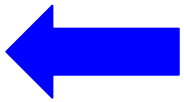
Unidade VII: Árvore Binária - Pesquisa e Caminhamento

Instituto de Ciências Exatas e Informática
Departamento de Ciência da Computação

Agenda

- Funcionamento básico da Pesquisa
- Algoritmo pesquisa em Java
- Análise de complexidade da Pesquisa
- Caminhamento

Agenda

- **Funcionamento básico da Pesquisa** 
- Algoritmo pesquisa em Java
- Análise de complexidade da Pesquisa
- Caminhamento

Funcionamento Básico da Pesquisa

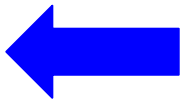
(1) Se a raiz estiver vazia, retornar uma **pesquisa negativa**

(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**

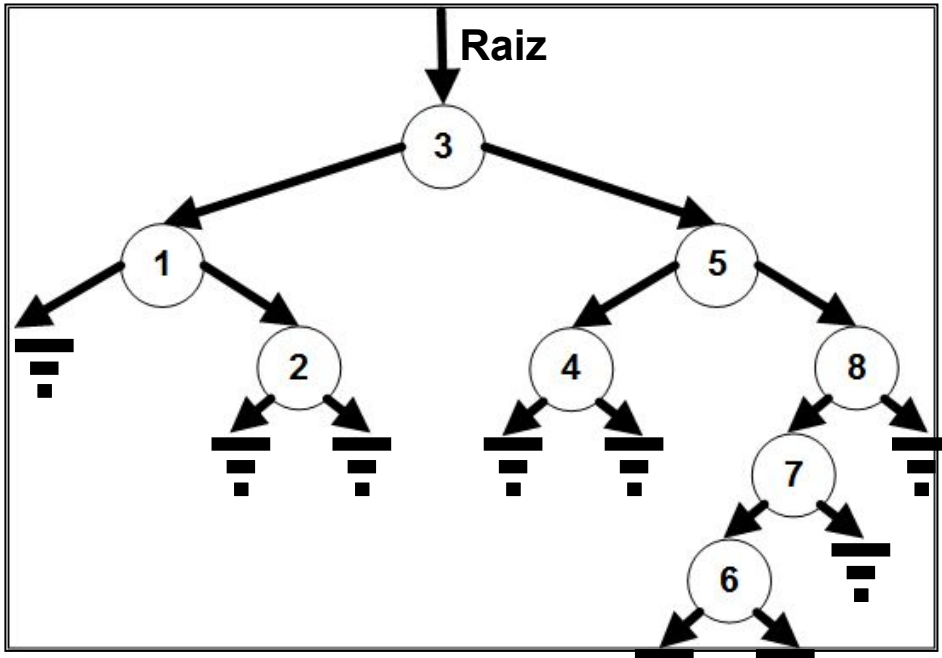
Agenda

- Funcionamento básico da Pesquisa
- **Algoritmo pesquisa em Java** 
- Análise de complexidade da Pesquisa
- Caminhamento

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

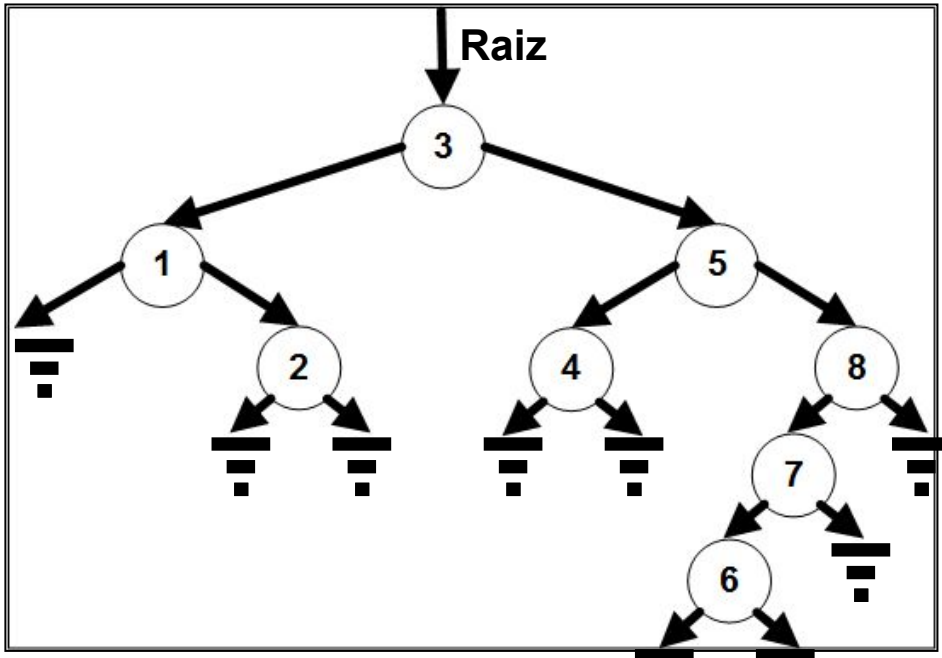


Algoritmo de Pesquisa em Java

```
//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;
}
```

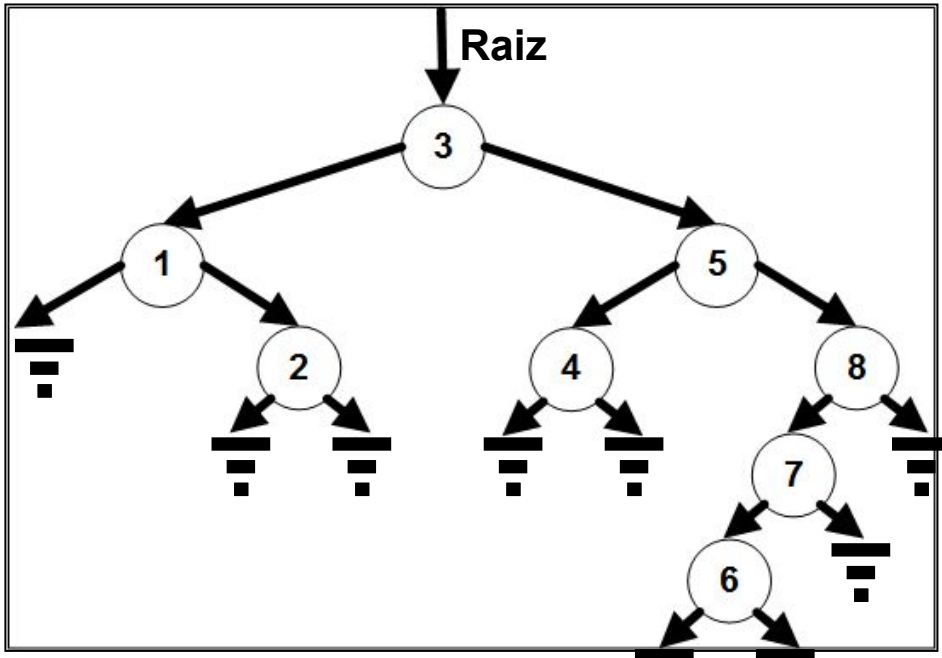


Algoritmo de Pesquisa em Java

```
//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;
}
```

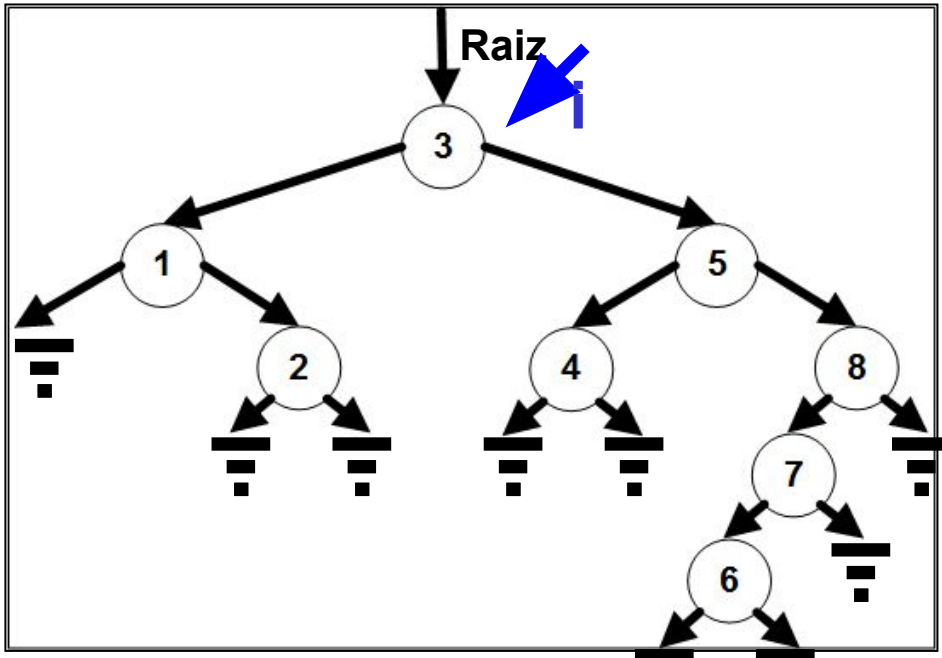


Algoritmo de Pesquisa em Java

```
//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;
}
```

