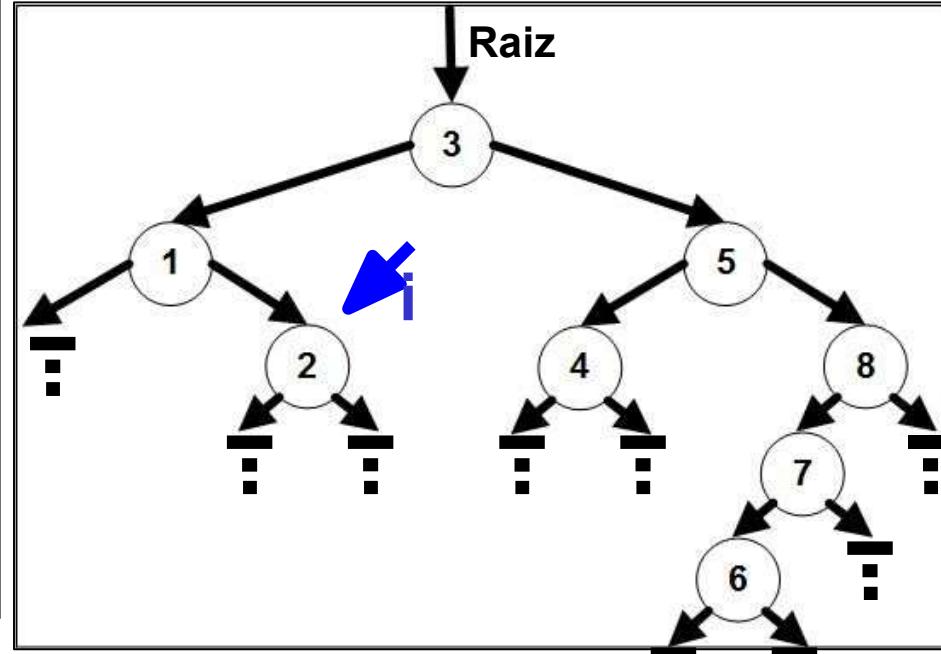
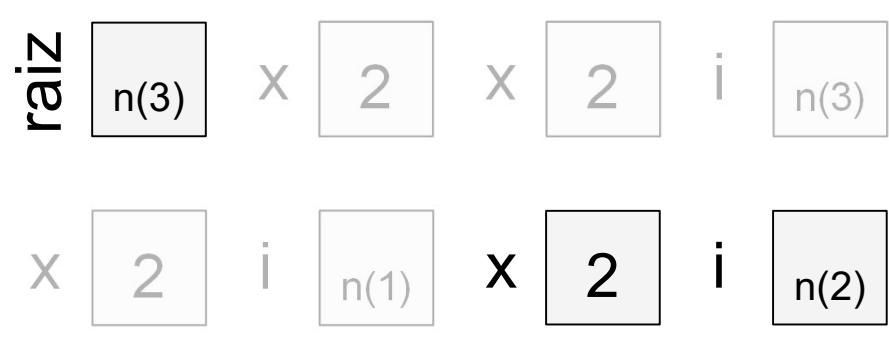
```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

Ops!! x == i.elemento

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



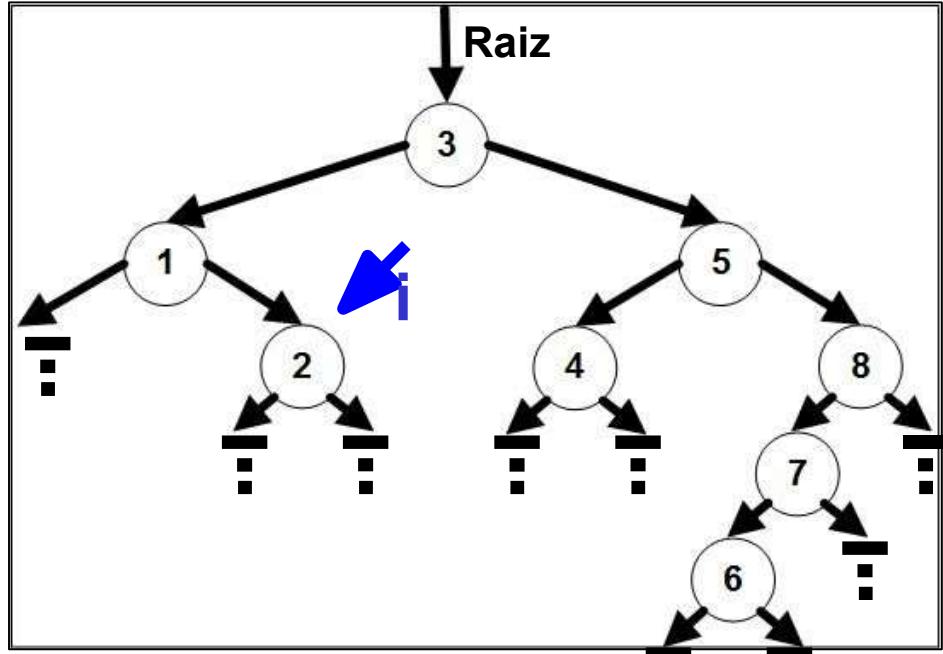
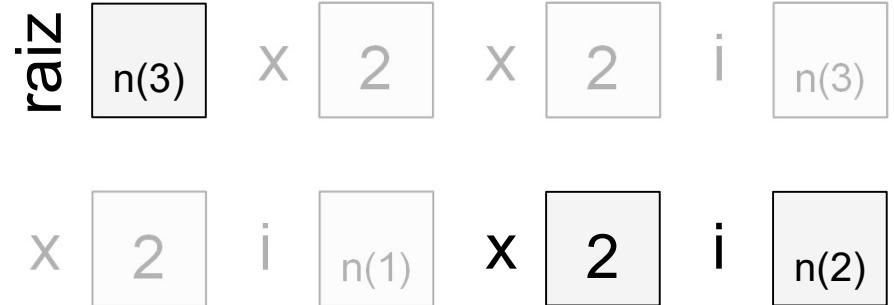
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
} else if(i.dir == null) { i = i.esq;
} else if(i.esq == null) { i = i.dir;
} else {
    i.esq = maiorEsq(i, i.esq); }
return i;
}
true: null == null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else { j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



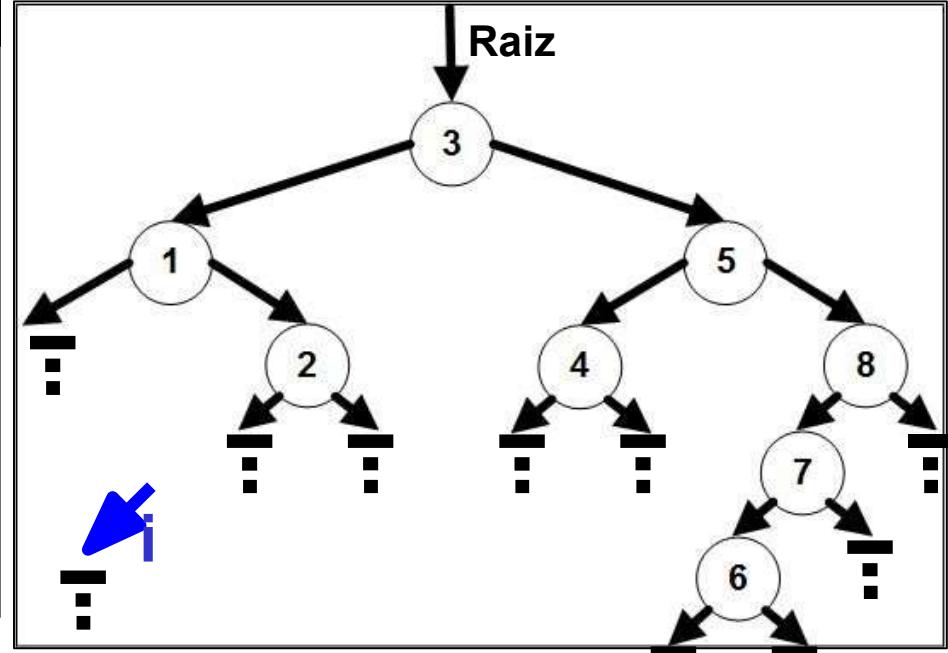
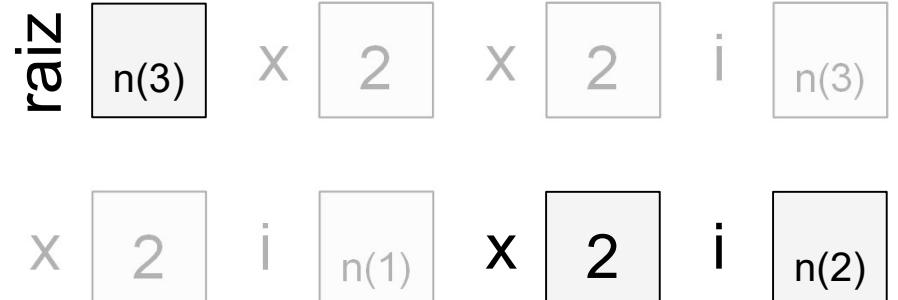
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



Algoritmo de Remoção em Java

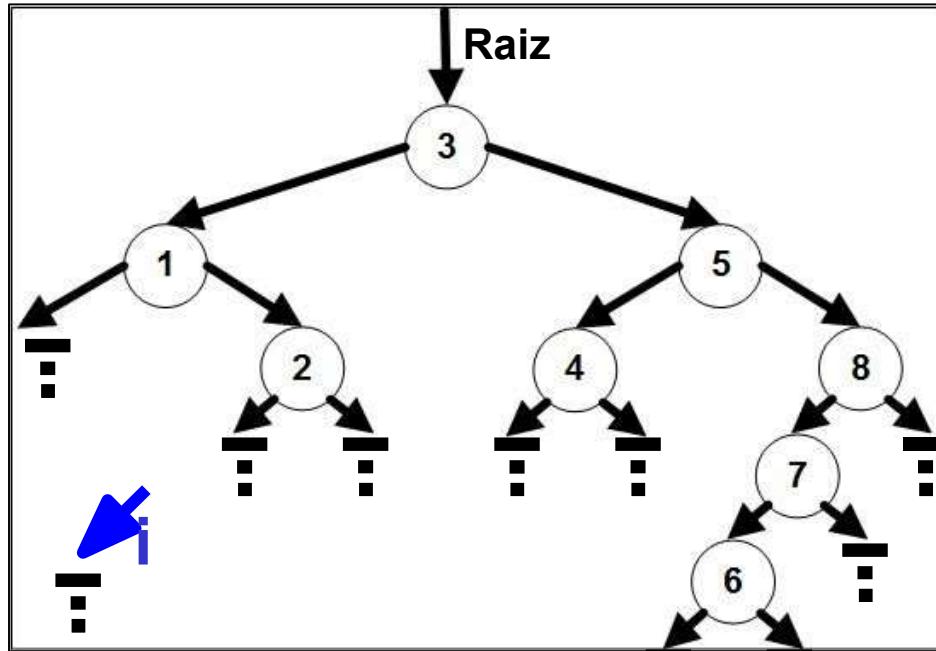
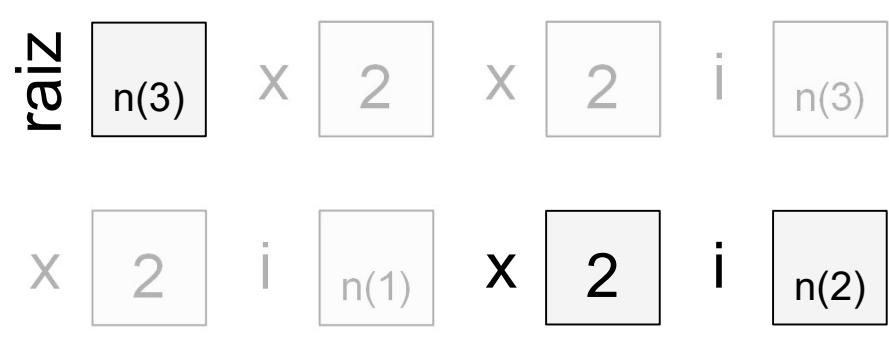
```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

Retorna null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



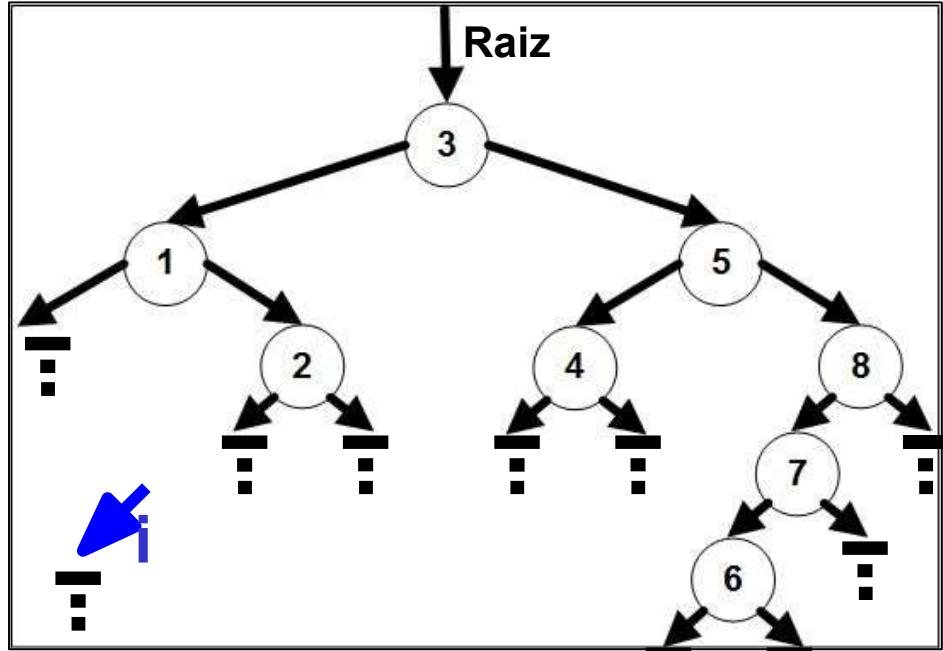
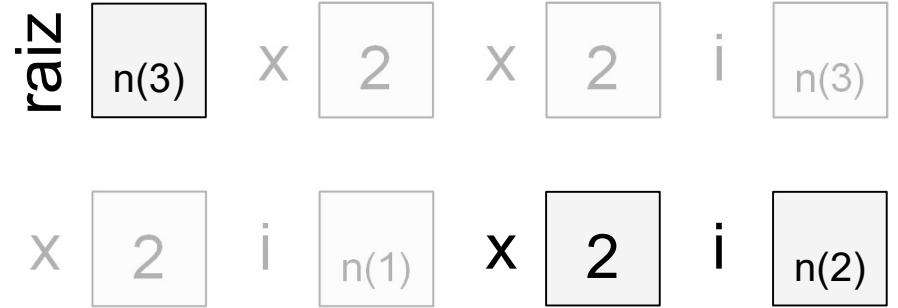
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



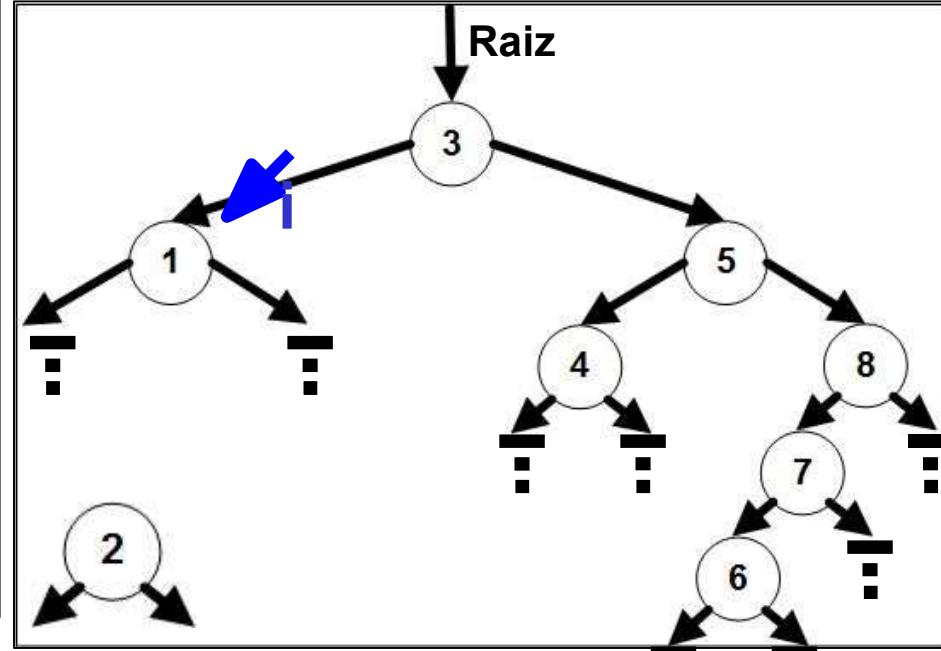
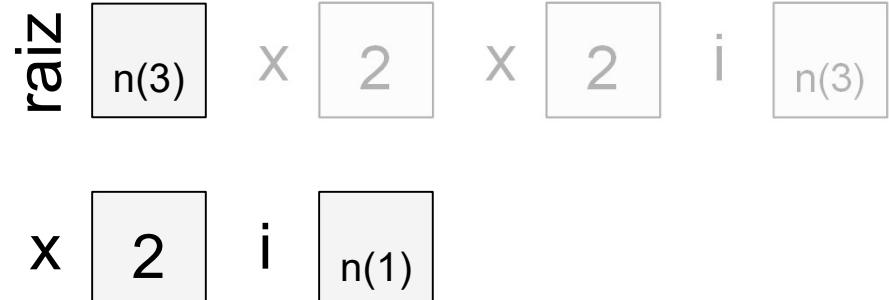
Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



Algoritmo de Remoção em Java

```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

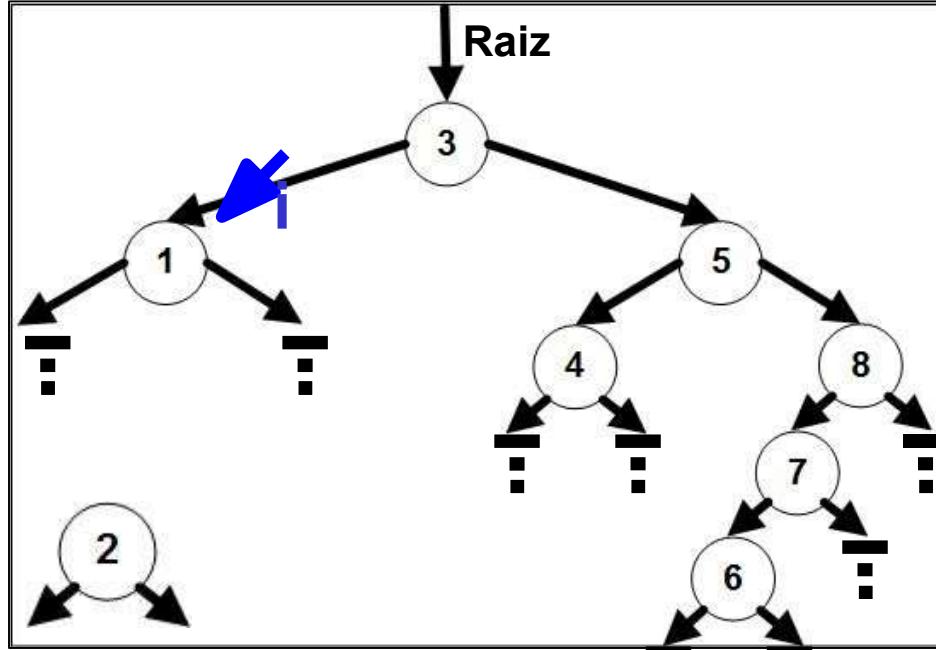
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

Retorna n(1)

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2 x 2 i n(3)

x 2 i n(1)



Algoritmo de Remoção em Java