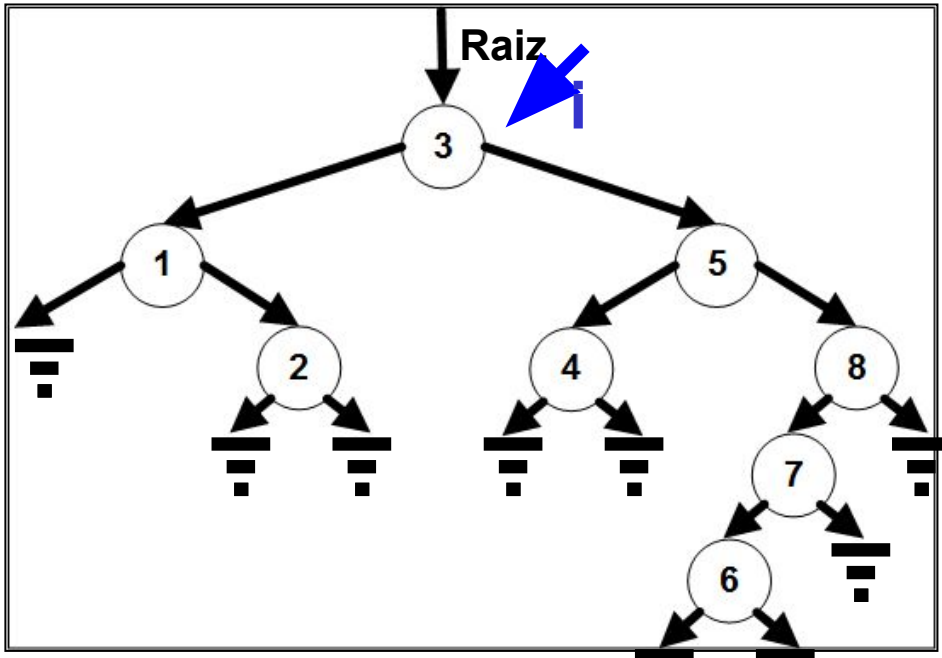


Algoritmo de Pesquisa em Java

```
//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;
}
```



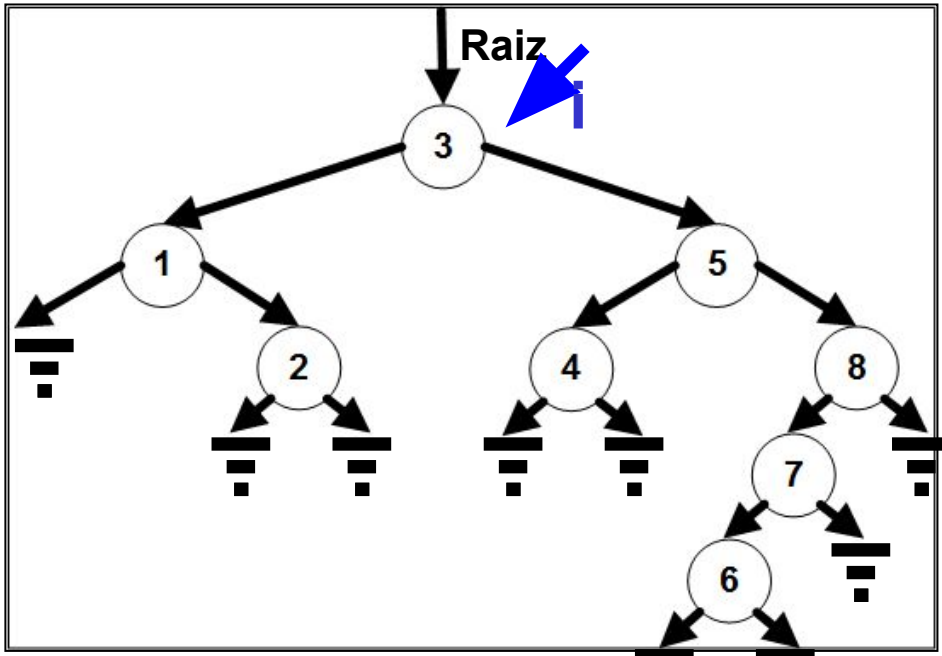
Algoritmo de Pesquisa em Java

```
//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



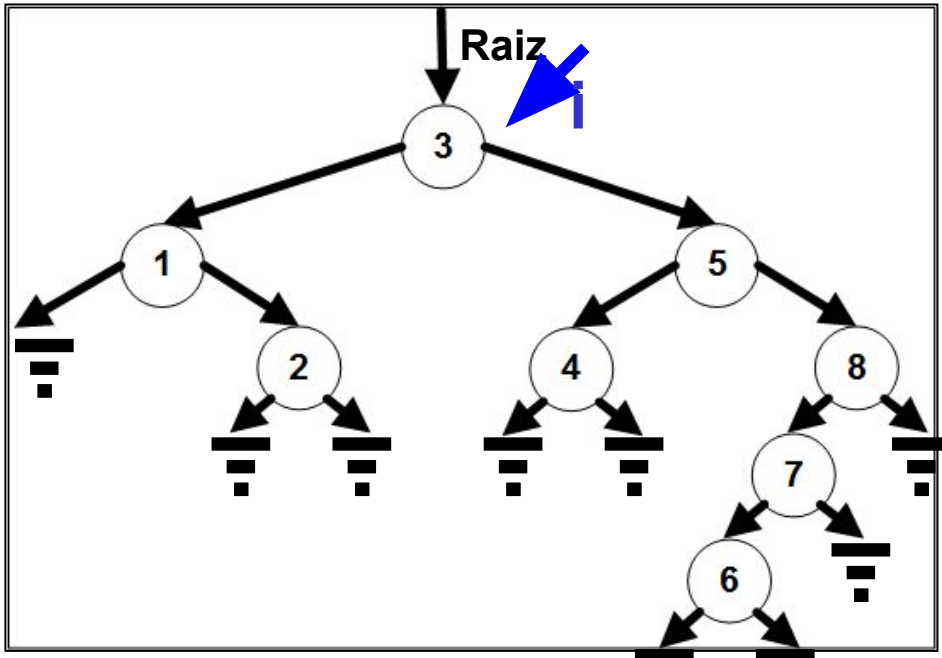
Algoritmo de Pesquisa em Java

```
//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



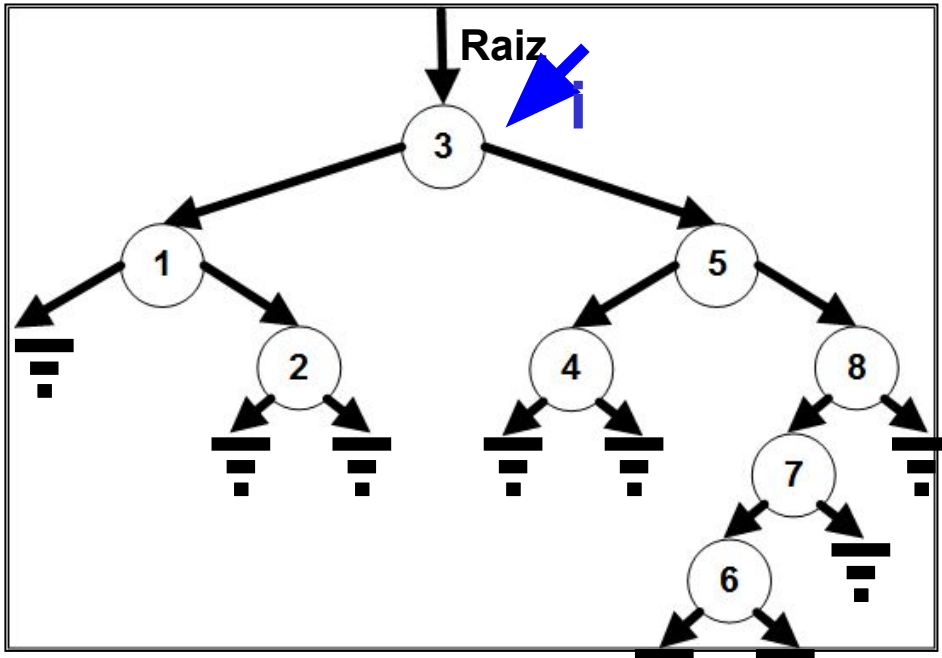
Algoritmo de Pesquisa em Java

```
//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

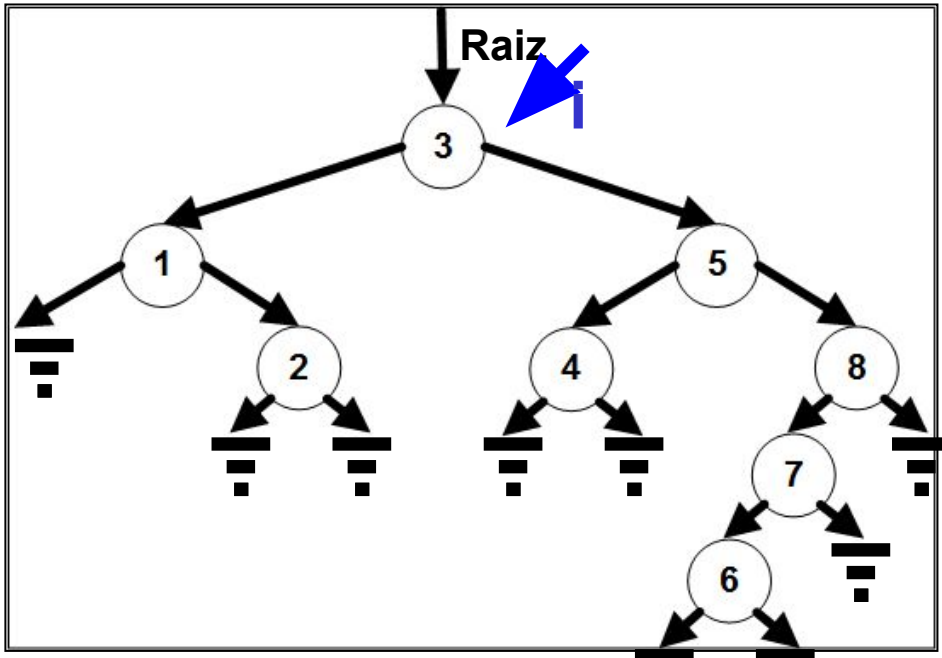


Algoritmo de Pesquisa em Java

```
//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;
}
```

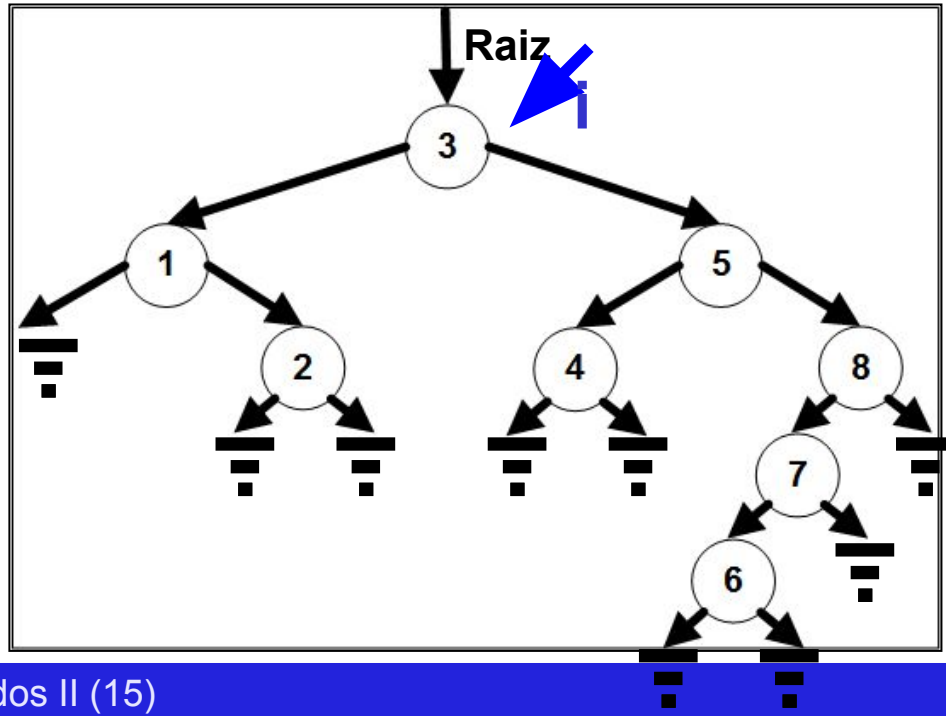


Algoritmo de Pesquisa em Java

```
//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;
}
```

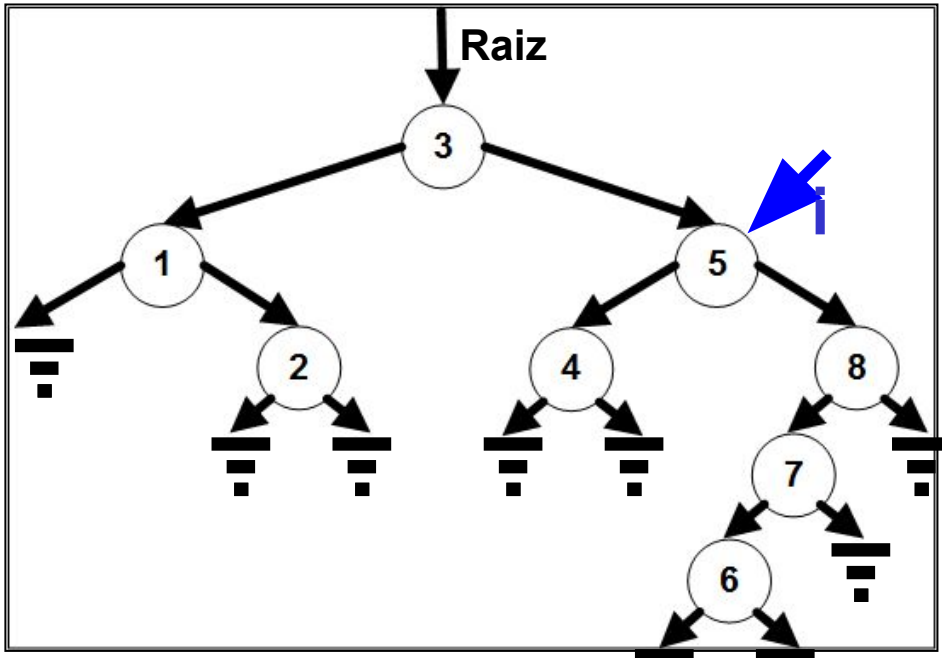


Algoritmo de Pesquisa em Java

```
//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;
}
```

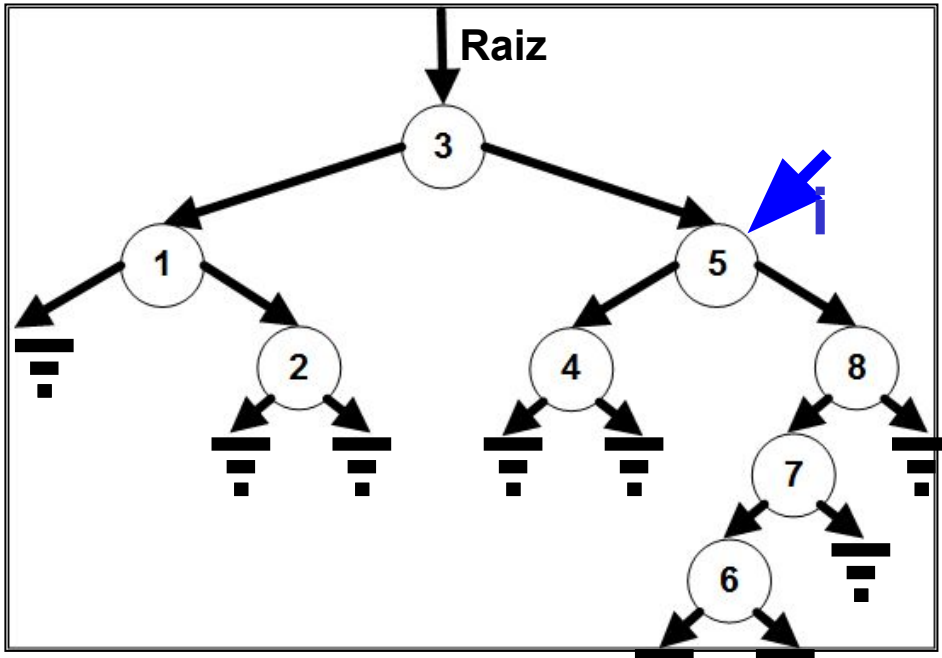


Algoritmo de Pesquisa em Java

```
//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;
}
```



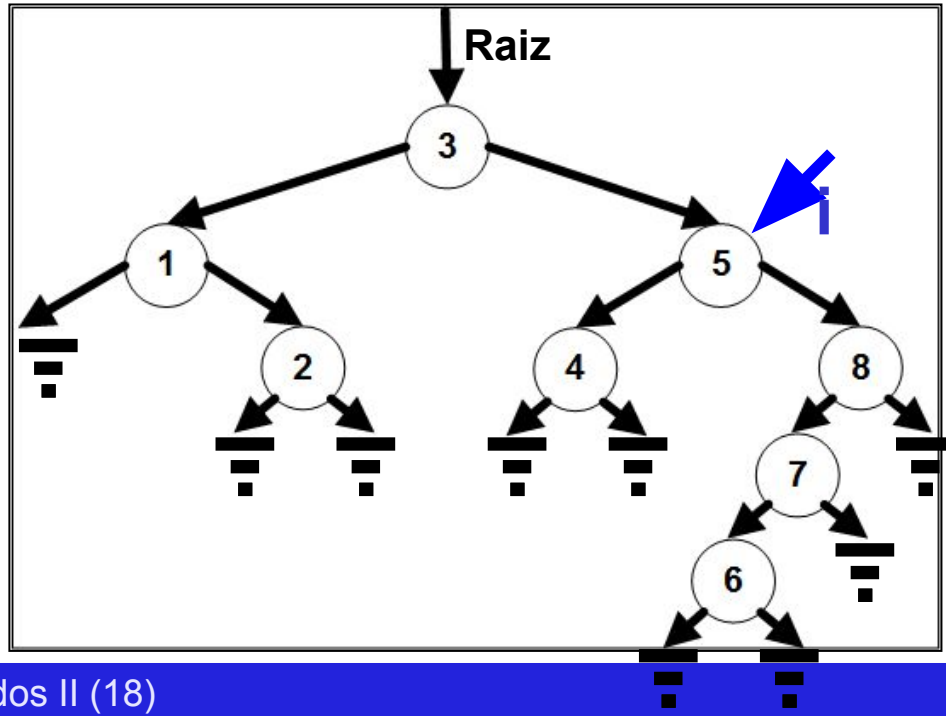
Algoritmo de Pesquisa em Java