```
//remover(2), folha
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

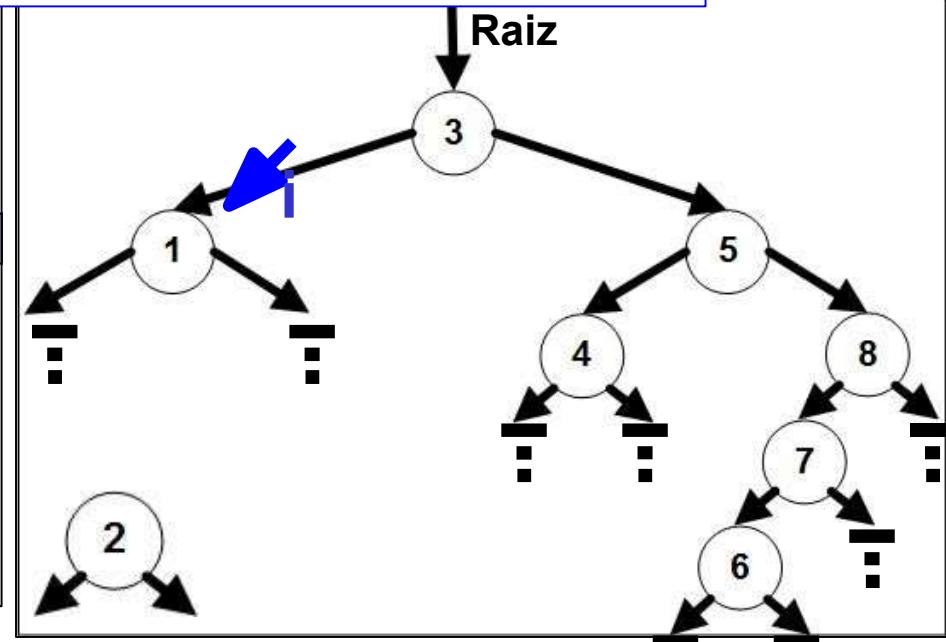
No remo

```
    if (i == null) {
    } else if (i.dir == null) {
    } else if (i.esq == null) {
    } else {
        i.esq = maiorEsq(i, i.esq);
        return i;
    }
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else { j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2 x 2 i n(3)

Após a coleta de lixo do Java
(que não controlamos quando ela acontece)...



Algoritmo de Remoção em Java

```
//remover(2), folha

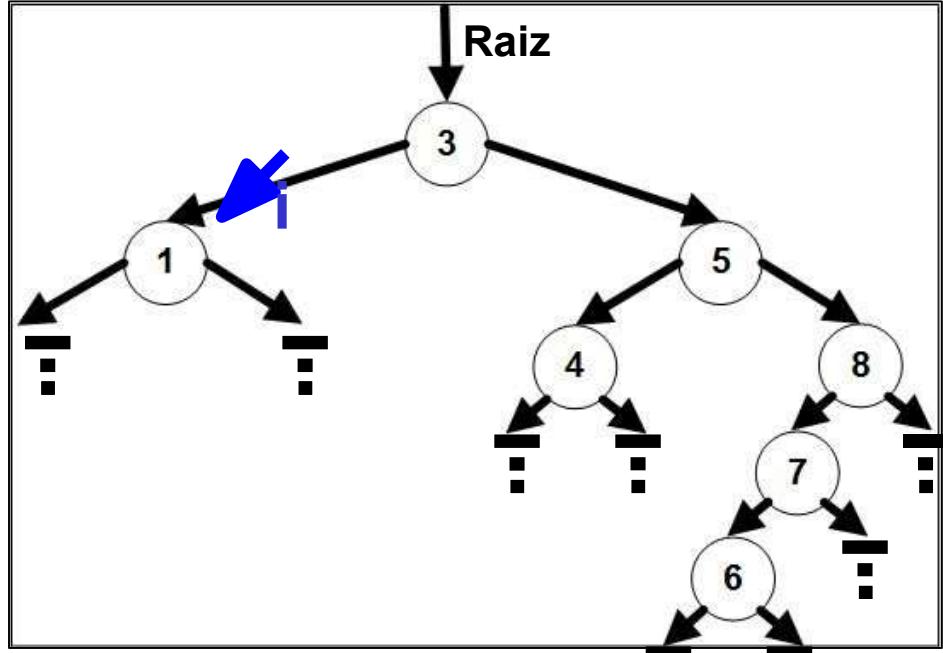
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2 x 2 i n(3)

x 2 i n(1)



Algoritmo de Remoção em Java

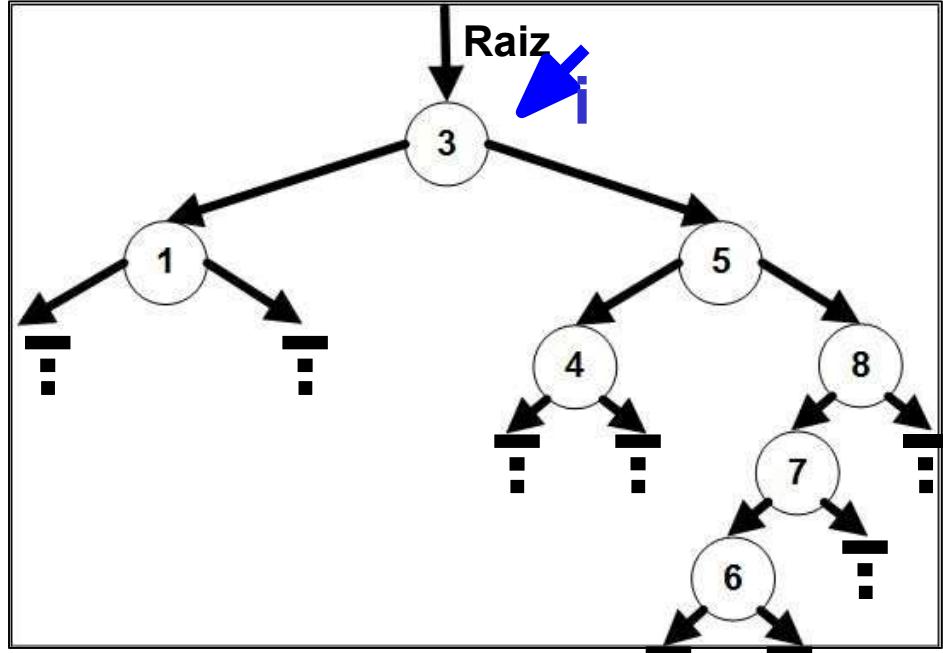
```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq=remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2 x 2 i n(3)



Algoritmo de Remoção em Java

```
//remover(2), folha

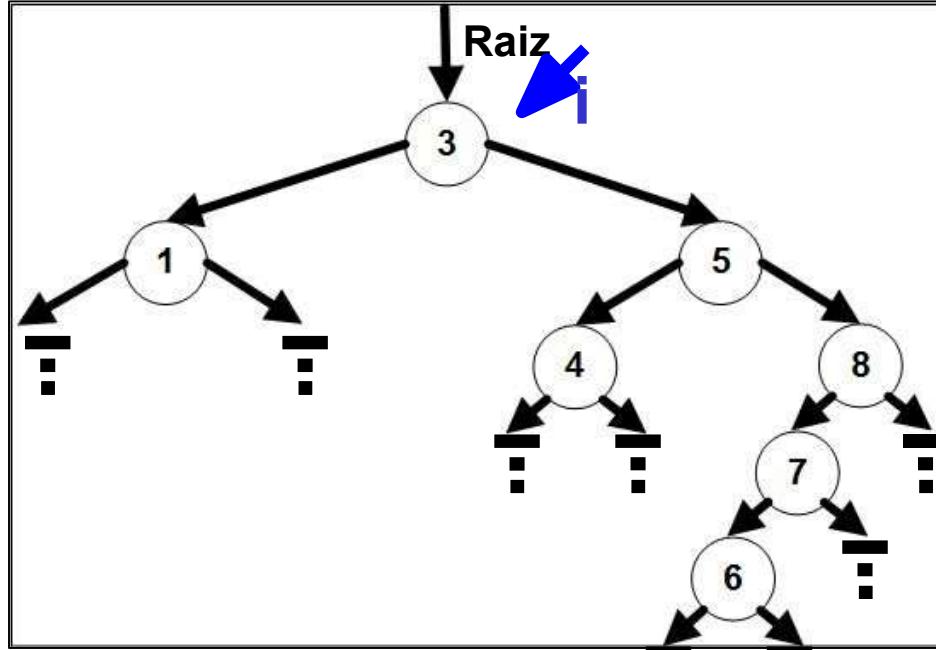
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

Retorna n(3)

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2 x 2 i n(3)



Algoritmo de Remoção em Java

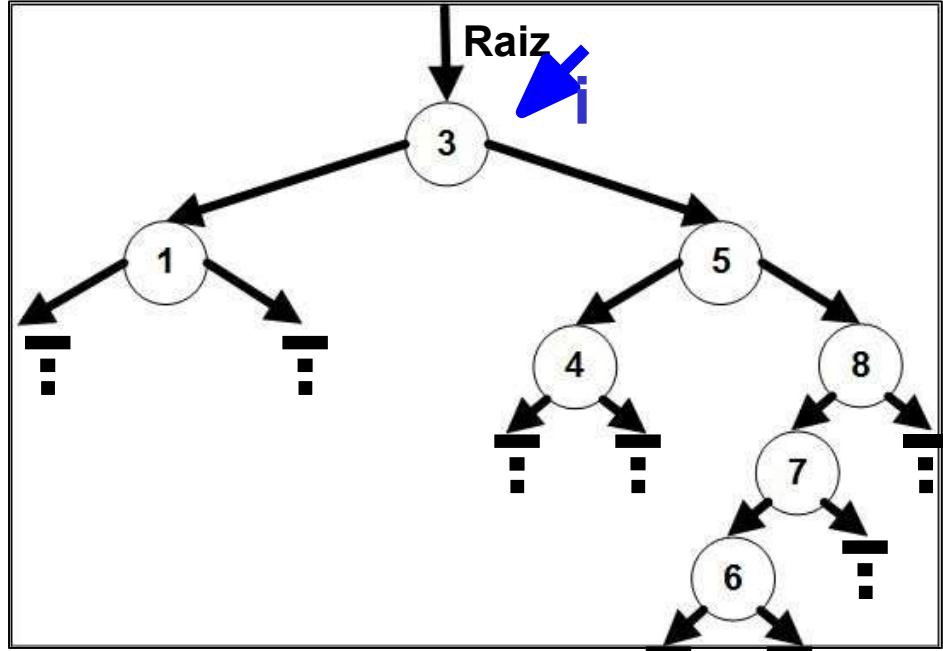
```
//remover(2), folha

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2 x 2 i n(3)



Algoritmo de Remoção em Java

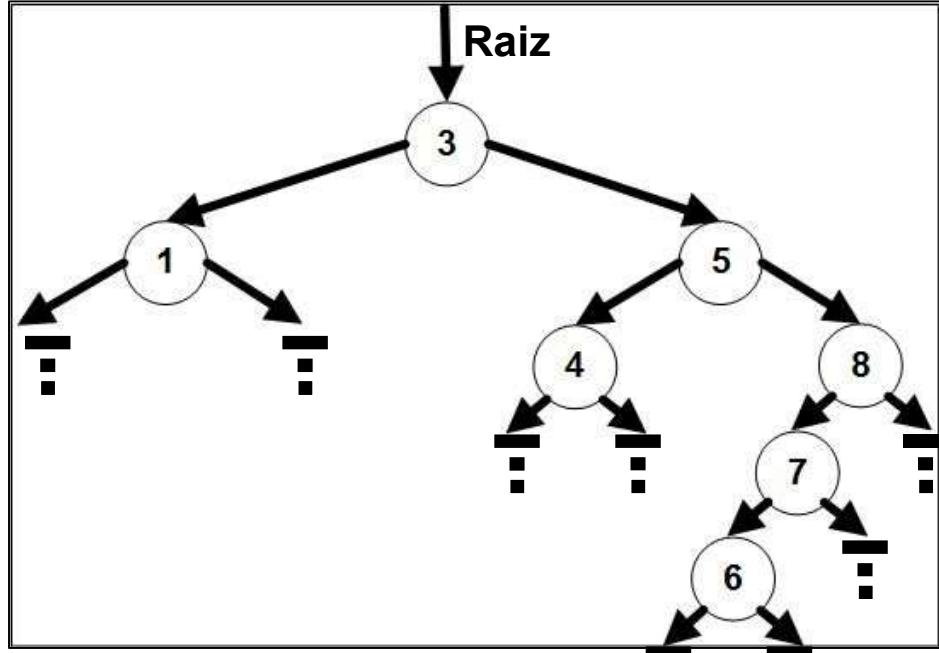
```
//remover(2), folha
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 2



Algoritmo de Remoção em Java

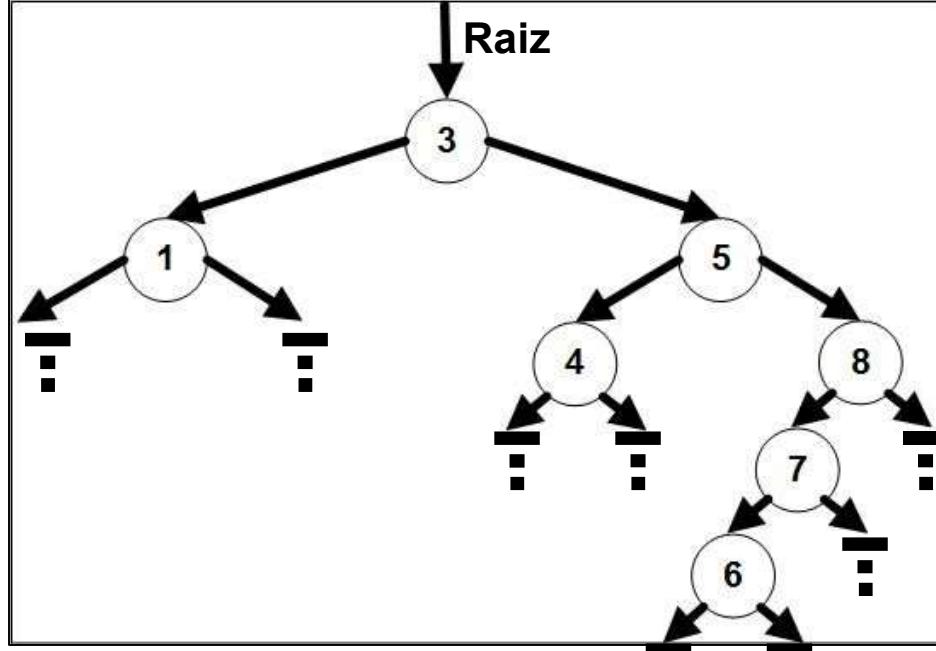
```
//remover(2), folha
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz
n(3)

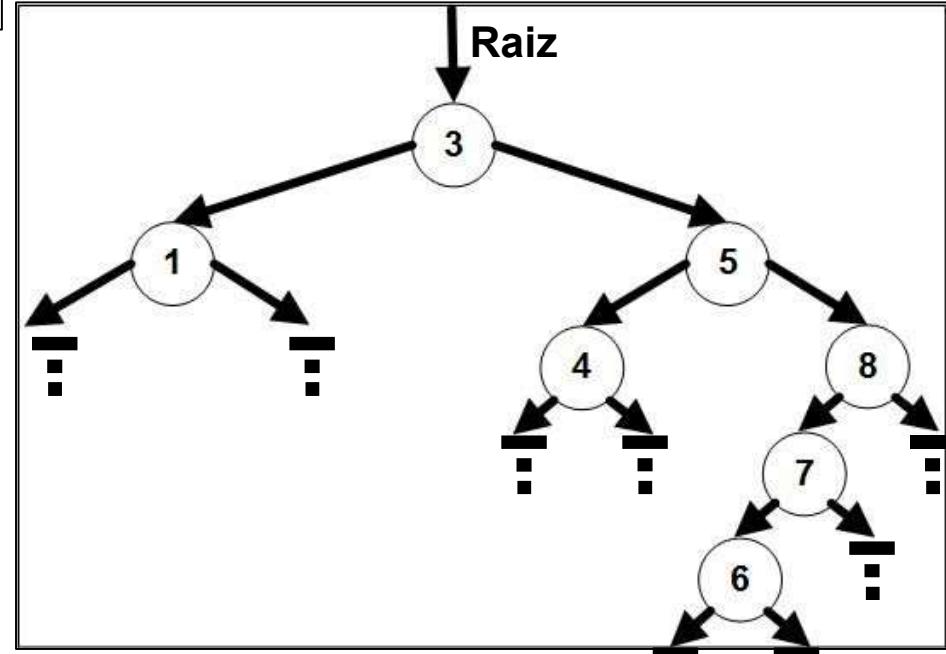


Algoritmo de Remoção em Java

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

raiz
n(3)

Voltando com o 2 antes
de fazer outra remoção

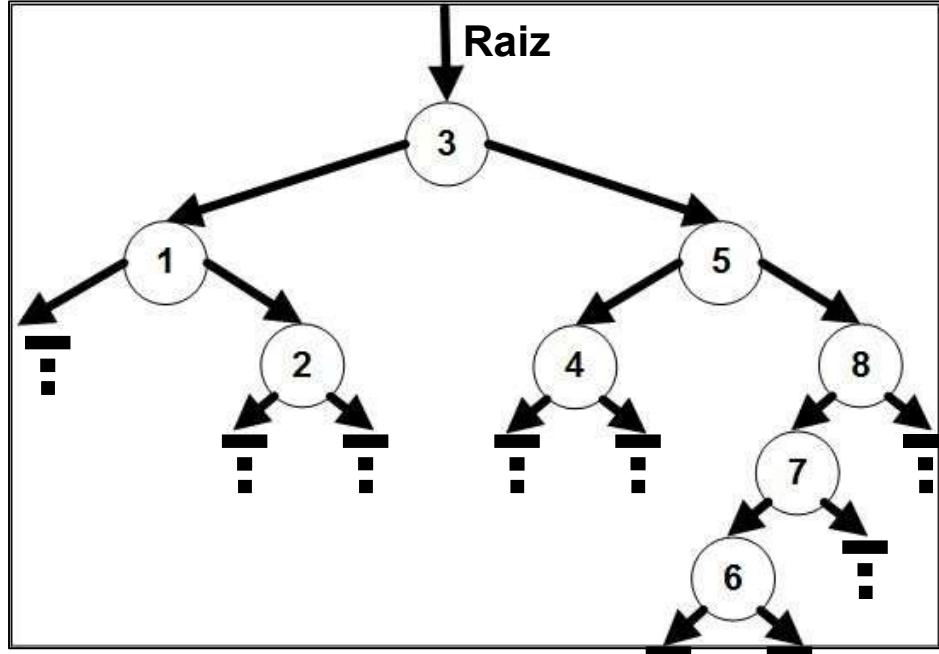


Algoritmo de Remoção em Java

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

raiz
n(3)

Vamos remover o 1 (tem um filho) de nossa árvore



Algoritmo de Remoção em Java

```
//remover(1), um filho
```

```
void remover(int x) throws Exception {
```

```
    raiz = remover(x, raiz);
}
```

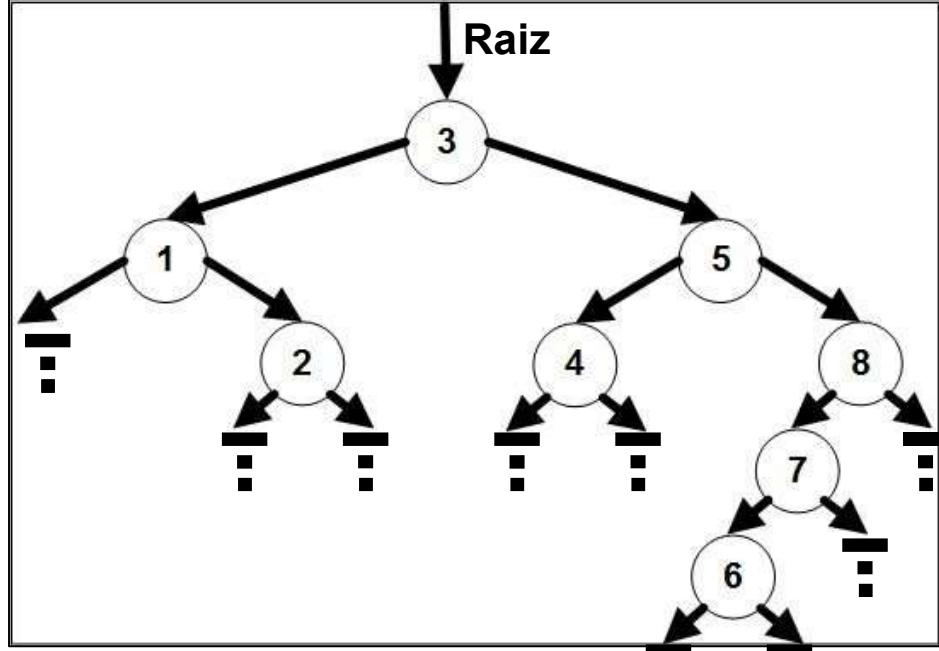
```
No remover(int x, No i) throws Exception {
```