```
//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



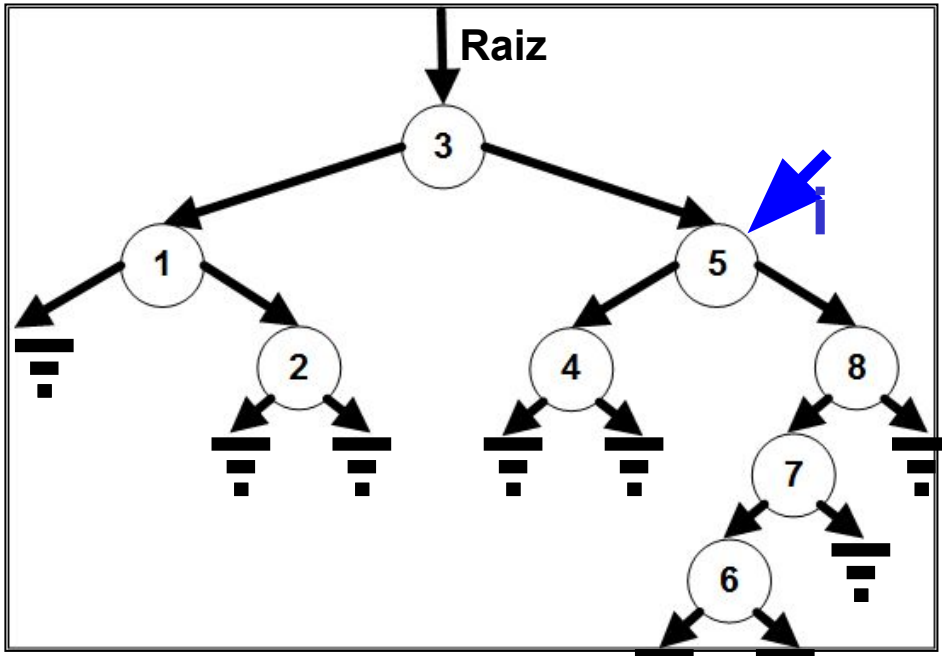
Algoritmo de Pesquisa em Java

```
//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



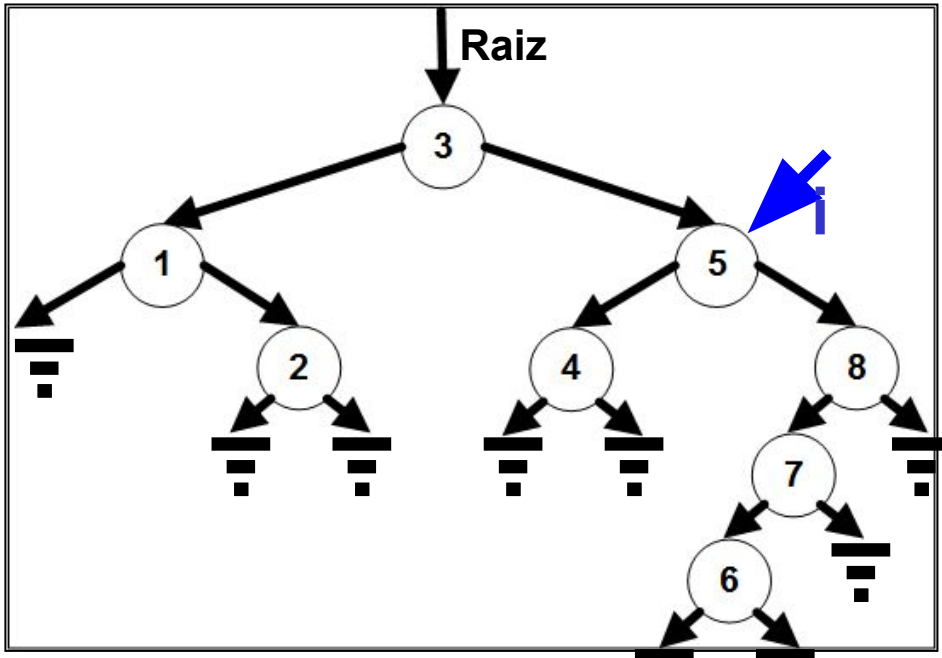
Algoritmo de Pesquisa em Java

```
//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

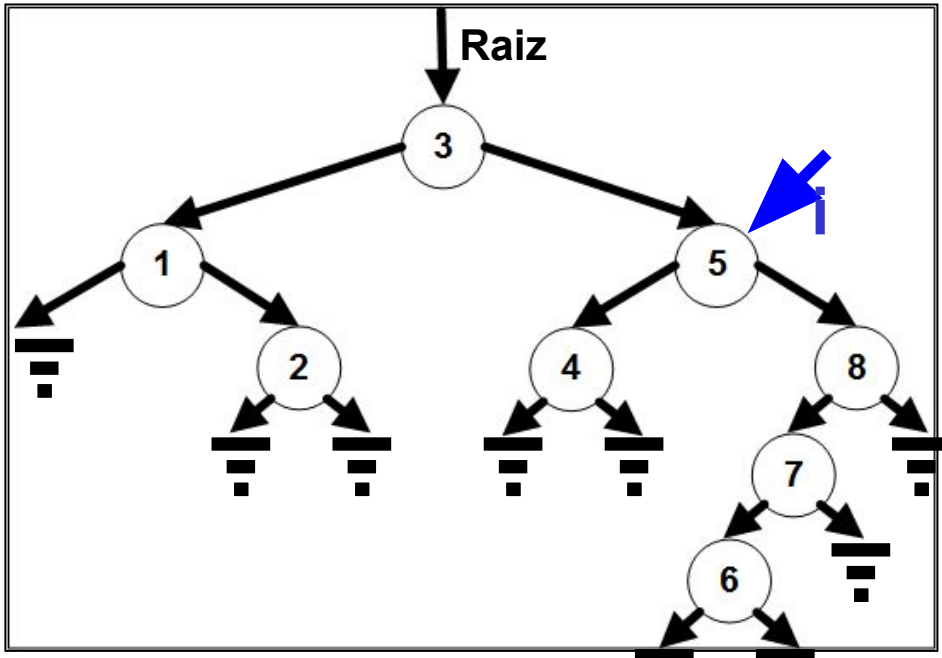


Algoritmo de Pesquisa em Java

```
//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;
}
```

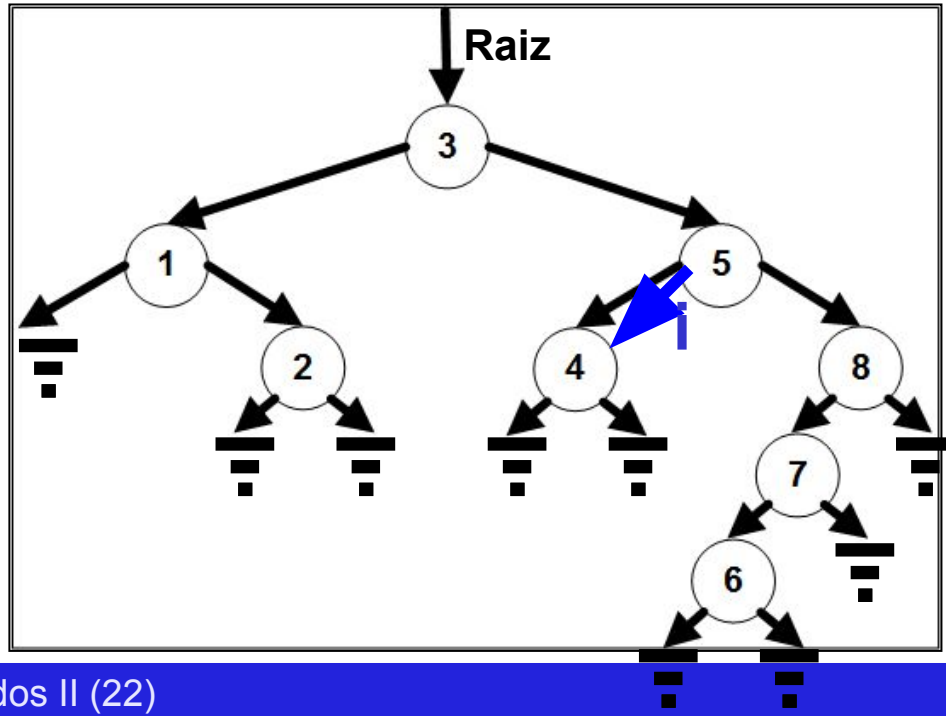


Algoritmo de Pesquisa em Java

```
//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;
}
```

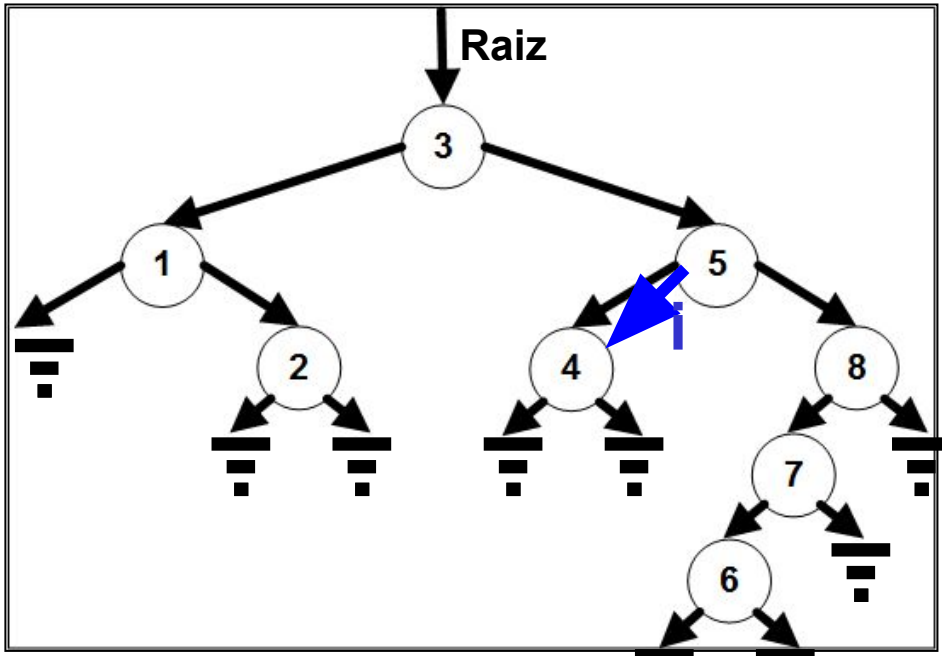


Algoritmo de Pesquisa em Java

```
//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;
}
```



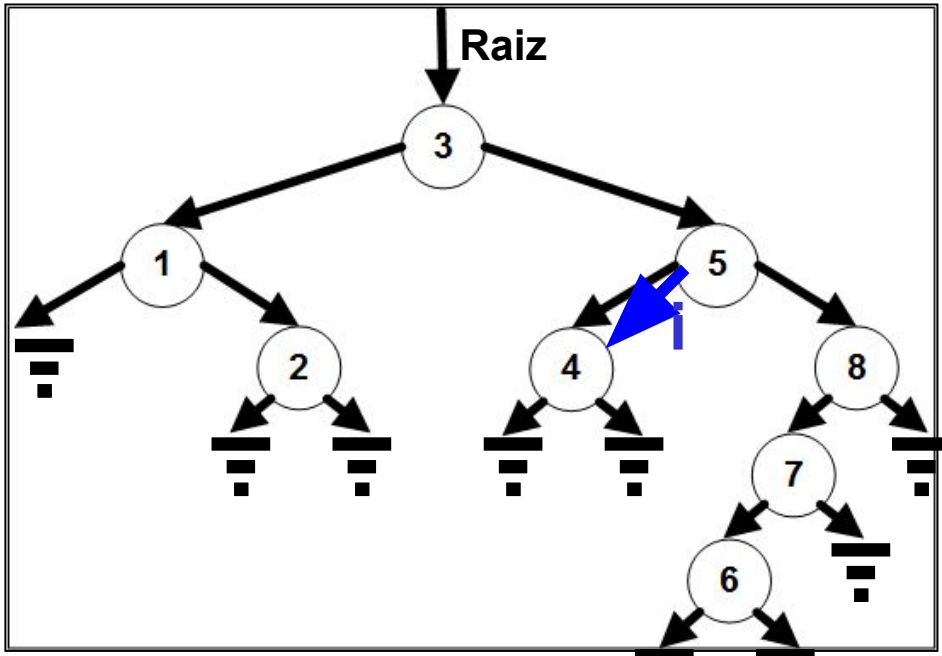
Algoritmo de Pesquisa em Java

```
//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



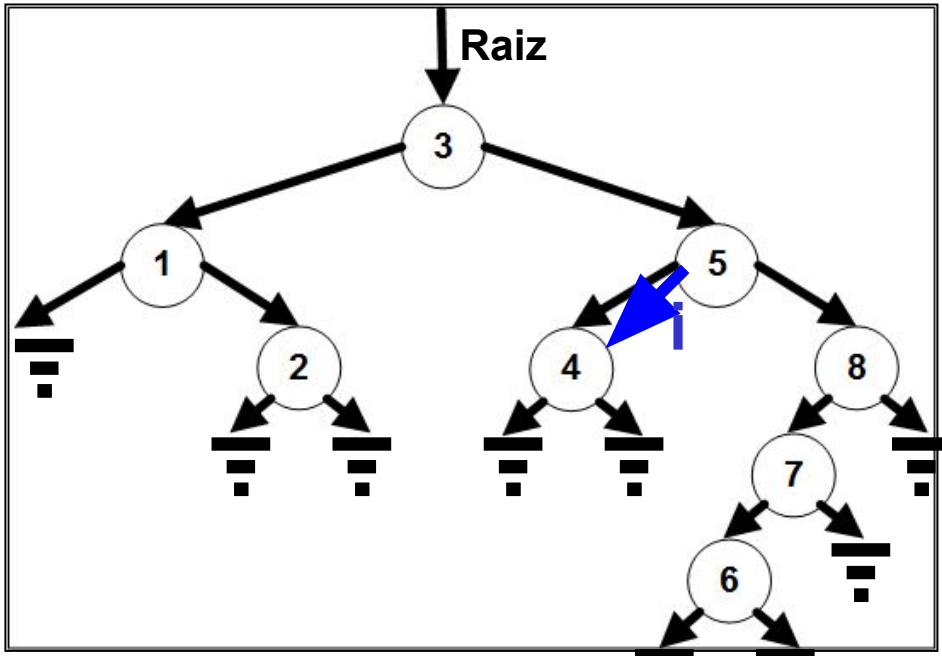
Algoritmo de Pesquisa em Java

```
//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

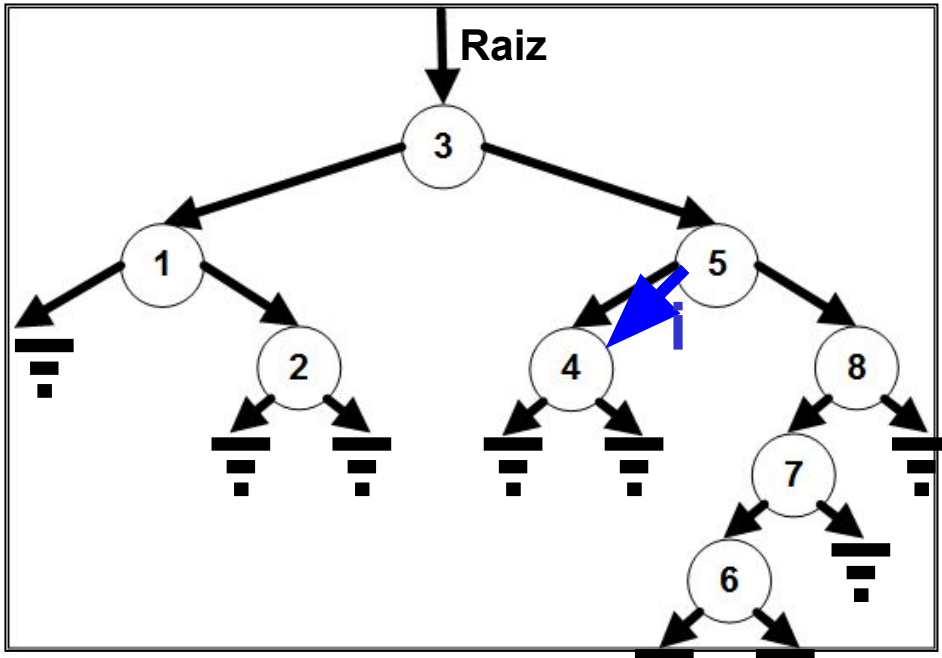


Algoritmo de Pesquisa em Java

```
//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;
}
```

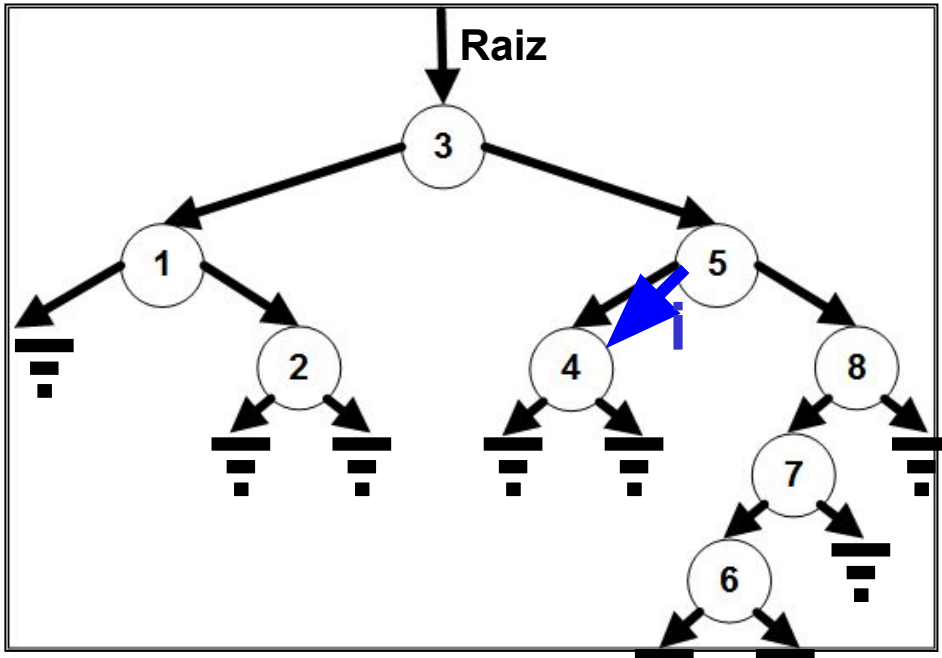


Algoritmo de Pesquisa em Java

```
//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;
}
```

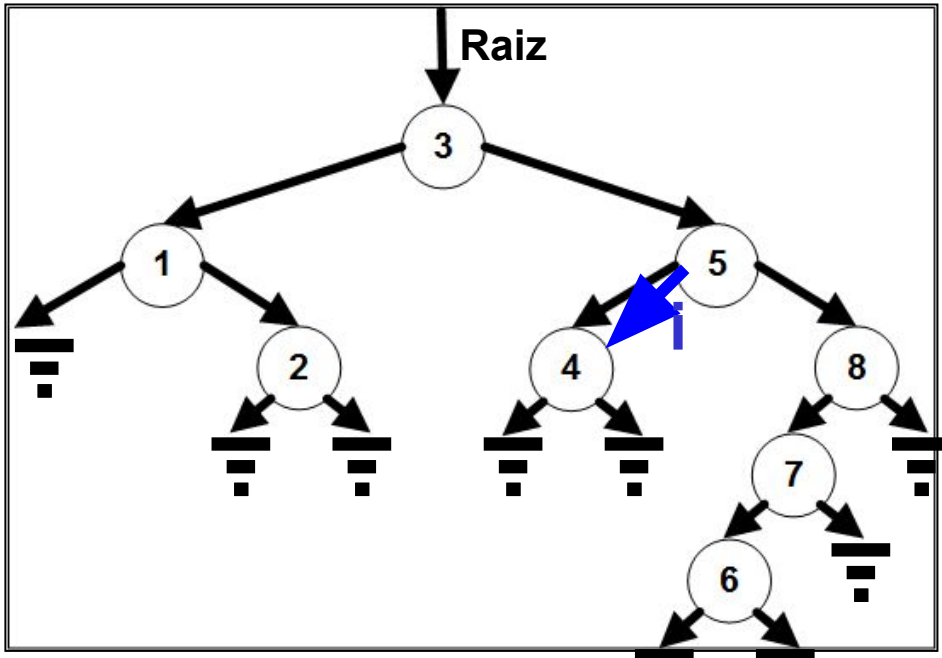


Algoritmo de Pesquisa em Java

```
//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;
}
```