```
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq);
    } else if(x > i.elemento) { i.dir = remover(x, i.dir);
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
```

```
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir);
    }
    return j;
}
```

raiz n(3) X 1



Algoritmo de Remoção em Java

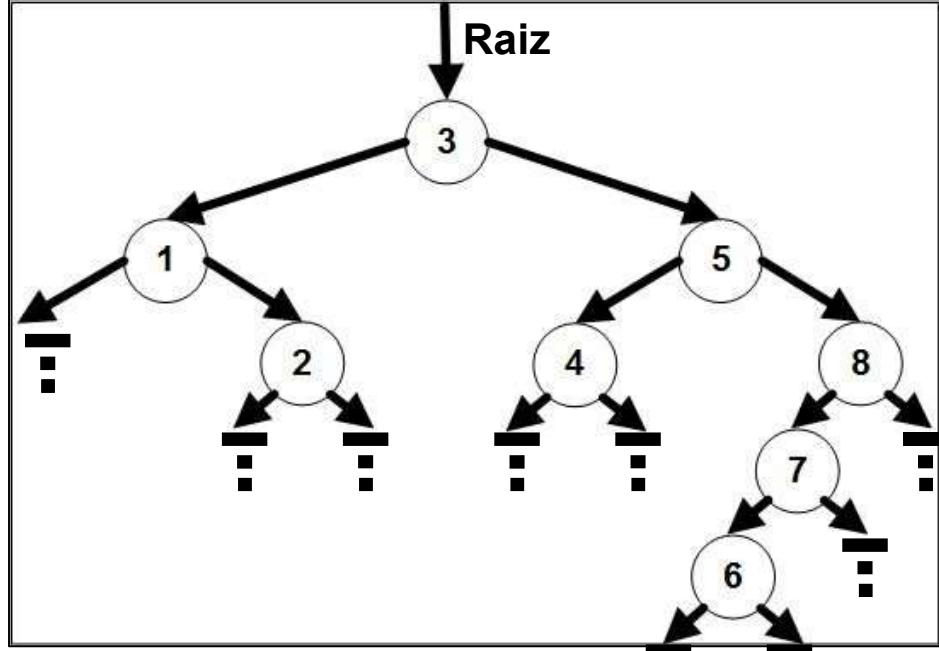
```
//remover(1), um filho
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 1



Algoritmo de Remoção em Java

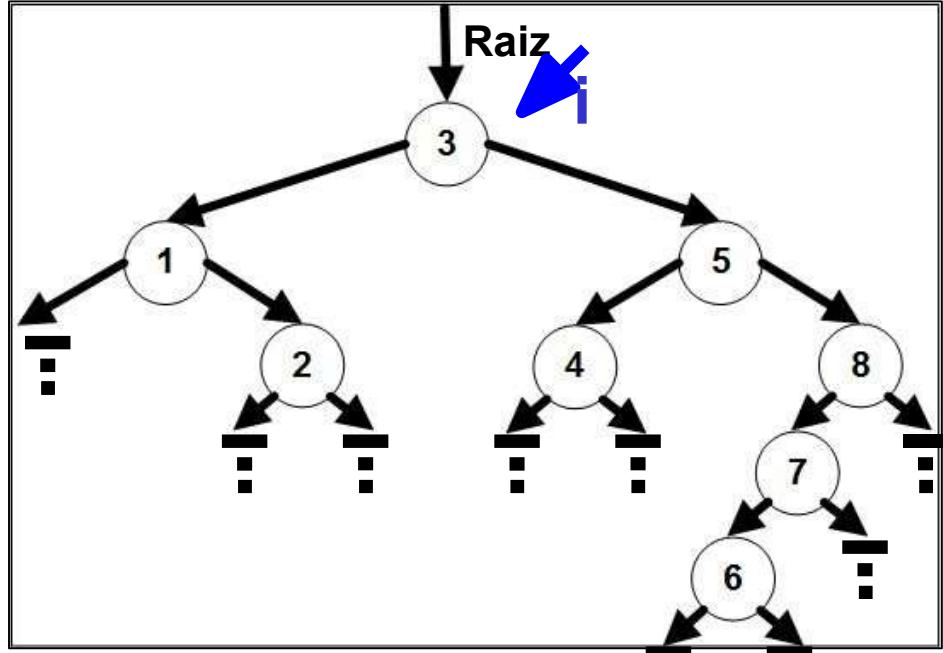
```
//remover(1), um filho
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 1 x 1 i n(3)



Algoritmo de Remoção em Java

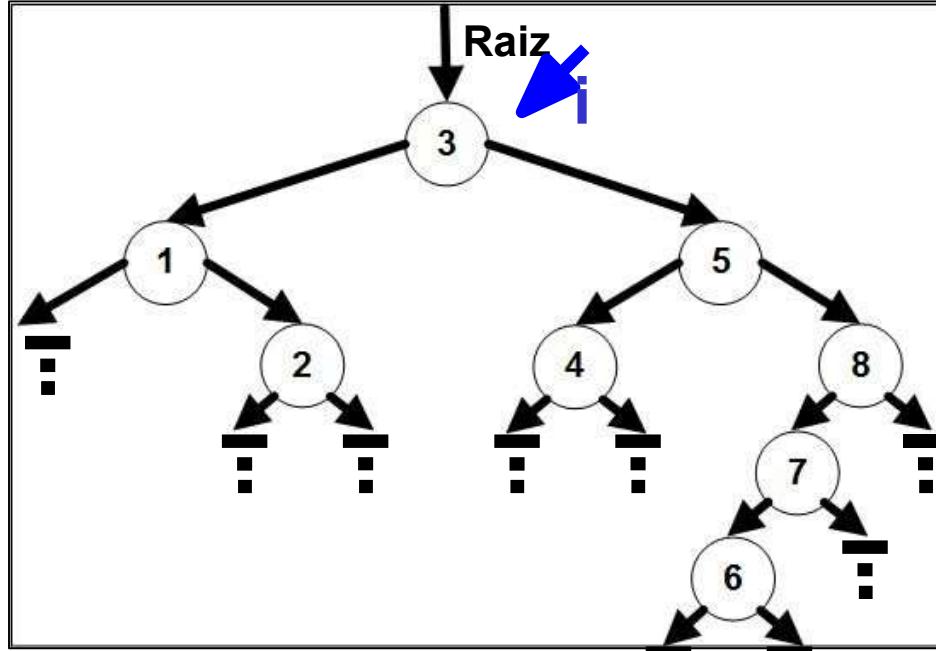
```
//remover(1), um filho

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!");
    } else if(x < i.elemento){ i.esq = remover(x, i.esq);
    } else if(x > i.elemento) { i.dir = remover(x, i.dir);
    } else if(i.dir == null) {   i = i.esq;
    } else if(i.esq == null) {   i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
} false: n(3) == null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 1 x 1 i n(3)



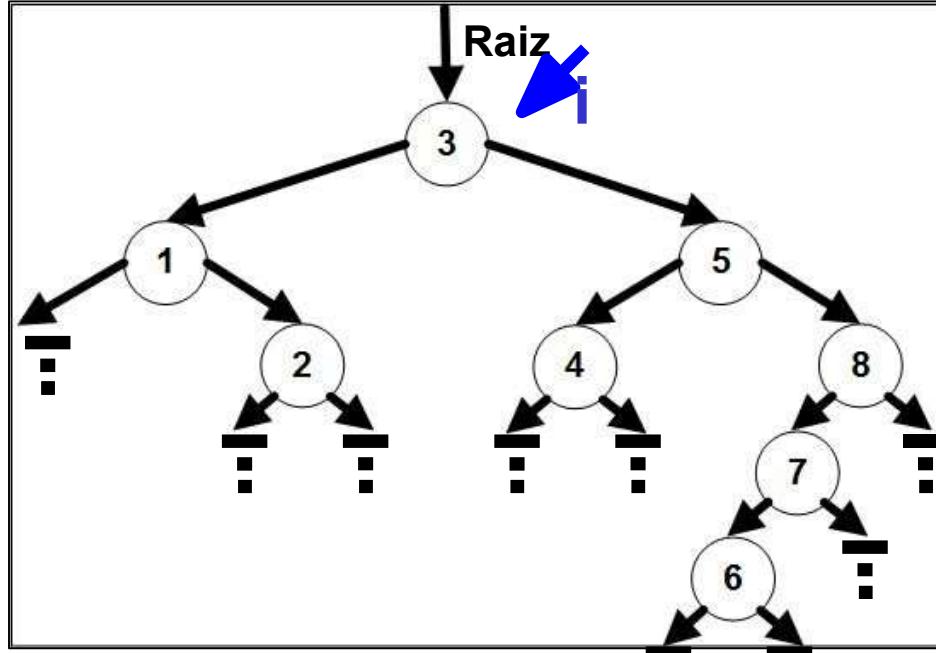
Algoritmo de Remoção em Java

```
//remover(1), um filho

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) {      throw new Exception("Erro!");
    } else if(x < i.elemento){ i.esq = remover(x, i.esq);
    } else if(x > i.elemento) { i.dir = remover(x, i.dir);
    } else if(i.dir == null) {   i = i.esq;
    } else if(i.esq == null) {   i = i.dir;
    } else {                  i.esq = maiorEsq(i, i.esq); }
    return i;
}                                true: 1 < 3

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else {                j.dir = maiorEsq(i, j.dir);      }
    return j;
}
```



Algoritmo de Remoção em Java

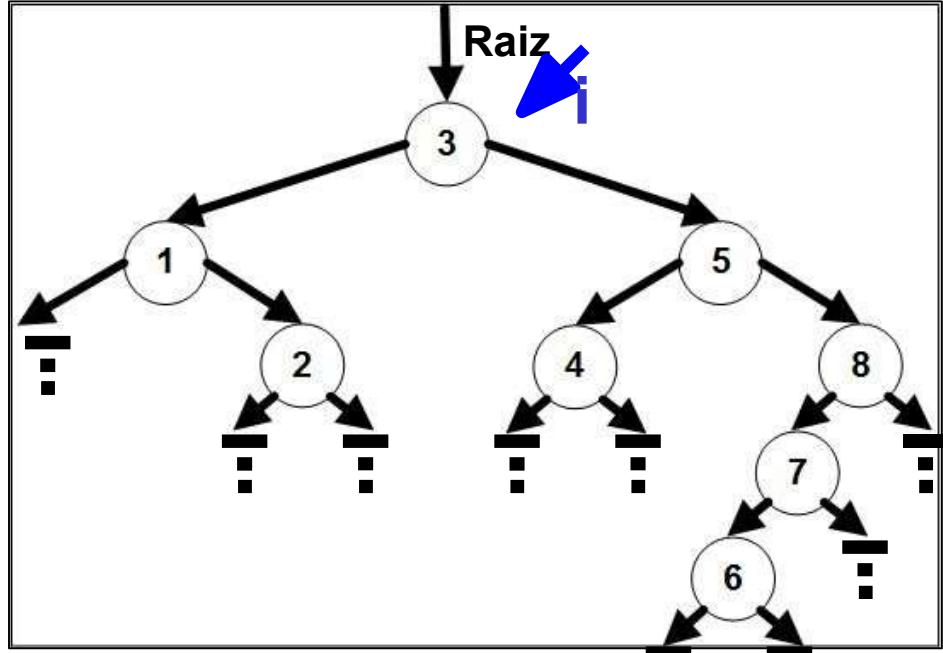
```
//remover(1), um filho

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    else if(x < i.elemento) { i.esq = remover(x, i.esq); }
    else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    else if(i.dir == null) { i = i.esq; }
    else if(i.esq == null) { i = i.dir; }
    else { i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else { j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 1 X 1 i n(3)



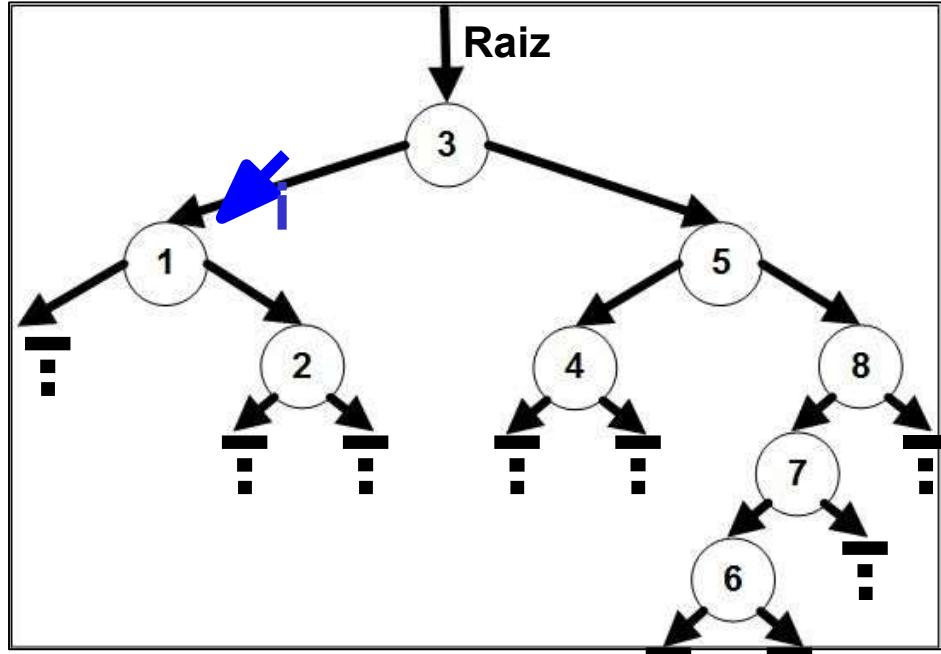
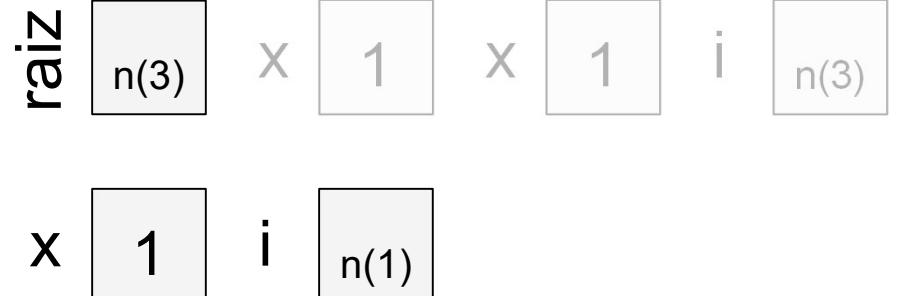
Algoritmo de Remoção em Java

```
//remover(1), um filho

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    else if(i.dir == null) { i = i.esq; }
    else if(i.esq == null) { i = i.dir; }
    else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



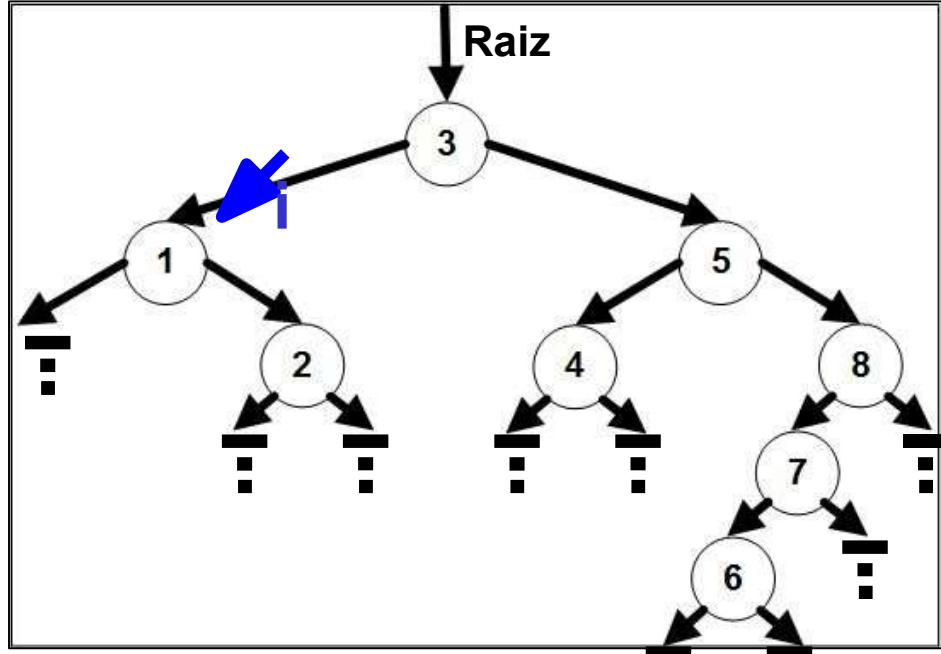
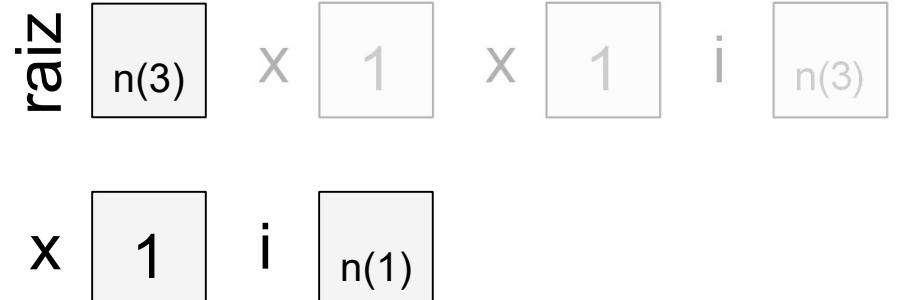
Algoritmo de Remoção em Java

```
//remover(1), um filho

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!");
    } else if(x < i.elemento){ i.esq = remover(x, i.esq);
    } else if(x > i.elemento) { i.dir = remover(x, i.dir);
    } else if(i.dir == null) {   i = i.esq;
    } else if(i.esq == null) {   i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
} false: n(1) == null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



Algoritmo de Remoção em Java

```
//remover(1), um filho

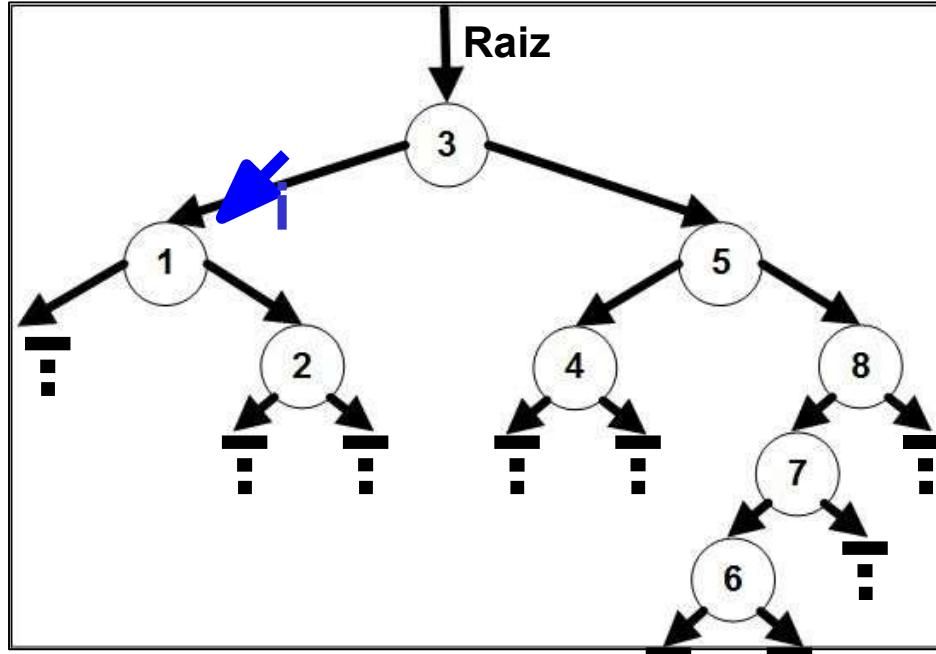
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
false: 1 < 1

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 1 x 1 i n(3)

x 1 i n(1)



Algoritmo de Remoção em Java

```
//remover(1), um filho

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

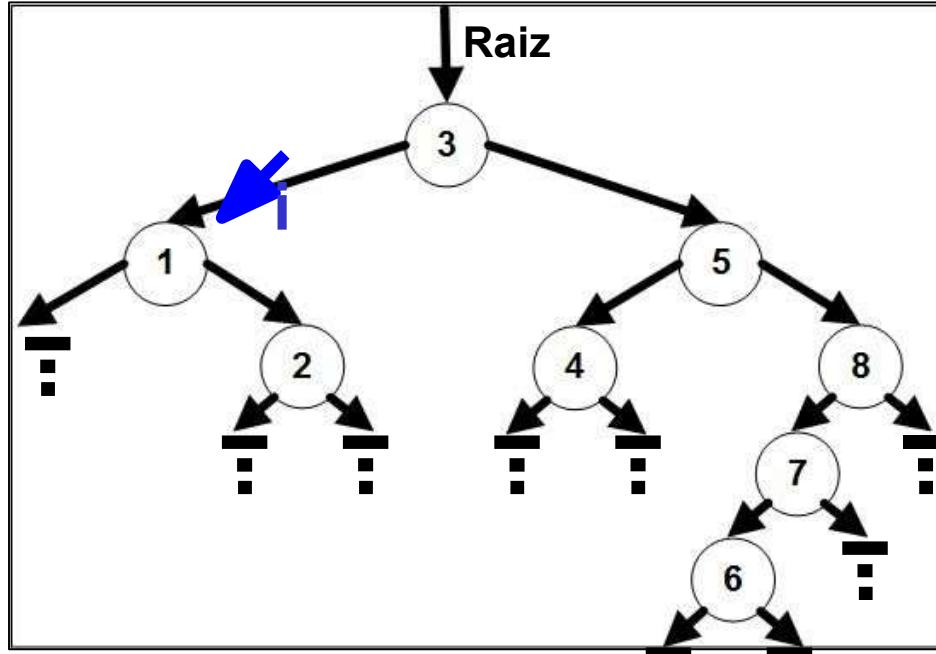
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento){ i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

false: 1 > 1

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 1 X 1 i n(3)

X 1 i n(1)



Algoritmo de Remoção em Java

```
//remover(1), um filho

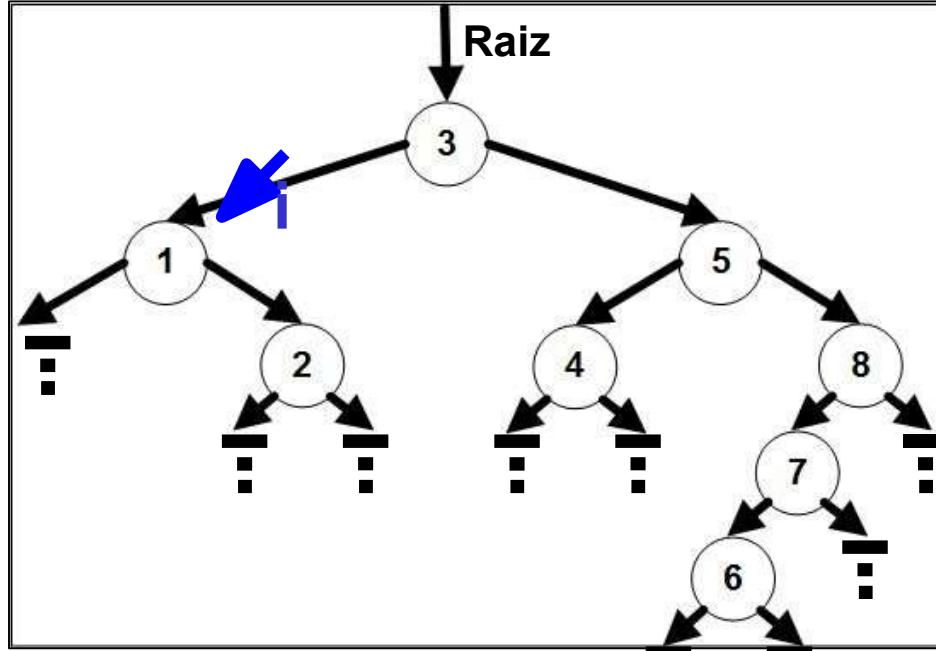
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
} else if(i.dir == null) { i = i.esq;
} else if(i.esq == null) { i = i.dir;
} else {
    i.esq = maiorEsq(i, i.esq); }
return i;
} false: n(2) == null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else { j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 1 X 1 i n(3)

X 1 i n(1)



Algoritmo de Remoção em Java

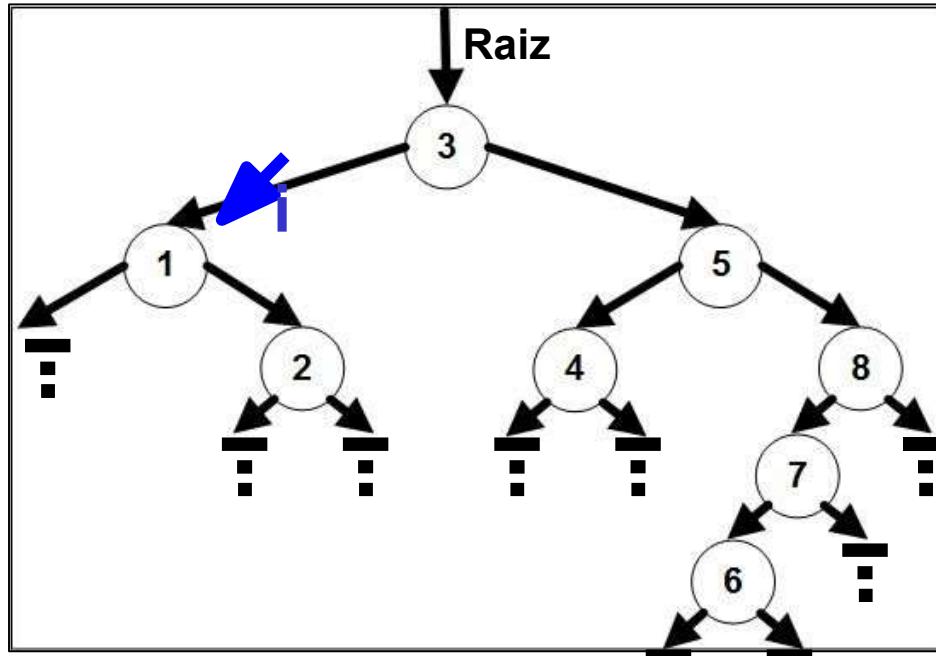
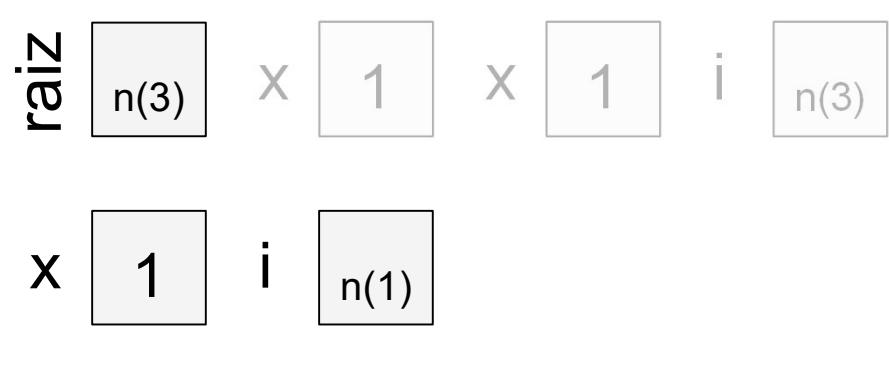
```
//remover(1), um filho

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

true: null == null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else { j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



Algoritmo de Remoção em Java

```
//remover(1), um filho

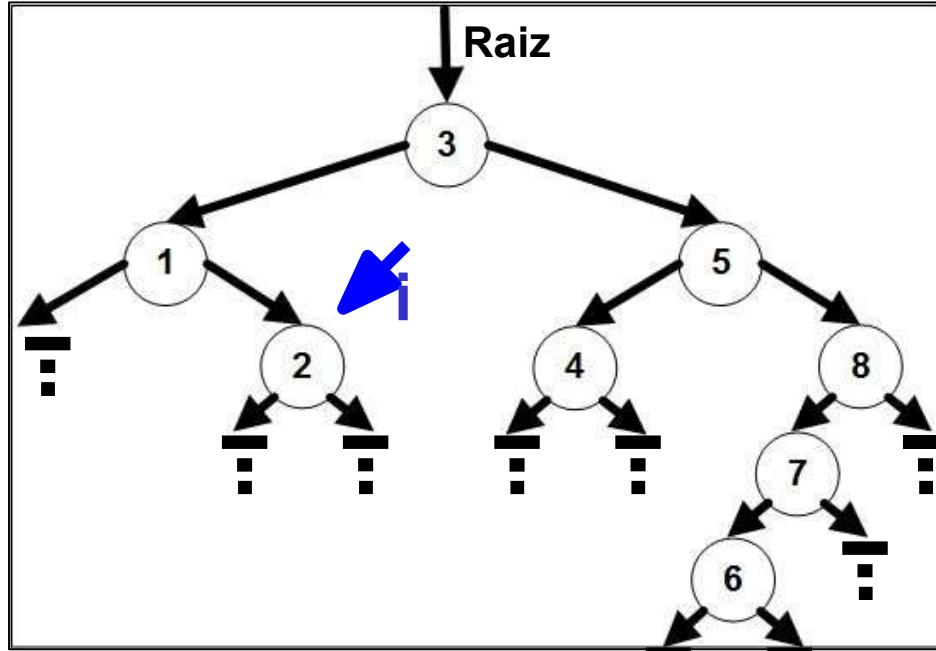
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 1 x 1 i n(3)

x 1 i n(1)



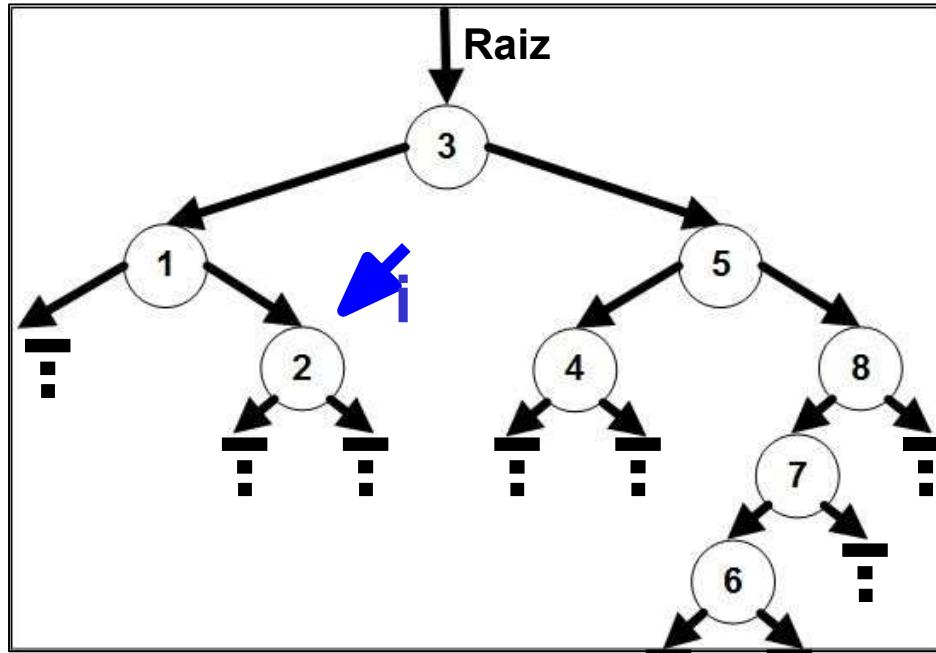
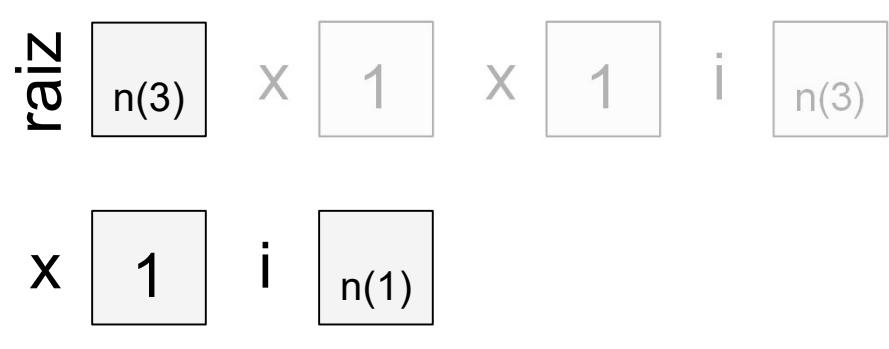
Algoritmo de Remoção em Java

```
//remover(1), um filho

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!");
    } else if(x < i.elemento){ i.esq = remover(x, i.esq);
    } else if(x > i.elemento) { i.dir = remover(x, i.dir);
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
Retorna n(2)

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



Algoritmo de Remoção em Java

```
//remover(1), um filho

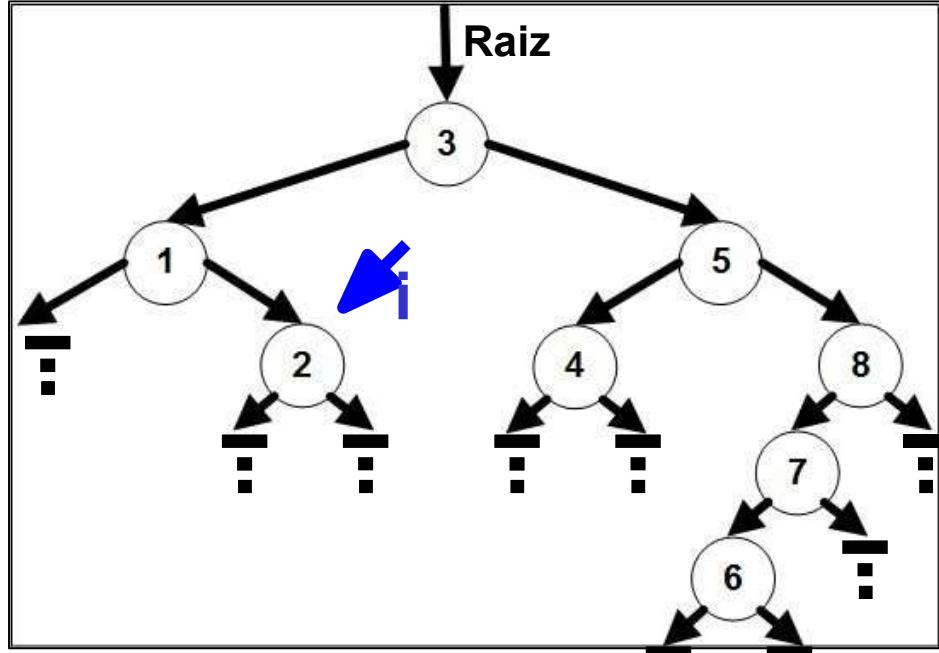
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 1 x 1 i n(3)

x 1 i n(1)



Algoritmo de Remoção em Java

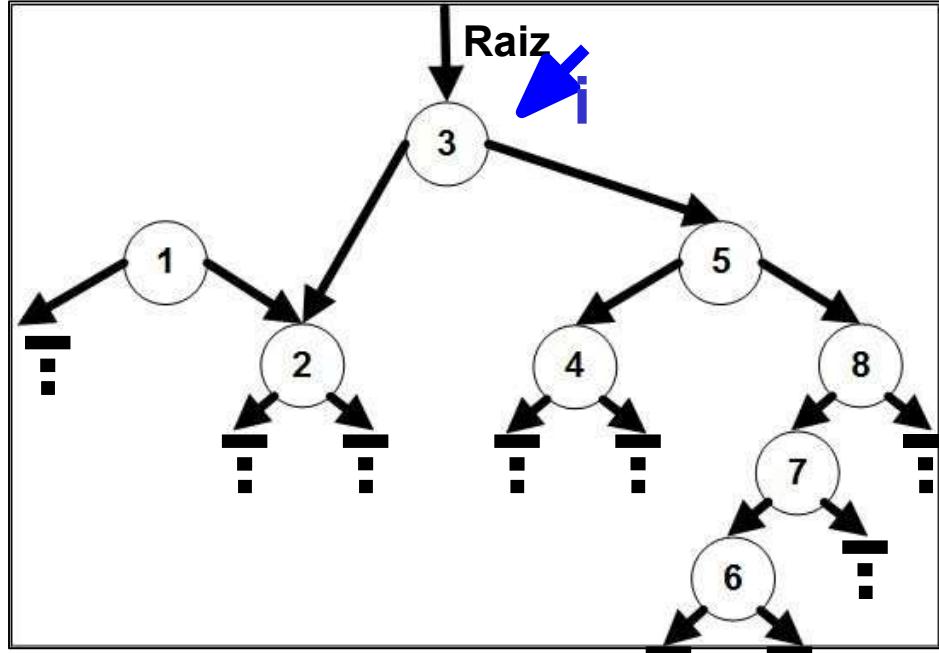
```
//remover(1), um filho

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq=remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 1 X 1 i n(3)



Algoritmo de Remoção em Java

```
//remover(1), um filho
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

No remo

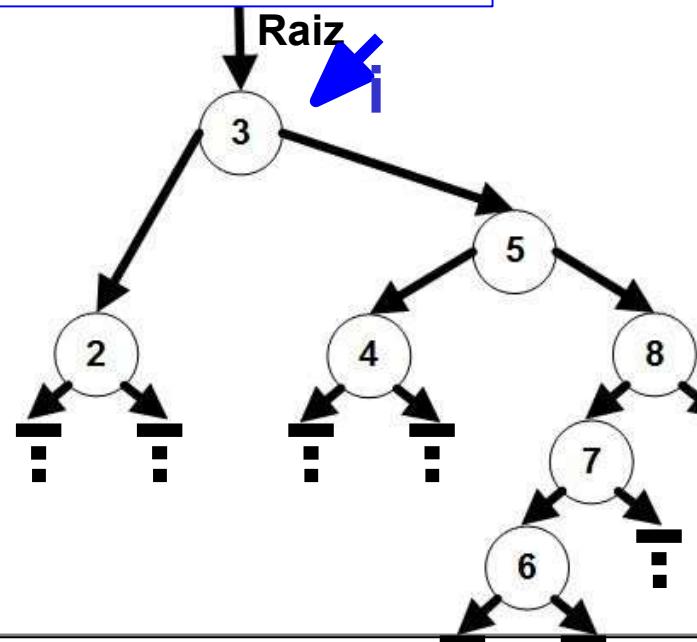
```
    if (i == null) {
        if (i == raiz)
            raiz = remover(x, raiz);
        else
            i = i.dir;
    } else if (i.esq == null) {
        i = i.esq;
    } else {
        i.esq = maiorEsq(i, i.esq);
    }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else { j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz	n(3)	x	1	x	1	i	n(3)
------	------	---	---	---	---	---	------

Após a coleta de lixo do Java

(que não controlamos quando ela acontece)...



Algoritmo de Remoção em Java

```
//remover(1), um filho
```

```
void remover(int x) throws Exception {
```

```
    raiz = remover(x, raiz);
```

```
}
```

```
No remo
```

```
    if (i ==
```

```
    } else
```

```
    } else
```

```
    } else if(i.dir == null) { i = i.esq;
```

```
    } else if(i.esq == null) { i = i.dir;
```

```
    } else { i.esq = maiorEsq(i, i.esq); }
```

```
    return i;
```

```
}
```

```
No maiorEsq(No i, No j) {
```

```
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
```

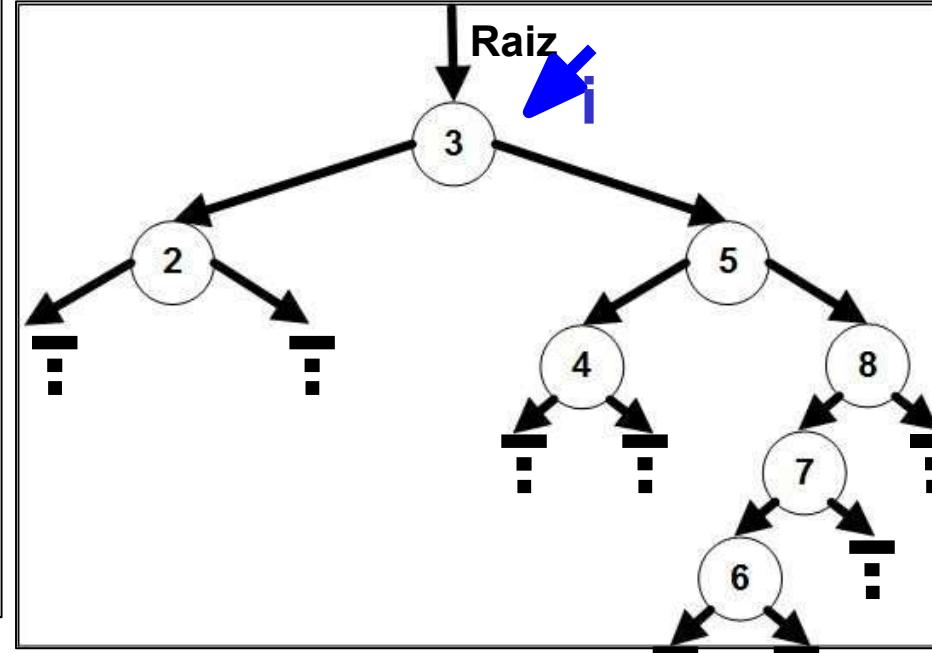
```
    else { j.dir = maiorEsq(i, j.dir); }
```