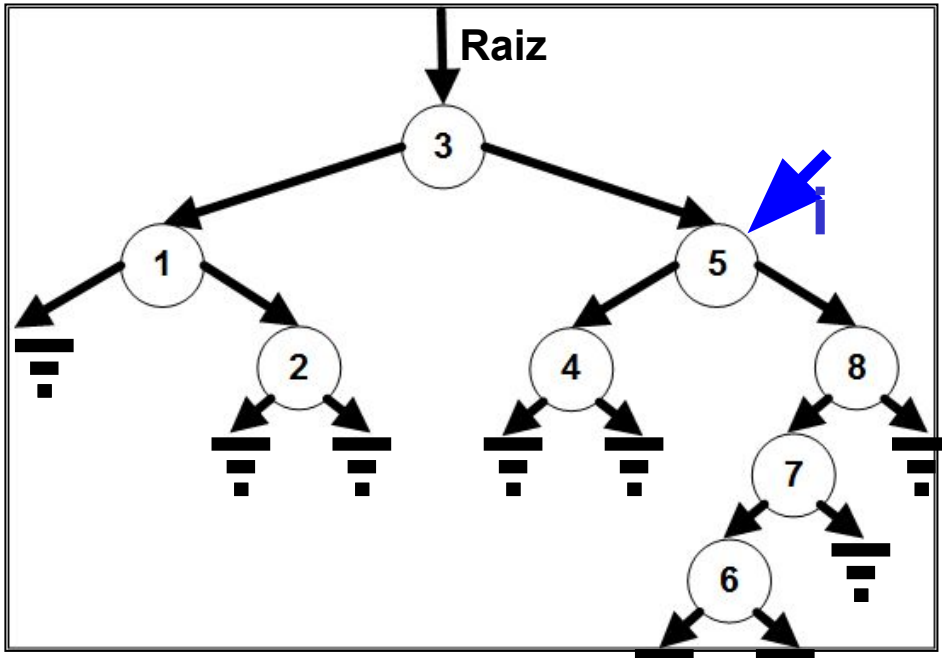


Algoritmo de Pesquisa em Java

```
//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;
}
```

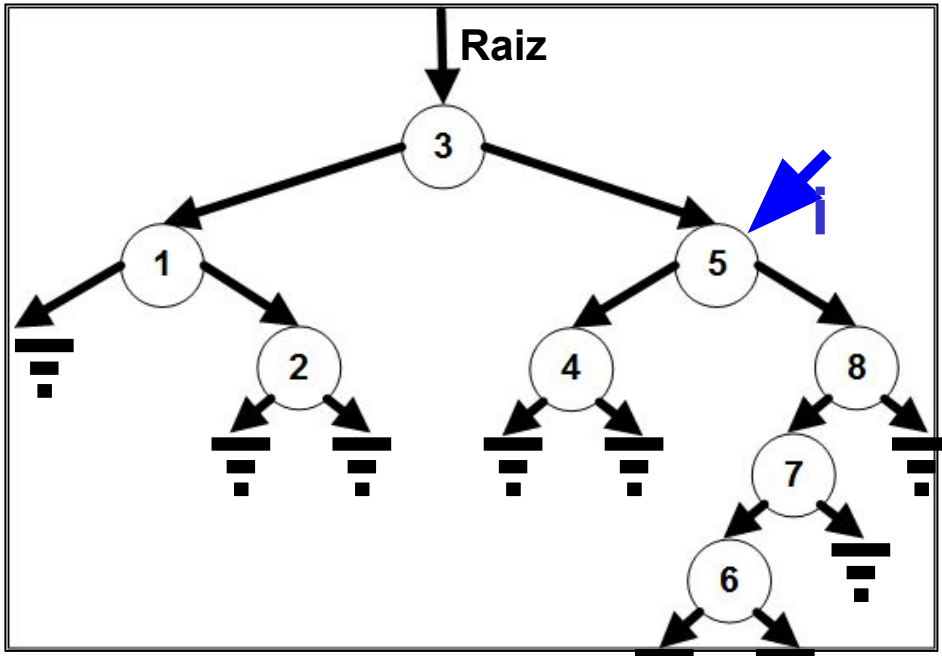


Algoritmo de Pesquisa em Java

```
//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;
}
```

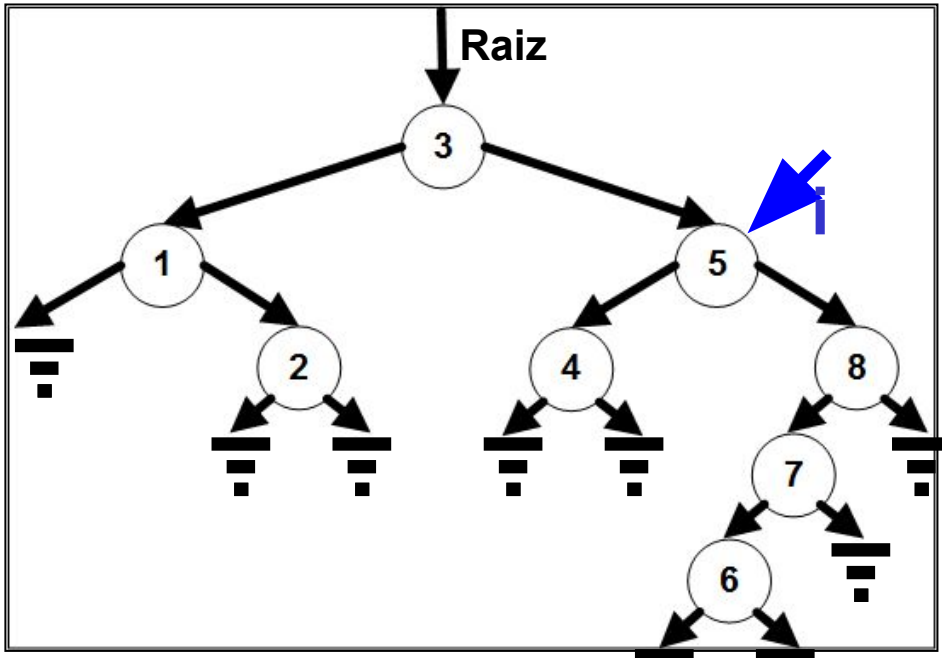


Algoritmo de Pesquisa em Java

```
//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;
}
```

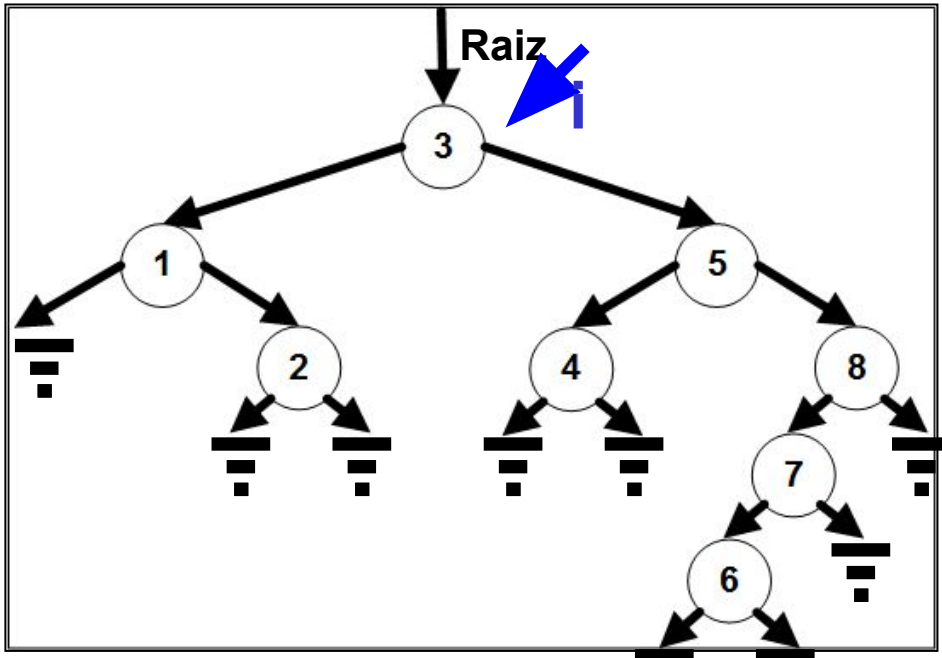


Algoritmo de Pesquisa em Java

```
//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;
}
```

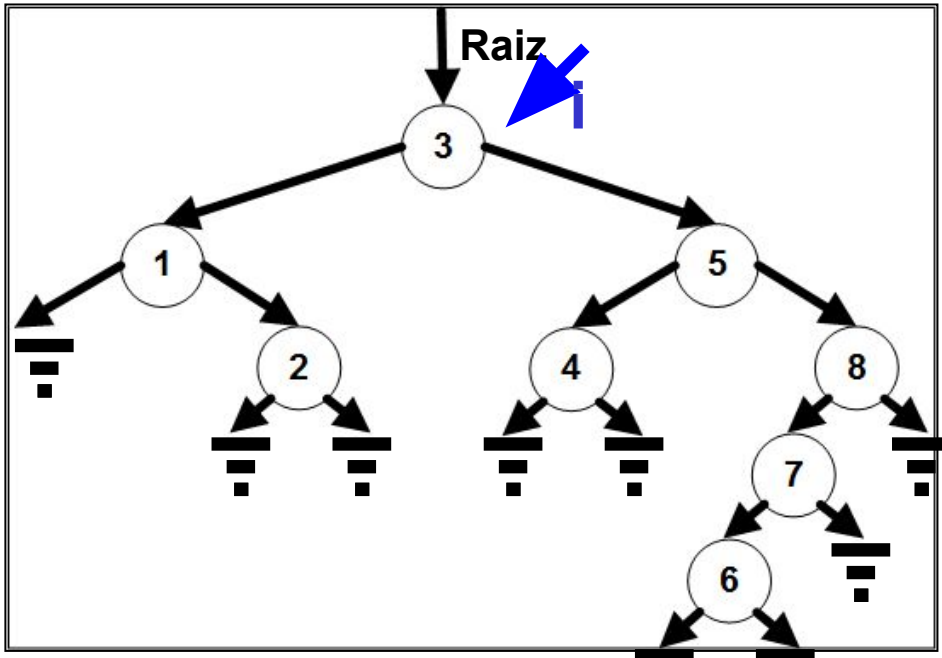


Algoritmo de Pesquisa em Java

```
//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;
}
```