```
    return j;
```

```
}
```

raiz n(3) x 1 x 1 i n(3)

De uma forma mais organizada ...



Algoritmo de Remoção em Java

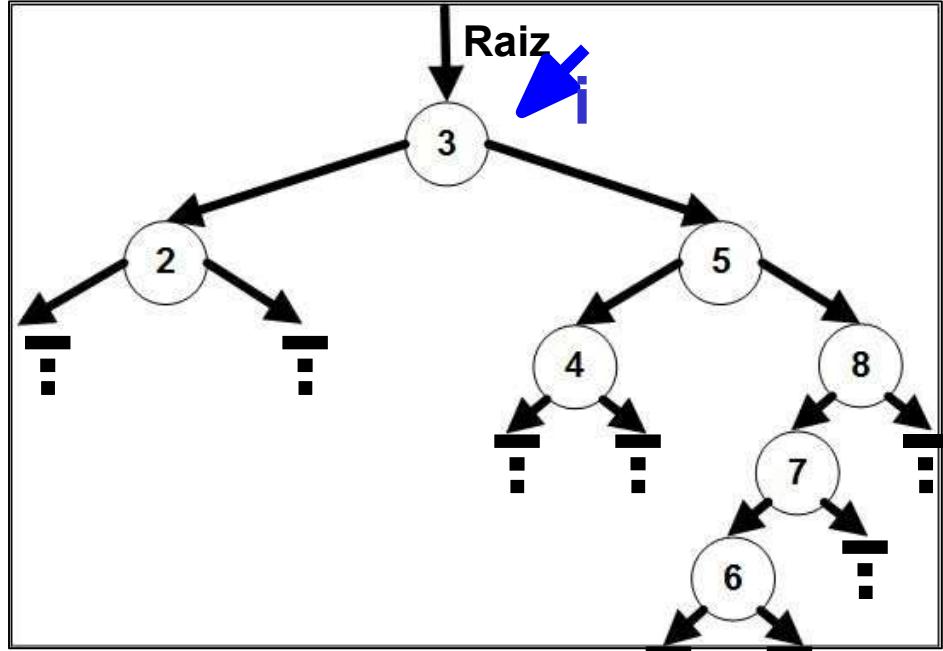
```
//remover(1), um filho

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    else if(i.dir == null) { i = i.esq; }
    else if(i.esq == null) { i = i.dir; }
    else { i.esq = maiorEsq(i, i.esq); }
    return i;
} Retorna n(3)

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else { j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 1 x 1 i n(3)



Algoritmo de Remoção em Java

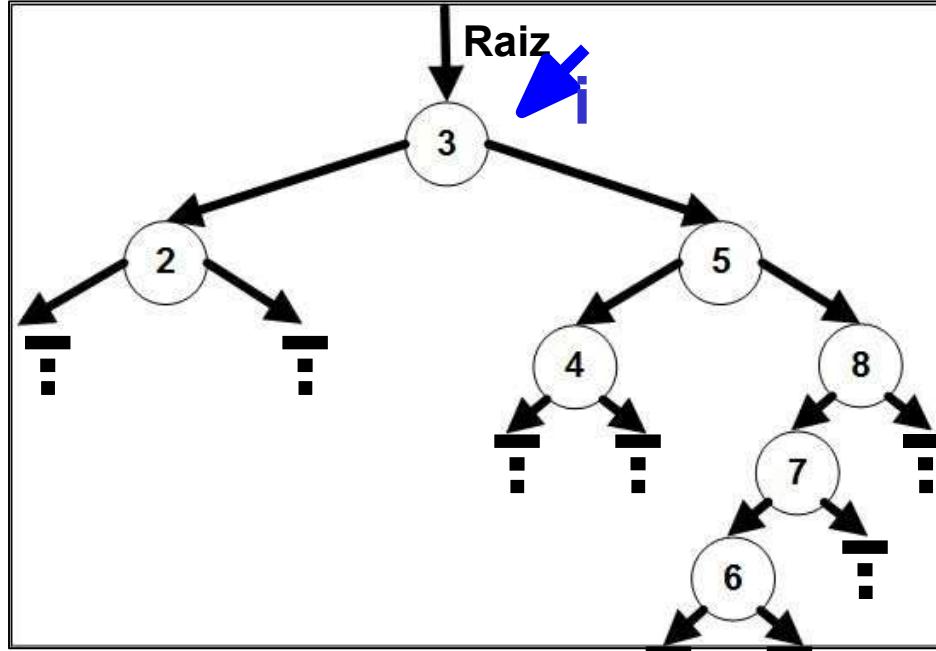
```
//remover(1), um filho

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    else if(i.dir == null) { i = i.esq; }
    else if(i.esq == null) { i = i.dir; }
    else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 1 X 1 i n(3)



Algoritmo de Remoção em Java

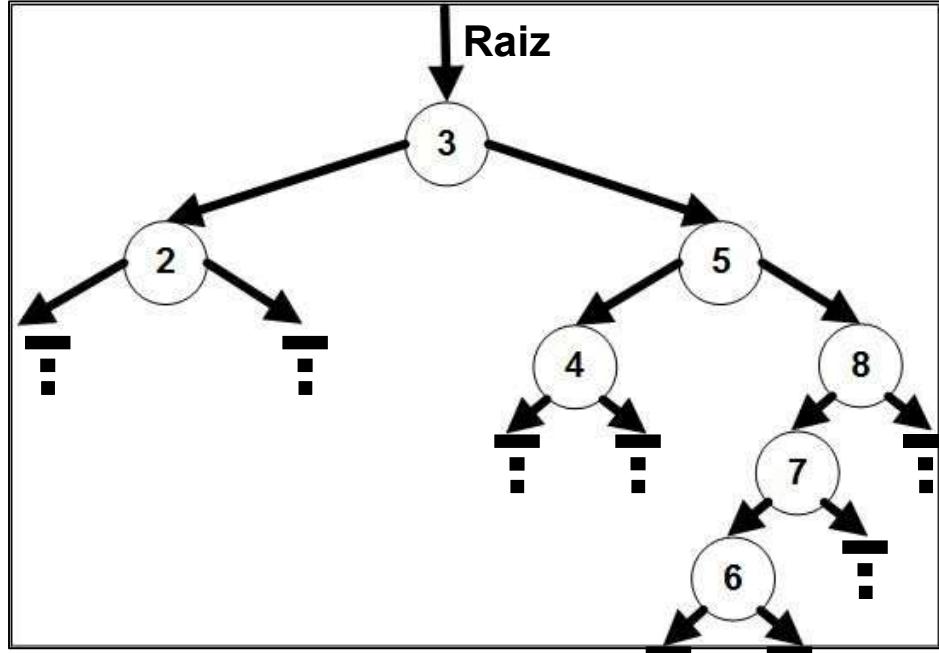
```
//remover(1), um filho

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    else if(i.dir == null) { i = i.esq; }
    else if(i.esq == null) { i = i.dir; }
    else { i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else { j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 1



Algoritmo de Remoção em Java

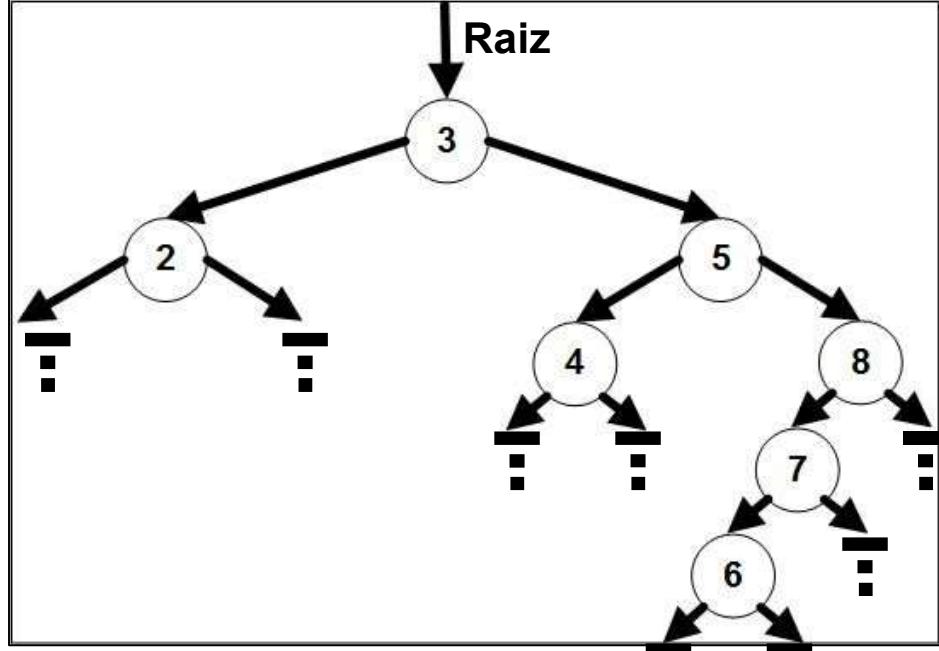
```
//remover(1), um filho
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz
n(3)

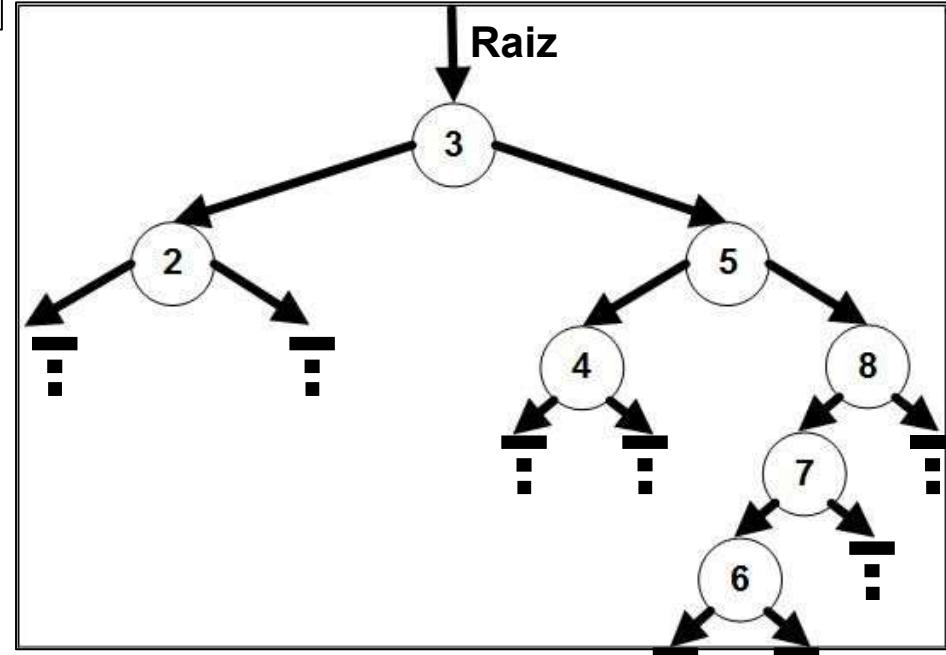


Algoritmo de Remoção em Java

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

raiz
n(3)

Voltando com o 1 antes
de fazer outra remoção

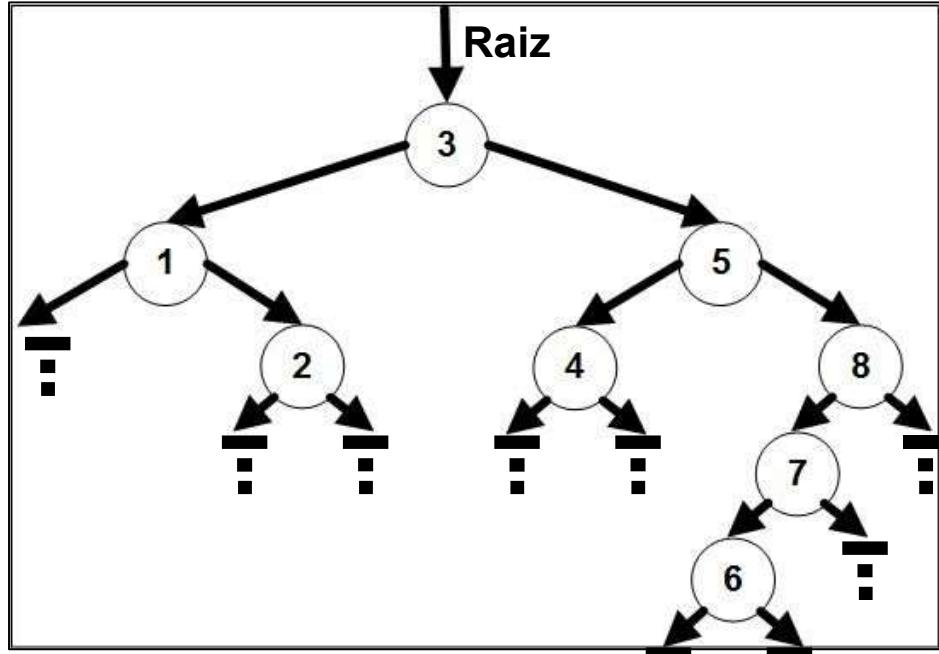


Algoritmo de Remoção em Java

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

raiz
n(3)

Vamos remover o 3 (tem dois filhos) de nossa árvore



Algoritmo de Remoção em Java

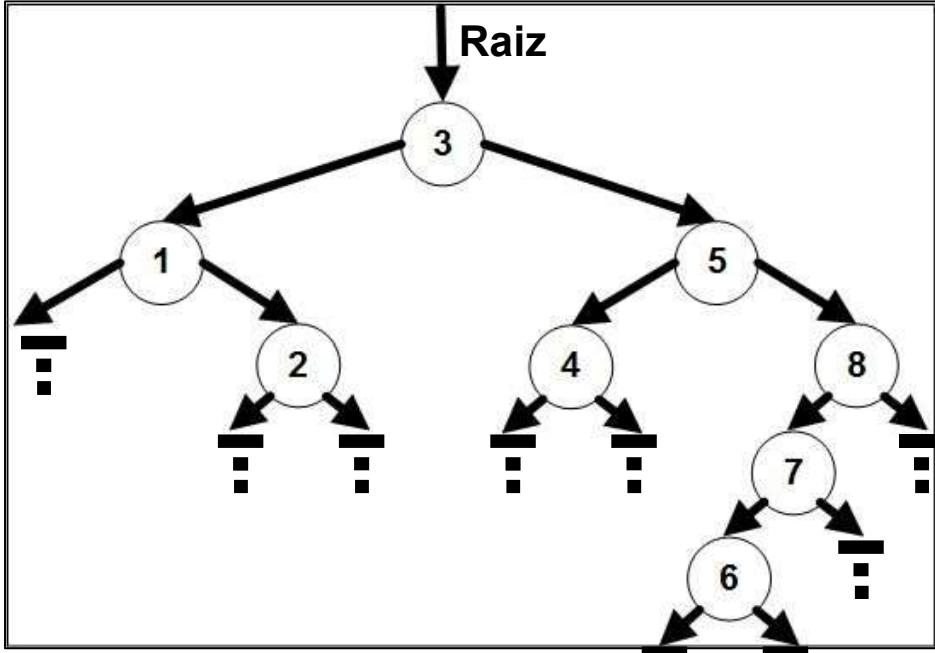
//remover(3), dois filhos

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!");
    } else if(x < i.elemento){ i.esq = remover(x, i.esq);
    } else if(x > i.elemento) { i.dir = remover(x, i.dir);
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) x 3



Algoritmo de Remoção em Java

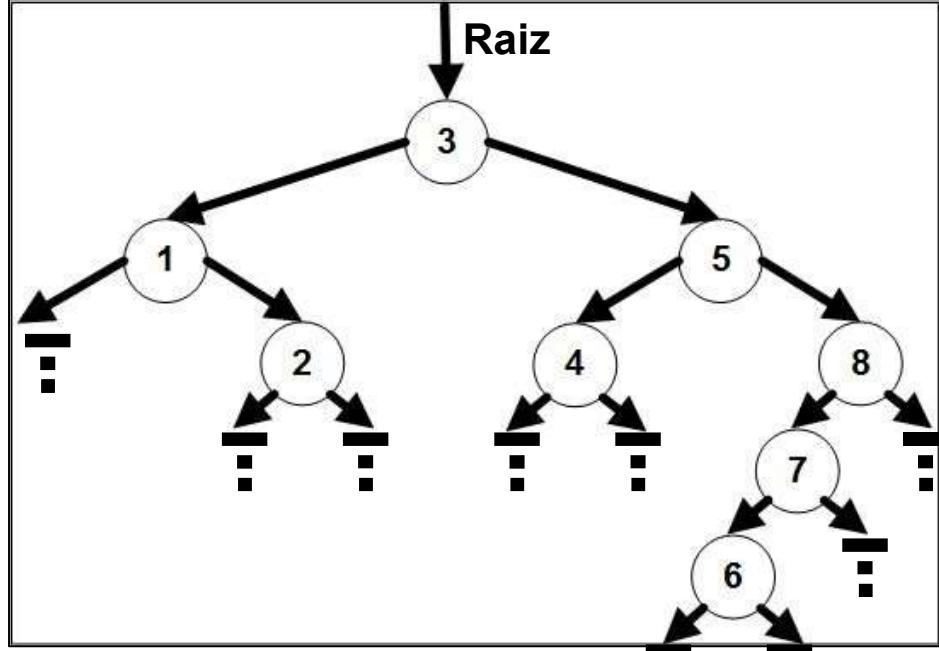
```
//remover(3), dois filhos
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 3



Algoritmo de Remoção em Java

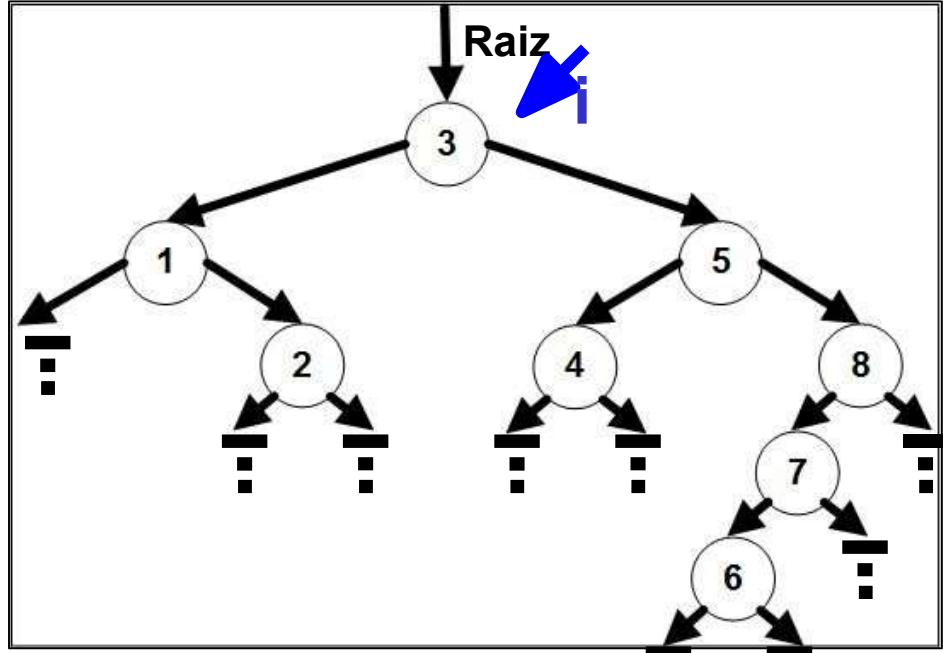
```
//remover(3), dois filhos
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 3 X 3 i n(3)



Algoritmo de Remoção em Java

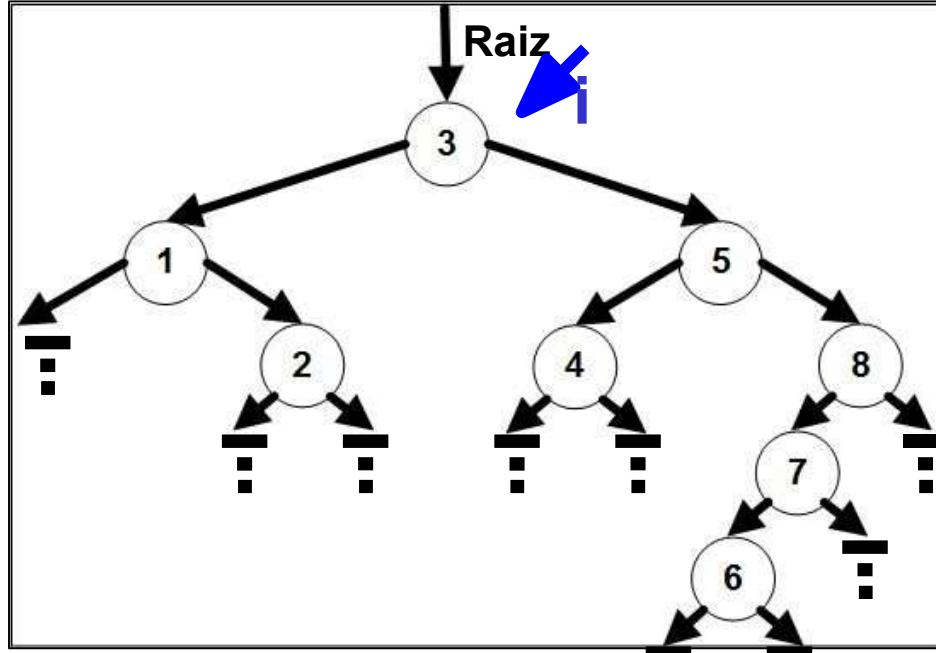
```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!");
    } else if(x < i.elemento){ i.esq = remover(x, i.esq);
    } else if(x > i.elemento) { i.dir = remover(x, i.dir);
    } else if(i.dir == null) {   i = i.esq;
    } else if(i.esq == null) {   i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
} false: n(3) == null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 3 X 3 i n(3)



Algoritmo de Remoção em Java

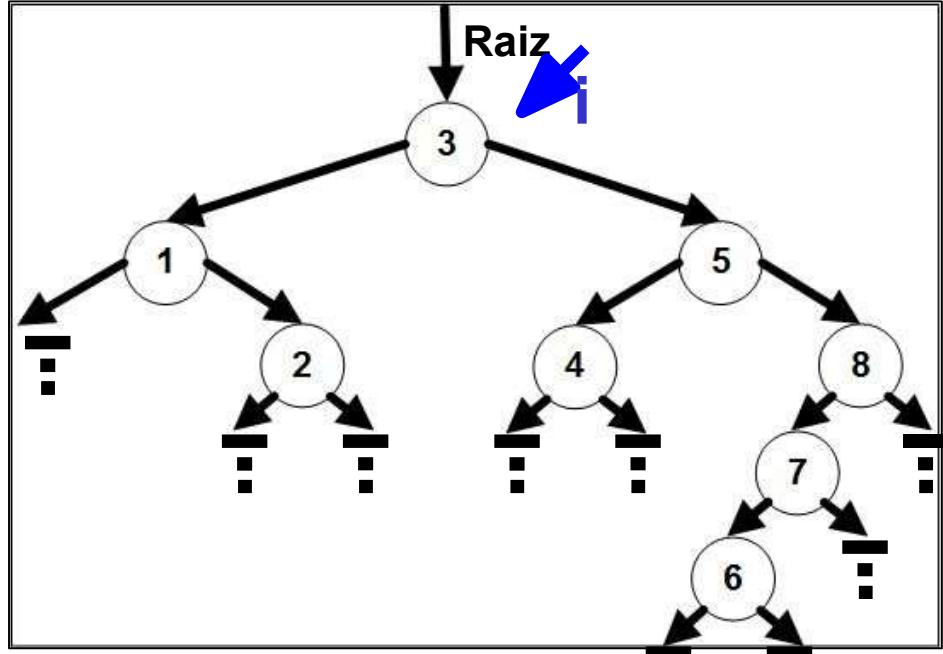
```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
false: 3 < 3

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 3 X 3 i n(3)



Algoritmo de Remoção em Java

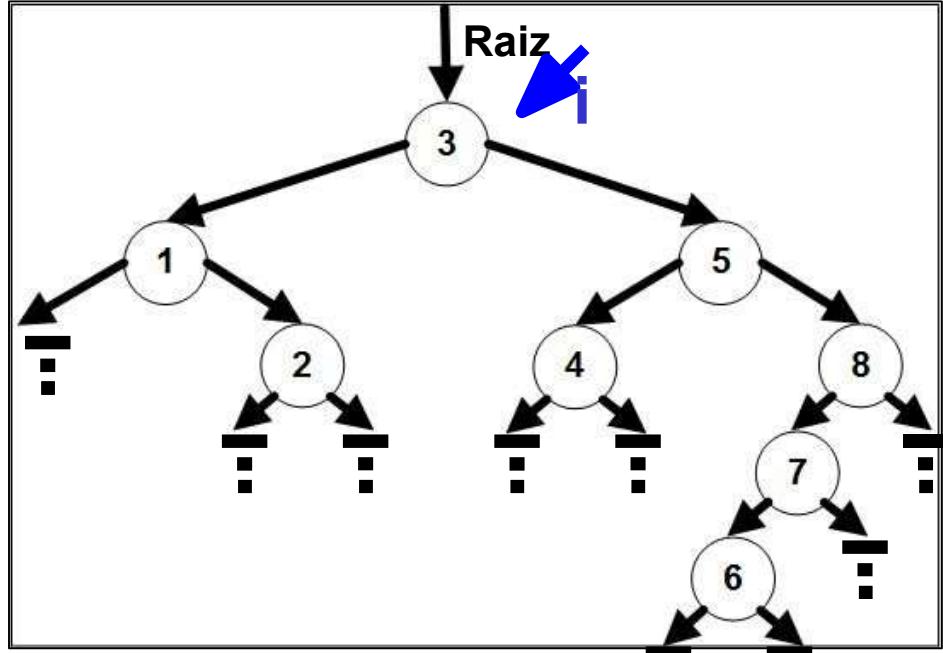
```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento){ i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
false: 3 > 3

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 3 X 3 i n(3)



Algoritmo de Remoção em Java

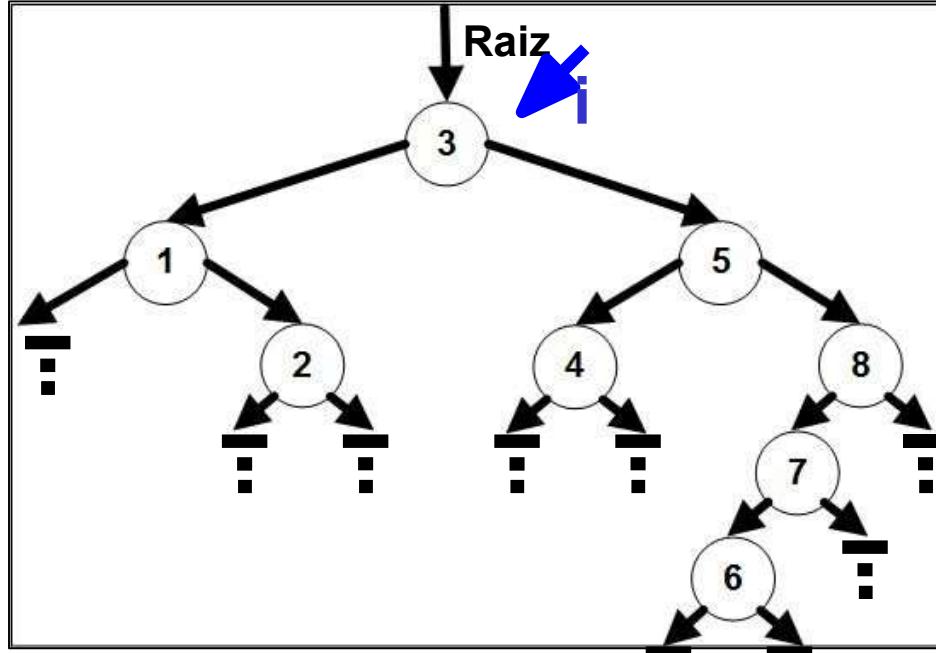
```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
} else if(i.dir == null) { i = i.esq;
} else if(i.esq == null) { i = i.dir;
} else {
    i.esq = maiorEsq(i, i.esq); }
return i;
} false: n(5) == false

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else { j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 3 X 3 i n(3)



Algoritmo de Remoção em Java

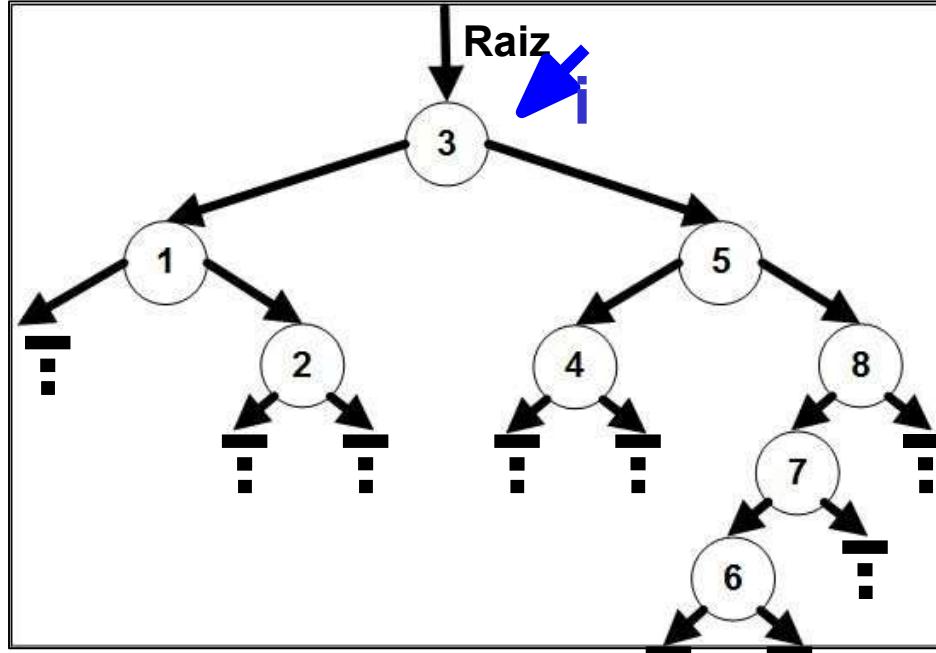
```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}                                false: n(1) == false

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else { j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 3 X 3 i n(3)



Algoritmo de Remoção em Java

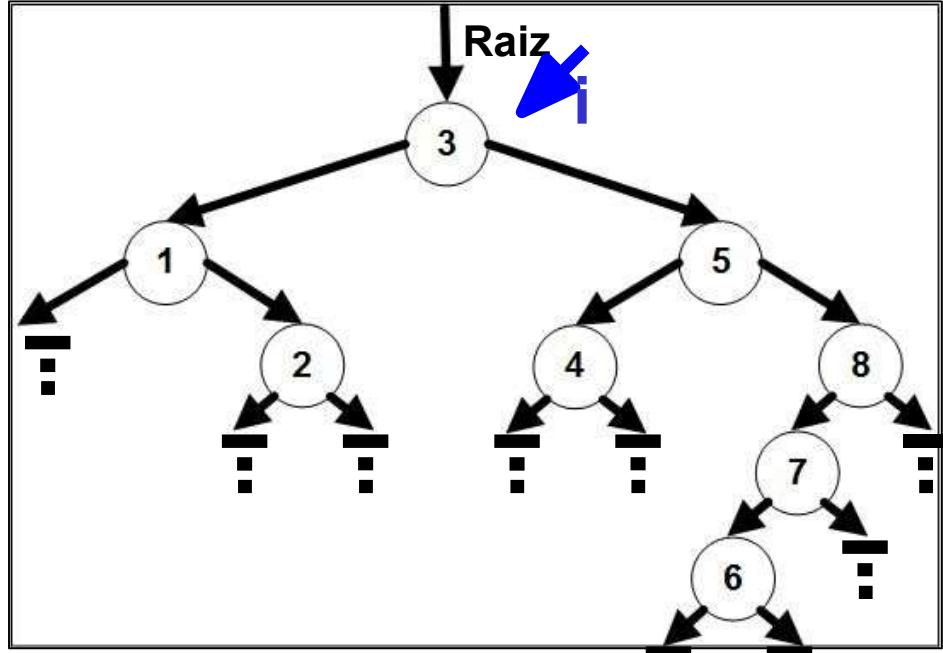
```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 3 X 3 i n(3)



Algoritmo de Remoção em Java

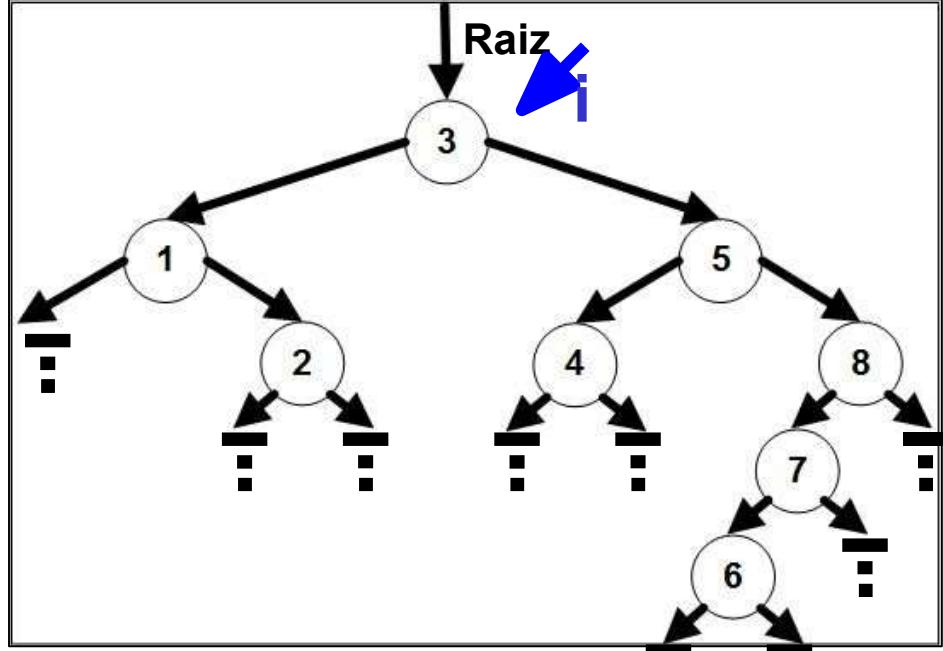
```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 3 X 3 i n(3)



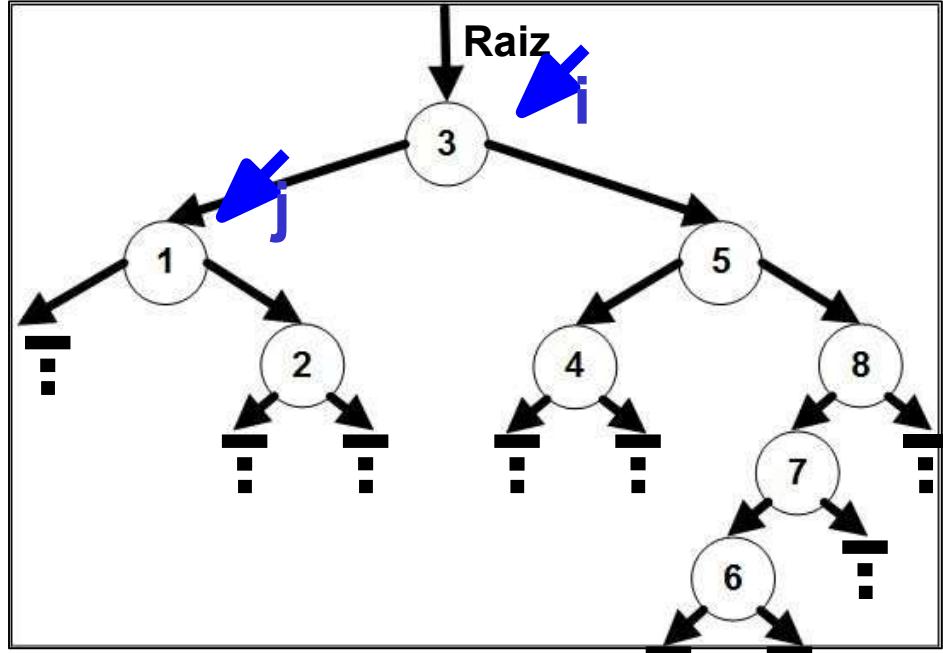
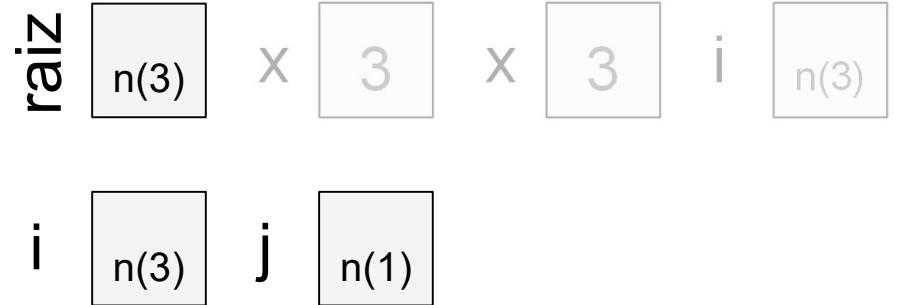
Algoritmo de Remoção em Java

```
//remover(3), dois filhos
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



Algoritmo de Remoção em Java

```
//remover(3), dois filhos

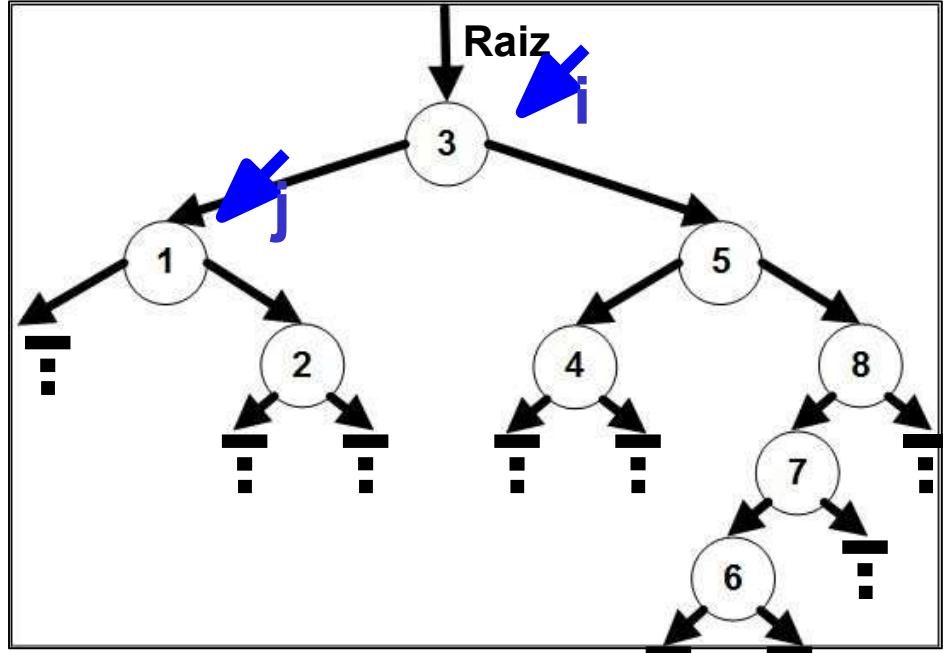
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}                                false: n(2) == null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(3) X 3 X 3 i n(3)

i n(3) j n(1)