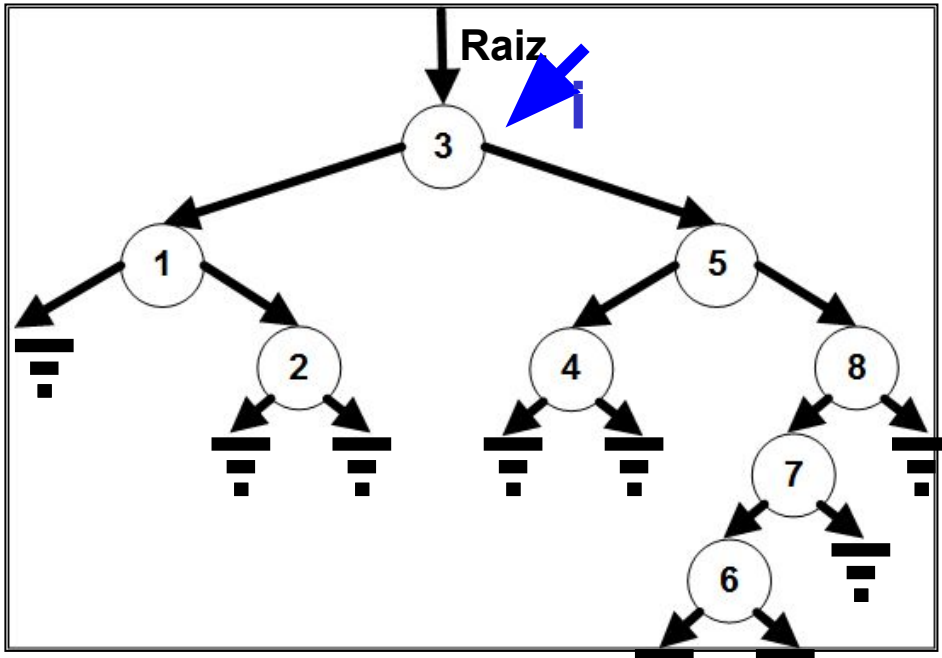


Algoritmo de Pesquisa em Java

```
//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;
}
```



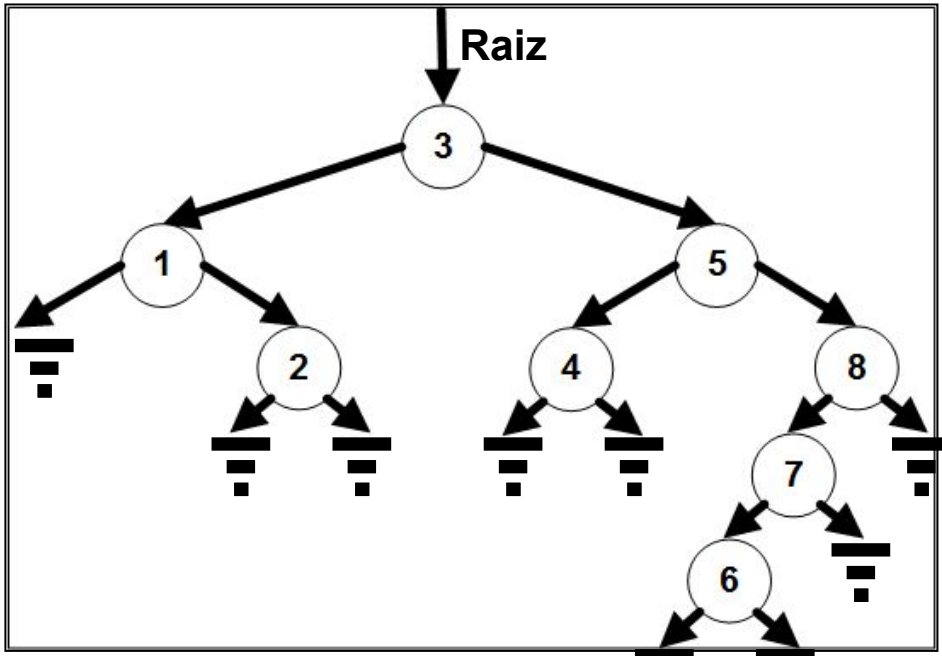
Algoritmo de Pesquisa em Java

```
//Pesquisar(4)

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

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;
}
```

retorna true

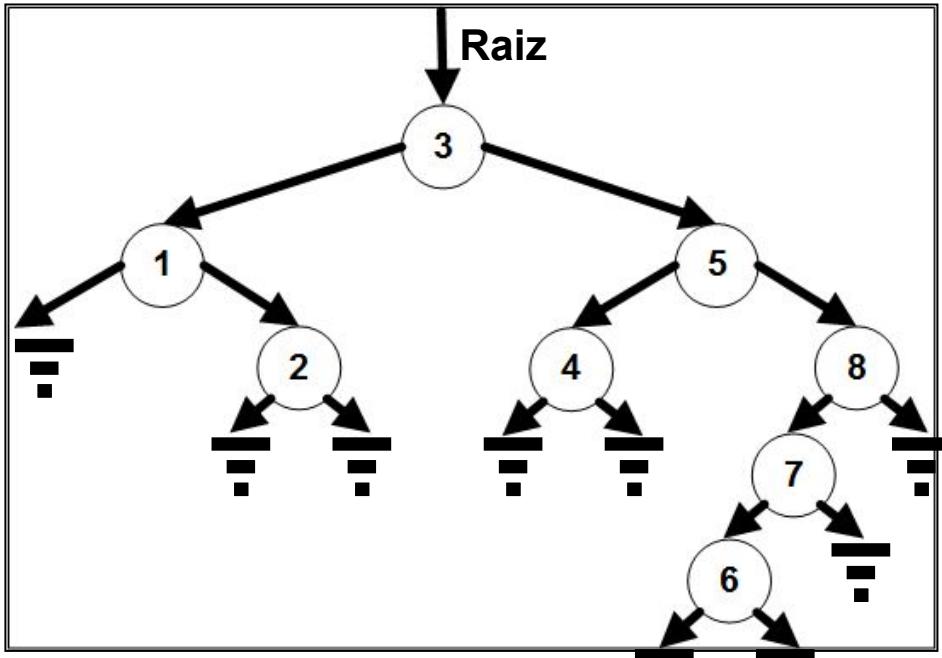


Algoritmo de Pesquisa em Java

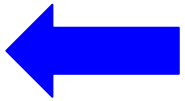
```
//Pesquisar(4)

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;
}
```



Agenda

- Funcionamento básico da Pesquisa
- Algoritmo pesquisa em Java
- **Análise de complexidade da Pesquisa** 
- Caminhamento

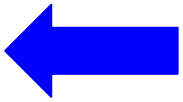
Análise de complexidade da Pesquisa

- **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

Agenda

- Funcionamento básico da Pesquisa
- Algoritmo pesquisa em Java
- Análise de complexidade da Pesquisa
- **Caminhamento** 