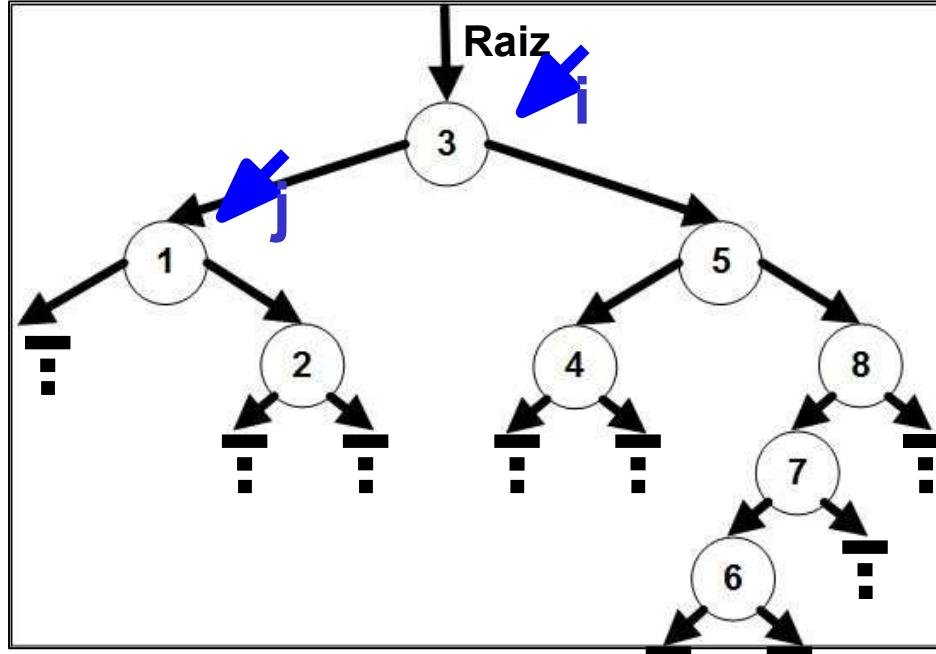
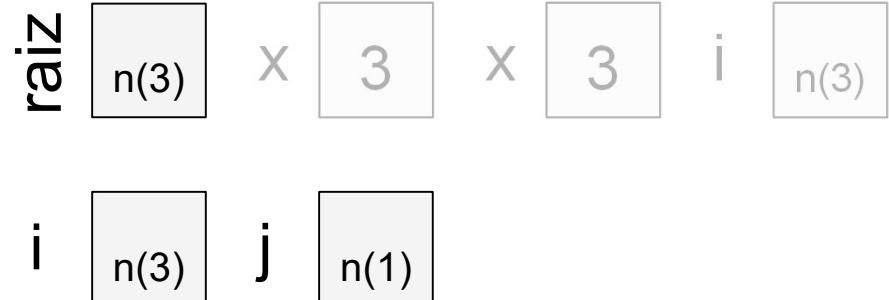
Algoritmo de Remoção em Java

```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



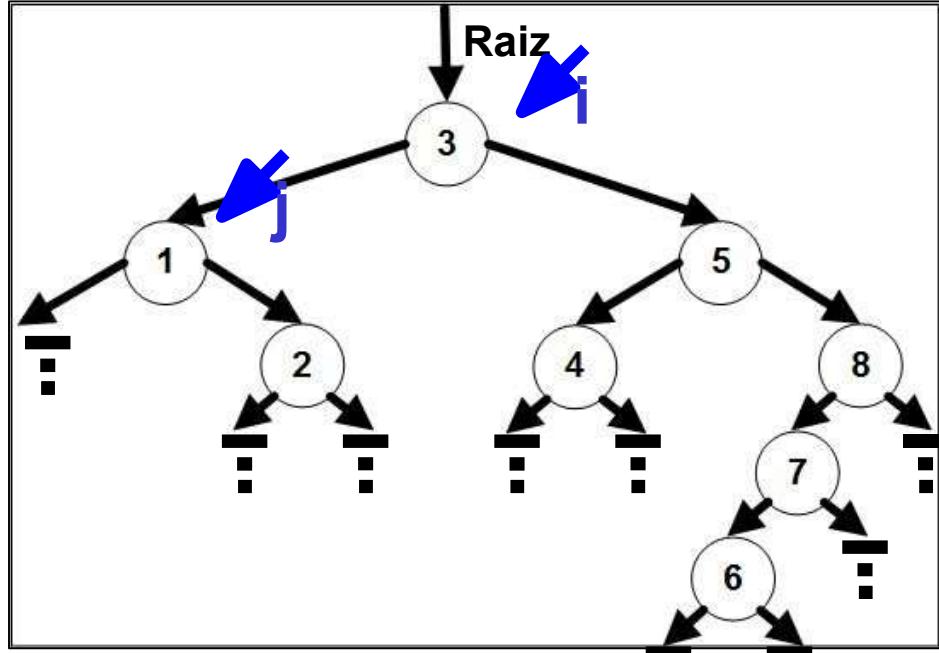
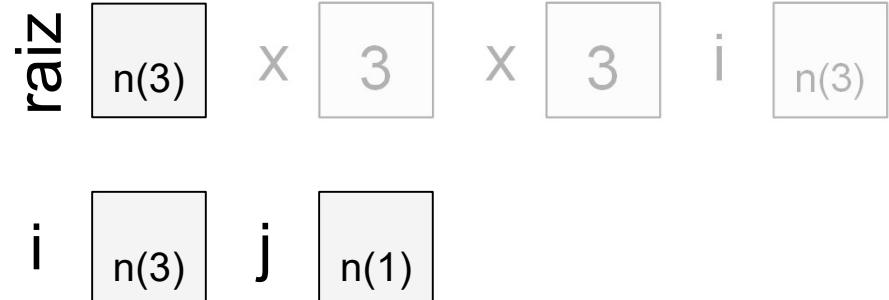
Algoritmo de Remoção em Java

```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



Algoritmo de Remoção em Java

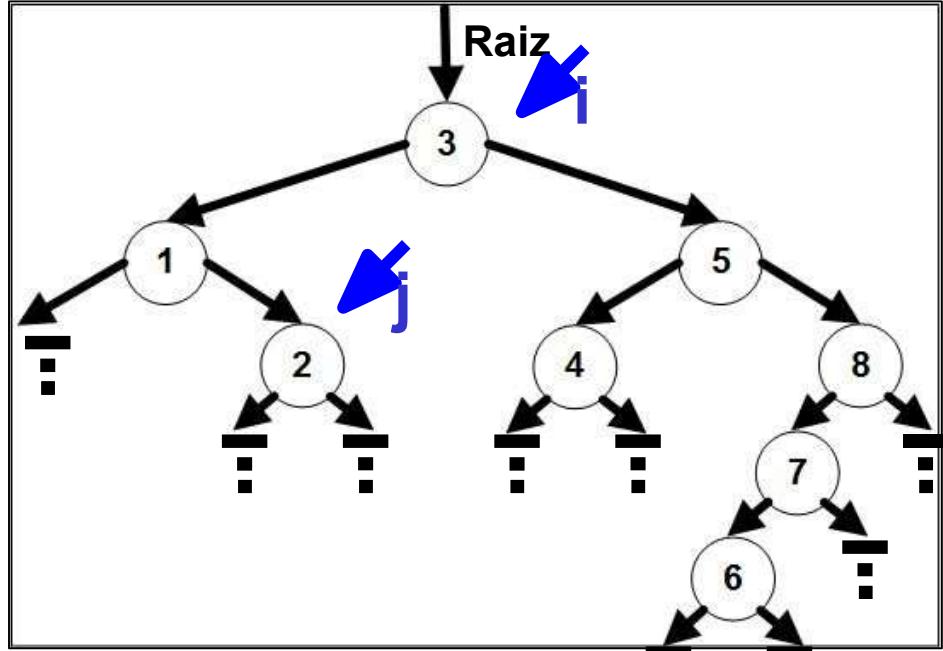
```
//remover(3), dois filhos
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz	n(3)	X	3	X	3	i	n(3)
i	n(3)	j	n(1)	i	n(3)	j	n(2)



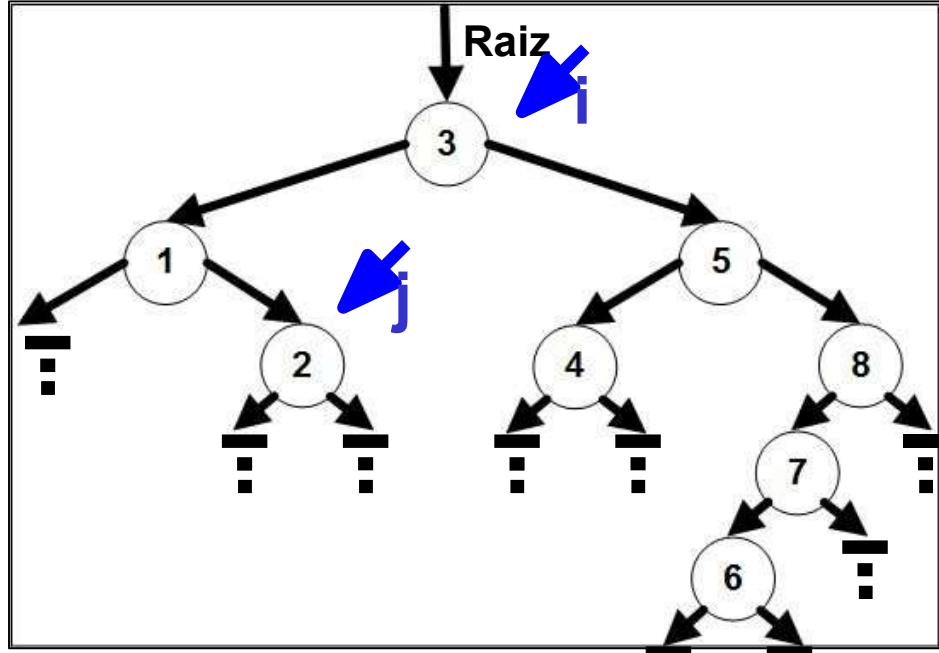
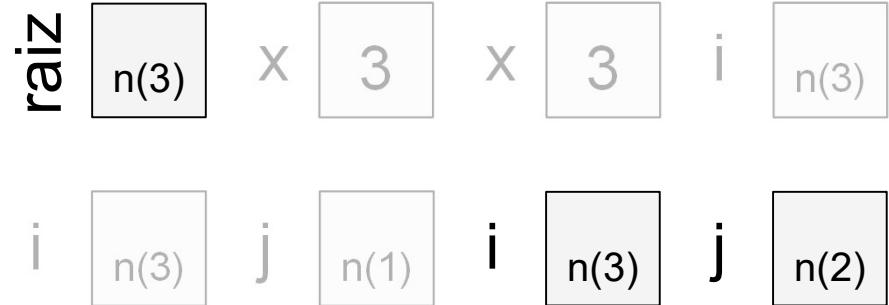
Algoritmo de Remoção em Java

```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
true: null == null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



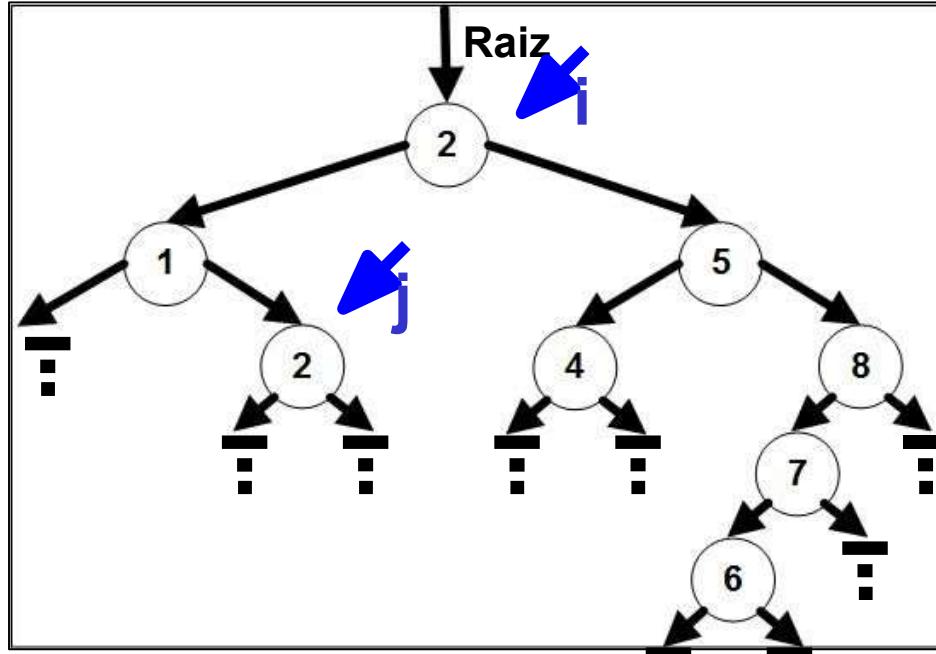
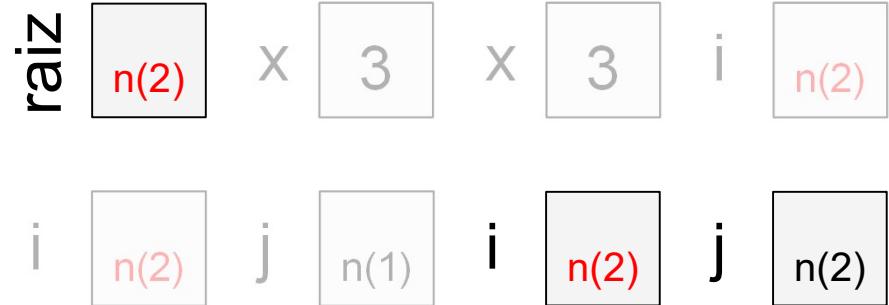
Algoritmo de Remoção em Java

```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento;j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



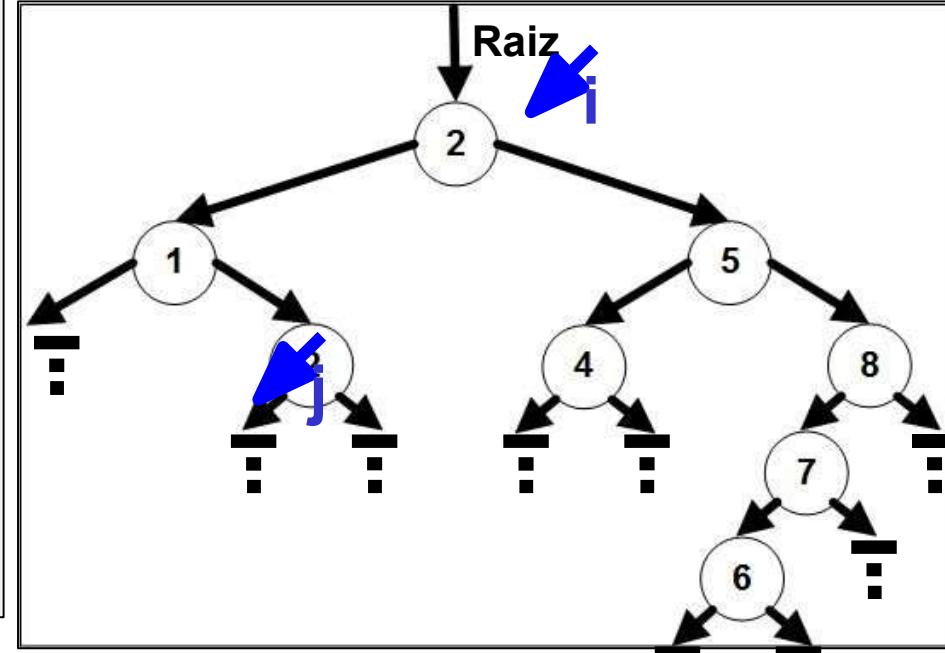
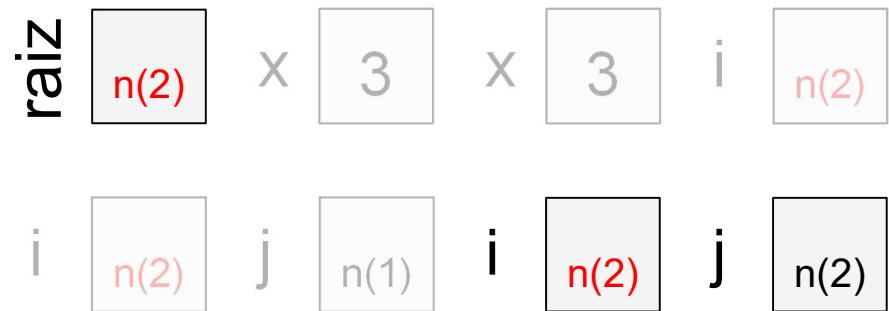
Algoritmo de Remoção em Java

```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



Algoritmo de Remoção em Java

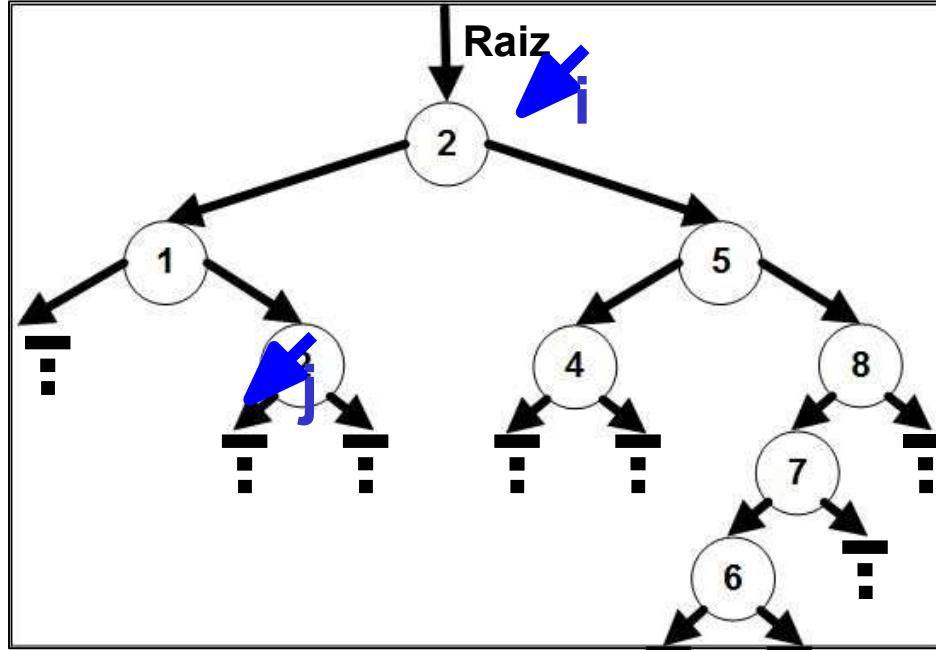
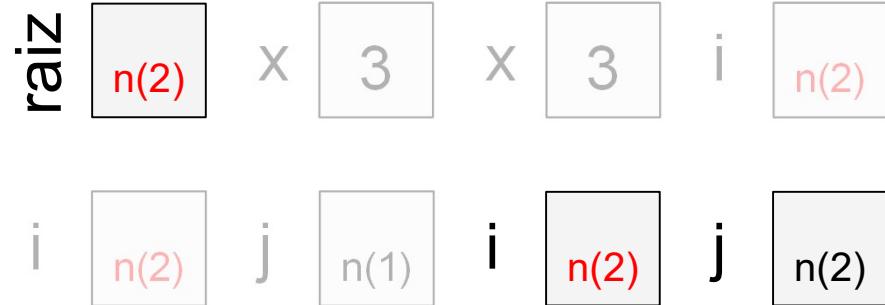
```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

Retornando null

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



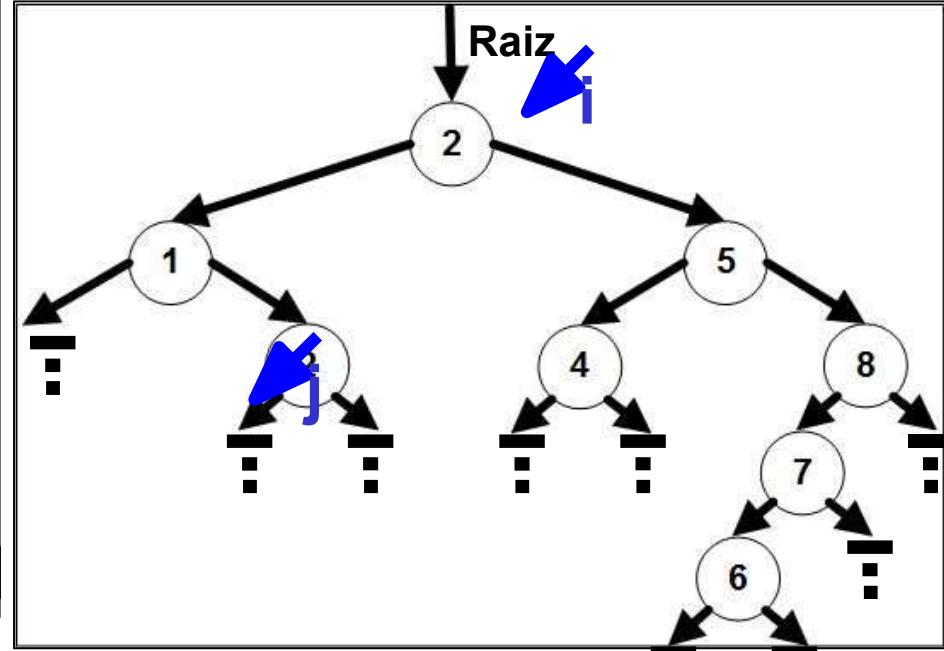
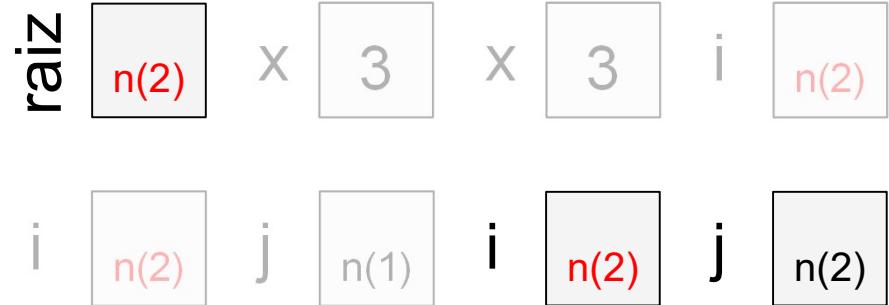
Algoritmo de Remoção em Java

```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



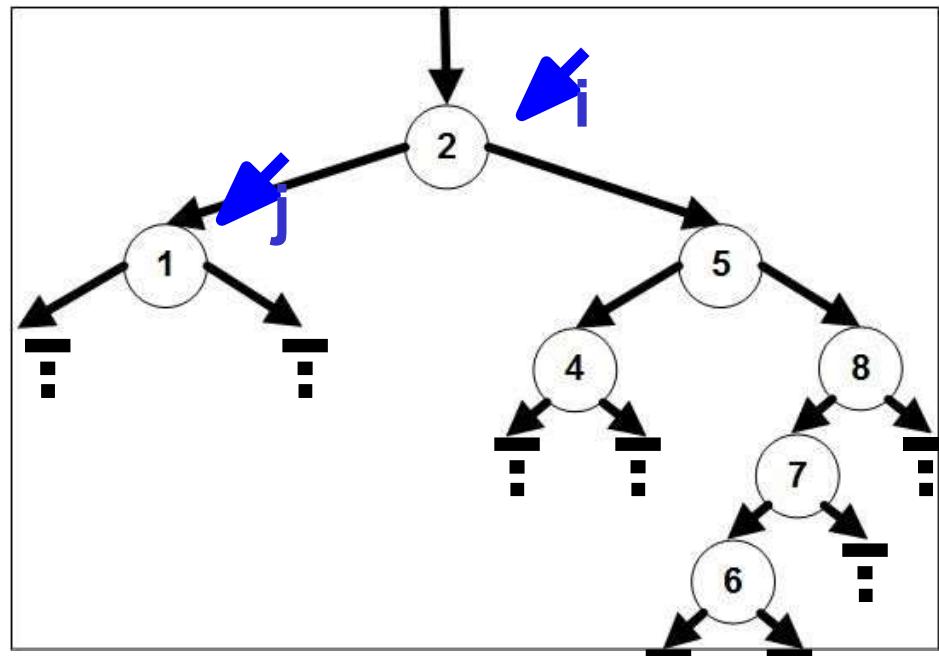
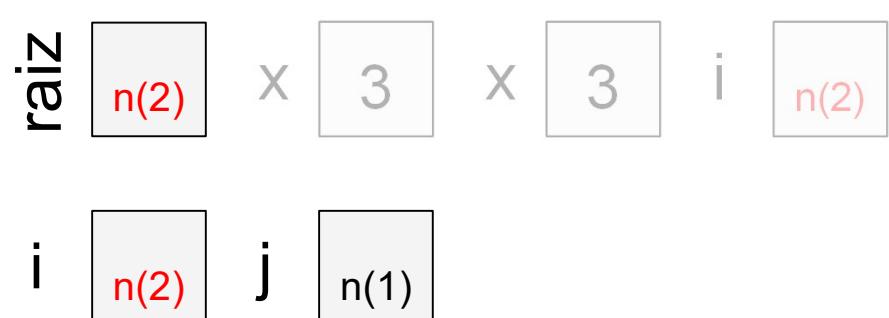
Algoritmo de Remoção em Java

```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```



Algoritmo de Remoção em Java

```
//remover(3), dois filhos
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

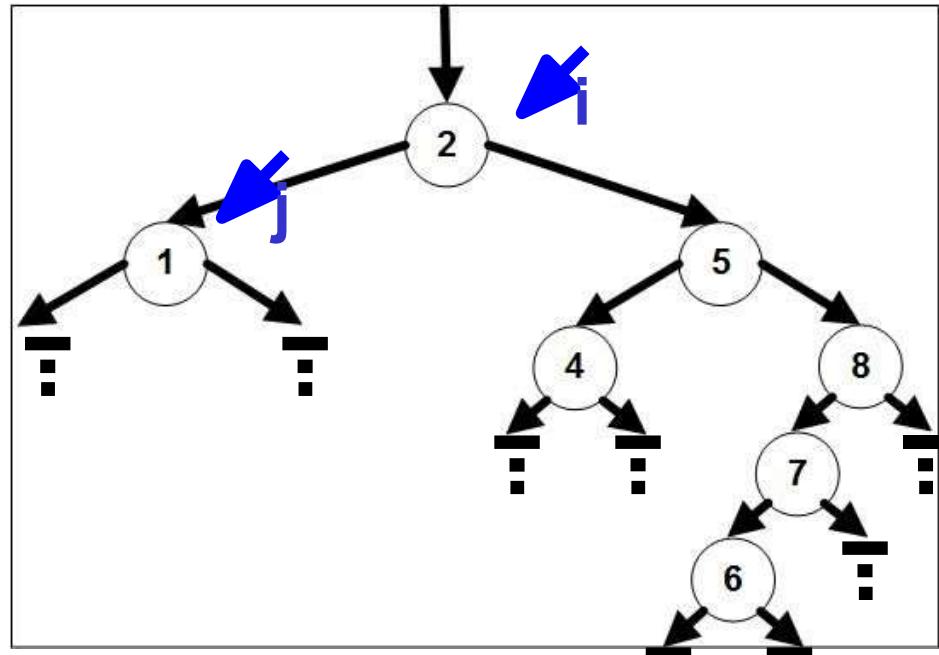
```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

Retornando n(1)

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir);
    }
    return j;
}
```

raiz n(2) x 3 x 3 i n(2)

i n(2) j n(1)



Algoritmo de Remoção em Java

```
//remover(3), dois filhos

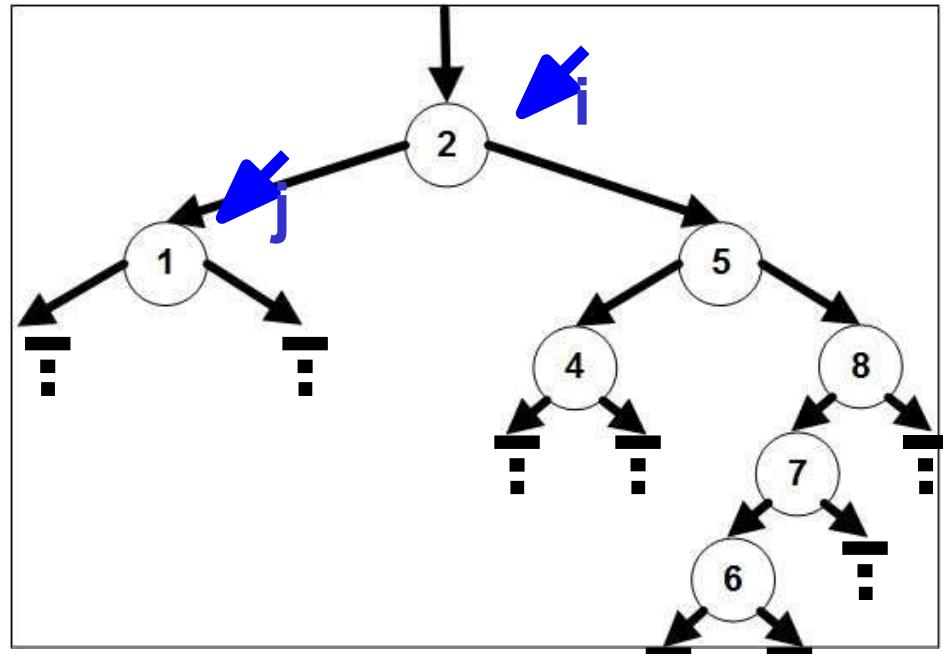
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(2) x 3 x 3 i n(2)

i n(2) j n(1)



Algoritmo de Remoção em Java

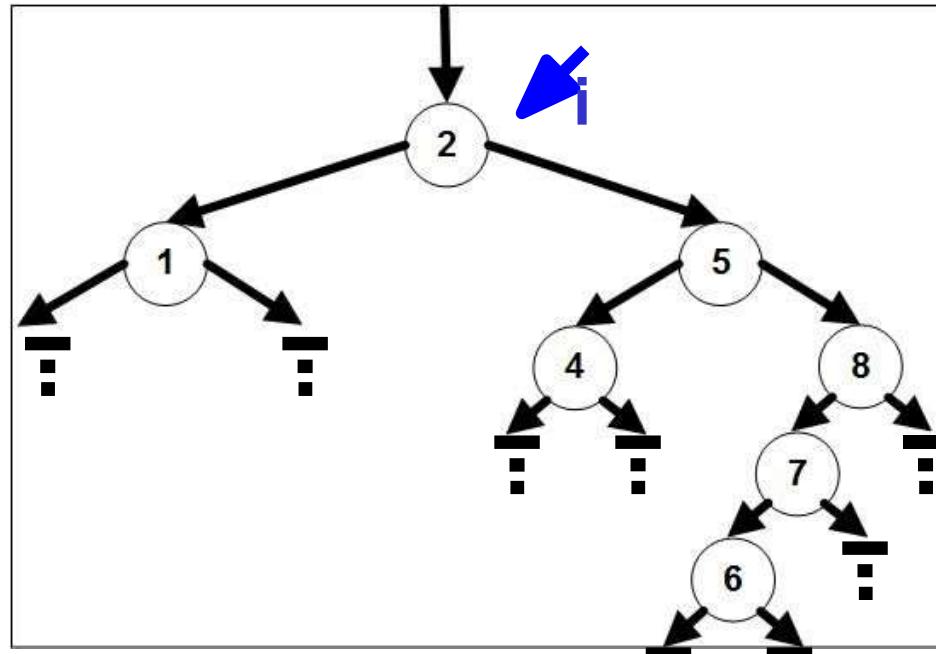
```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq; }
    } else if(i.esq == null) { i = i.dir; }
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(2) X 3 X 3 i n(2)



Algoritmo de Remoção em Java

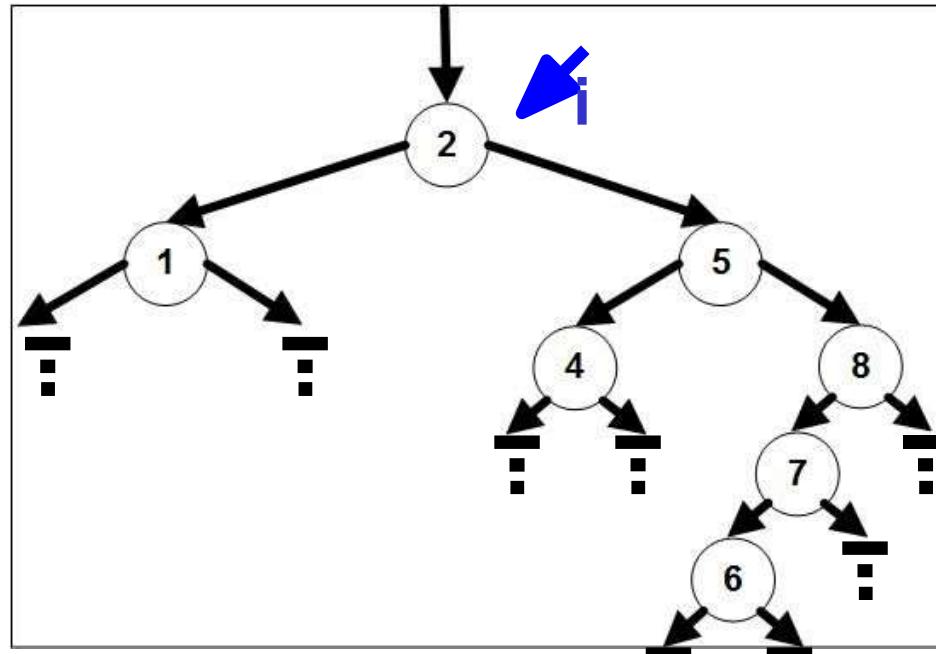
```
//remover(3), dois filhos

void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}

No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!");
    } else if(x < i.elemento){ i.esq = remover(x, i.esq);
    } else if(x > i.elemento) { i.dir = remover(x, i.dir);
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
Retorna n(2)

No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(2) X 3 X 3 i n(2)



Algoritmo de Remoção em Java

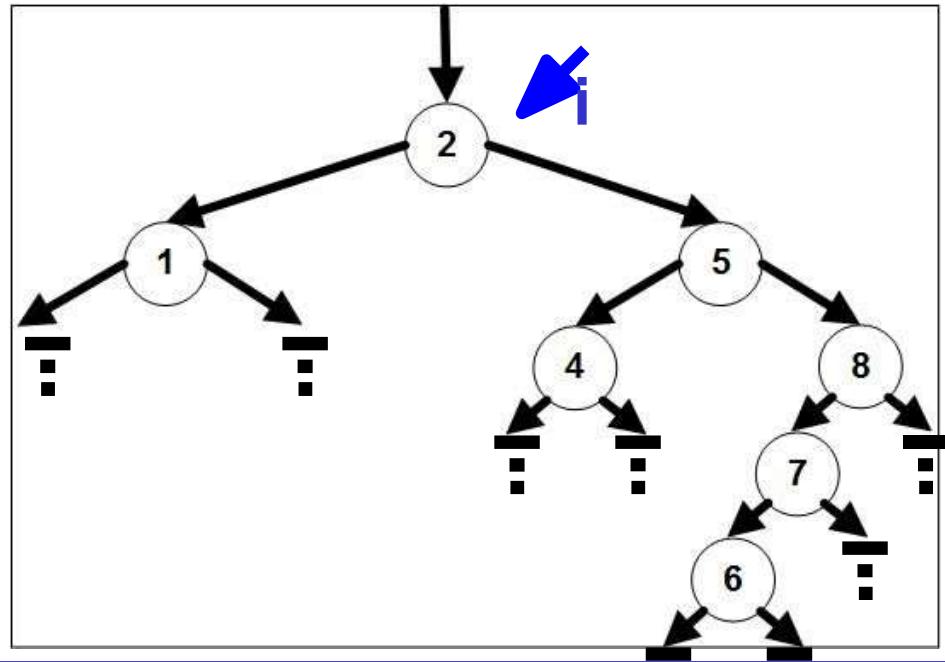
```
//remover(3), dois filhos
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(2) X 3 X 3 i n(2)



Algoritmo de Remoção em Java

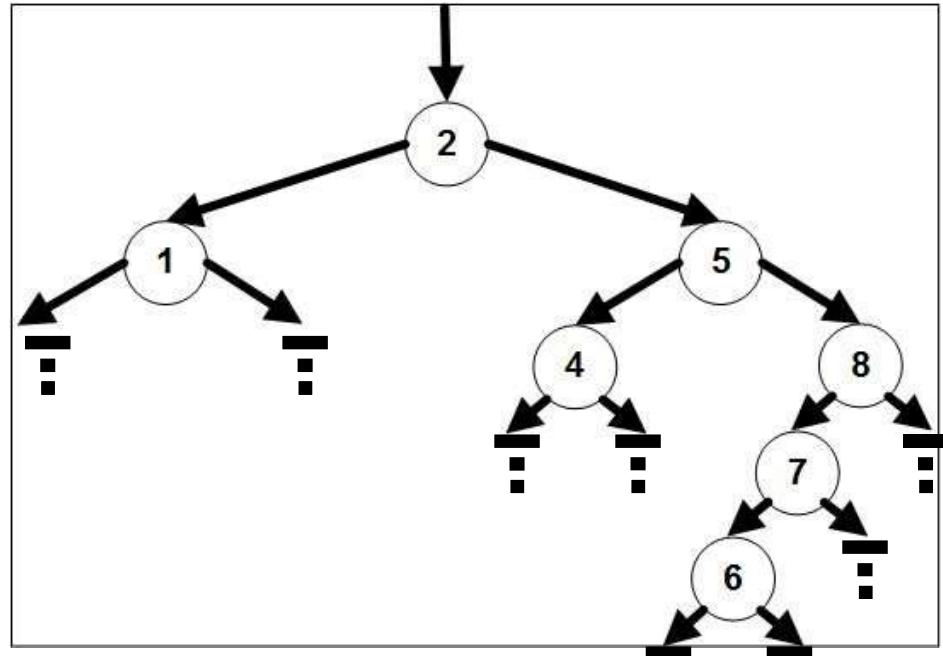
```
//remover(3), dois filhos
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz n(2) x 3



Algoritmo de Remoção em Java

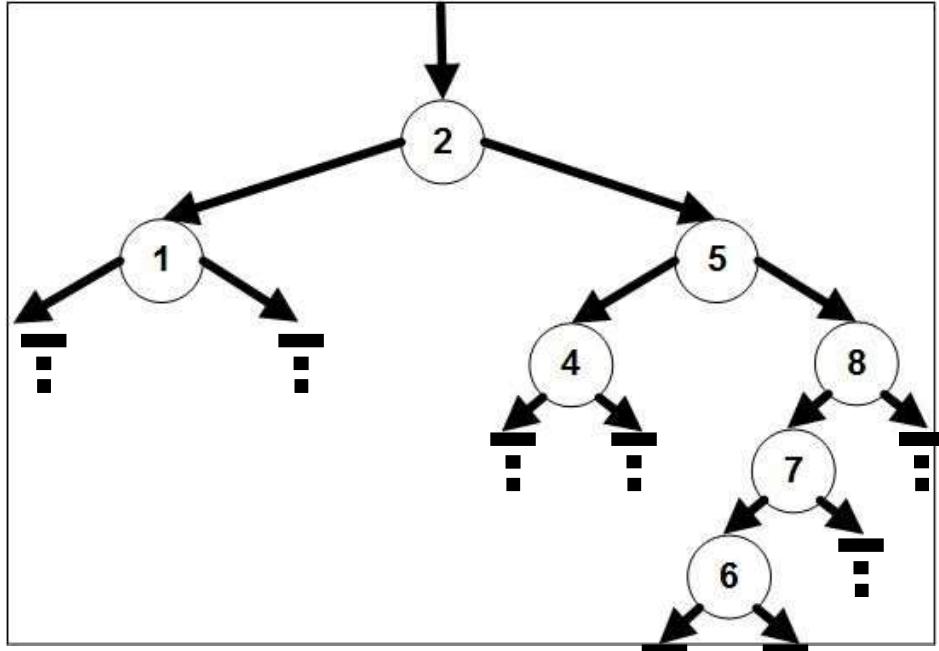
```
//remover(3), dois filhos
```

```
void remover(int x) throws Exception {
    raiz = remover(x, raiz);
}
```

```
No remover(int x, No i) throws Exception {
    if (i == null) { throw new Exception("Erro!"); }
    } else if(x < i.elemento){ i.esq = remover(x, i.esq); }
    } else if(x > i.elemento) { i.dir = remover(x, i.dir); }
    } else if(i.dir == null) { i = i.esq;
    } else if(i.esq == null) { i = i.dir;
    } else {
        i.esq = maiorEsq(i, i.esq); }
    return i;
}
```

```
No maiorEsq(No i, No j) {
    if (j.dir == null){ i.elemento=j.elemento; j=j.esq; }
    } else {
        j.dir = maiorEsq(i, j.dir); }
    return j;
}
```

raiz
n(2)



Análise de complexidade da Remoção

- **Melhor Caso:** $\Theta(1)$ comparações e acontece, por exemplo, na raiz
- **Pior Caso:** $\Theta(n)$ comparações e acontece, por exemplo, quando inserimos os elementos em ordem e o elemento desejado remover na folha
- **Caso Médio:** $\Theta(\lg(n))$ comparações e acontece, por exemplo, quando a árvore está balanceada e desejamos remover um elemento localizado em uma das folhas

Próxima aula

- (AVL) Balanceamento de árvore binária