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

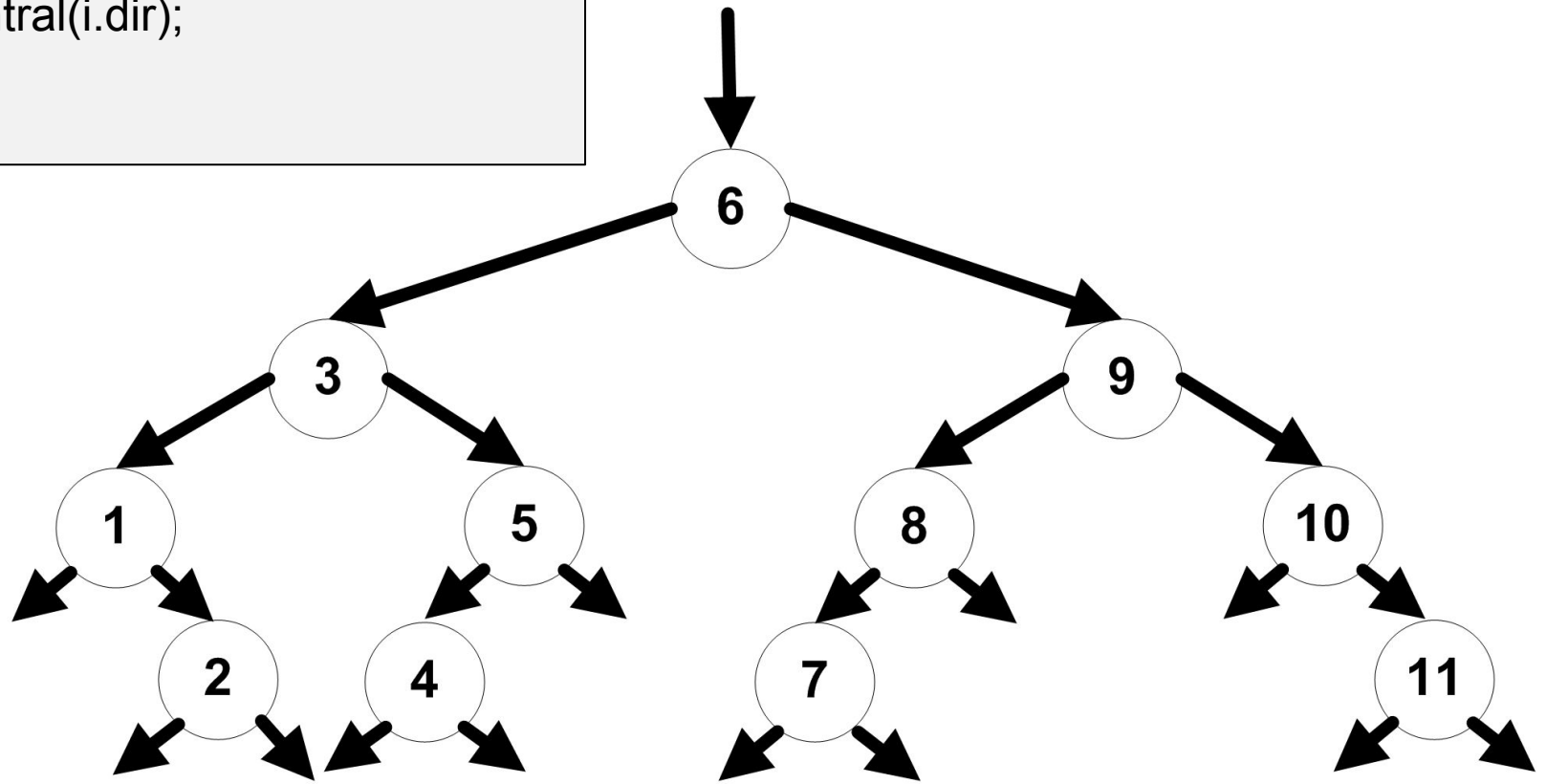
```
void caminharCentral(No i) {
  if (i != null) {
    caminharCentral(i.esq);
    System.out.print(i.elemento + " ");
    caminharCentral(i.dir);
  }
}
                          //central ou em ordem
```

```
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

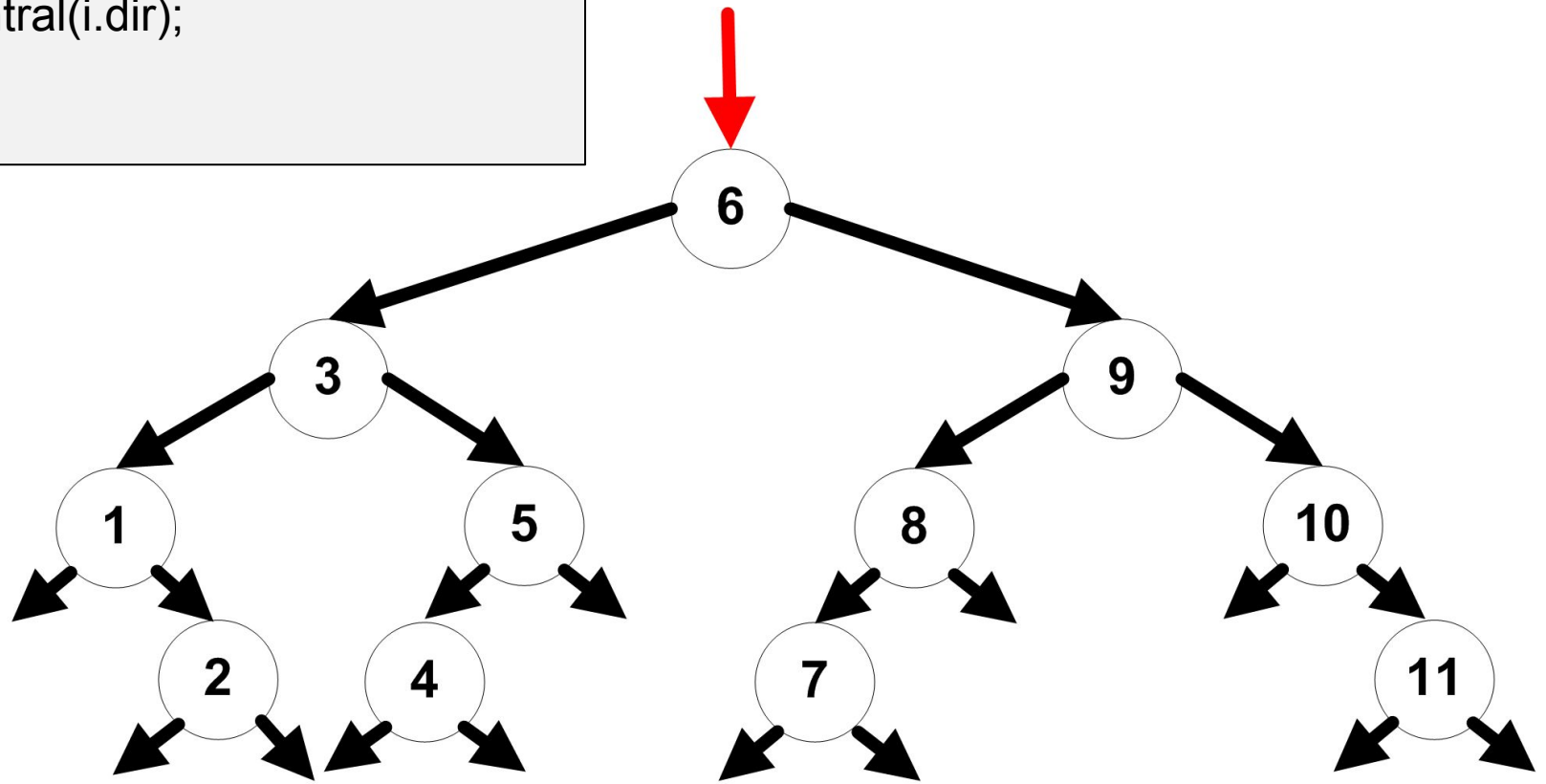
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	

Caminhamento Central ou Em Ordem

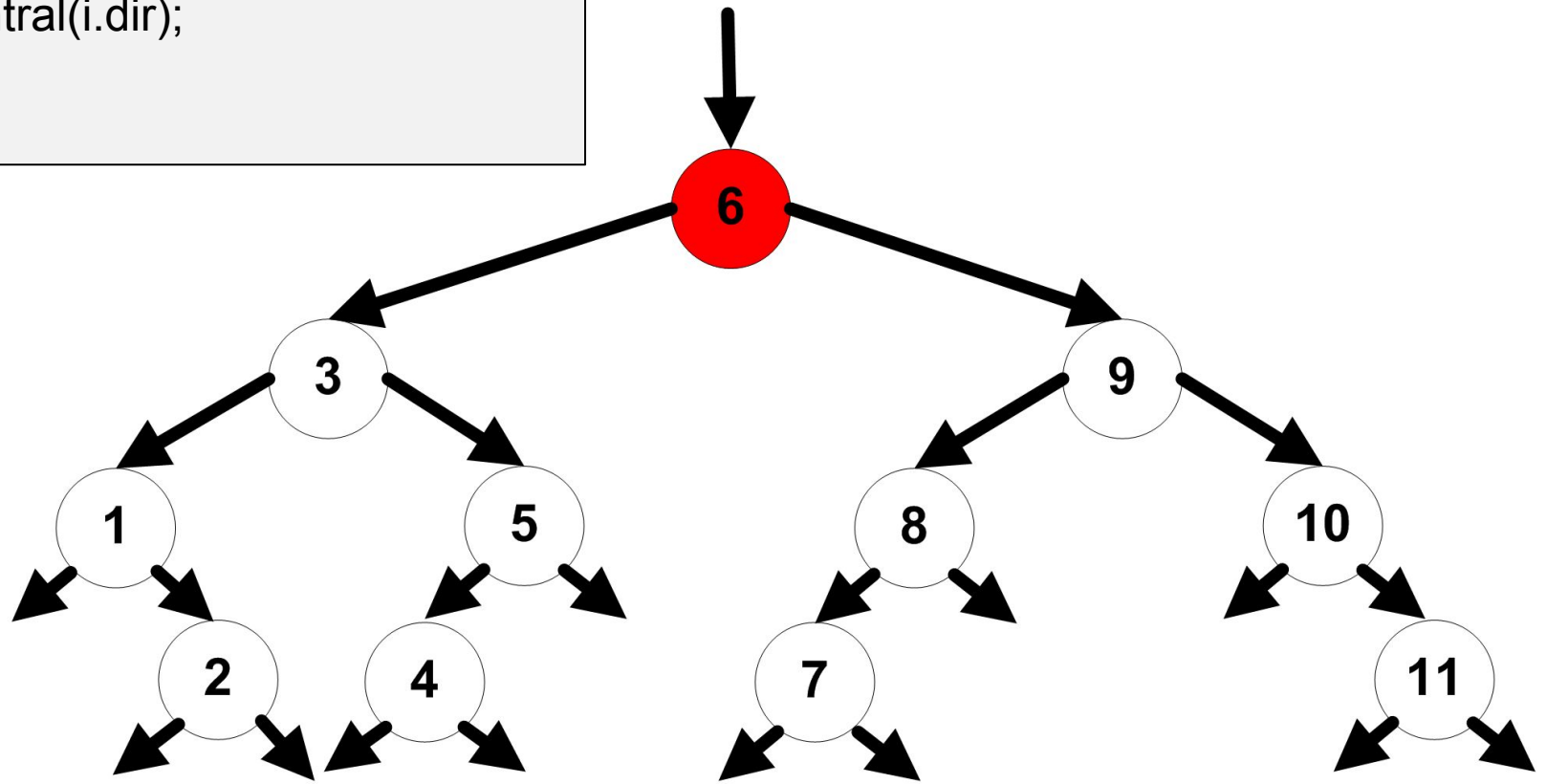
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	

Caminhamento Central ou Em Ordem

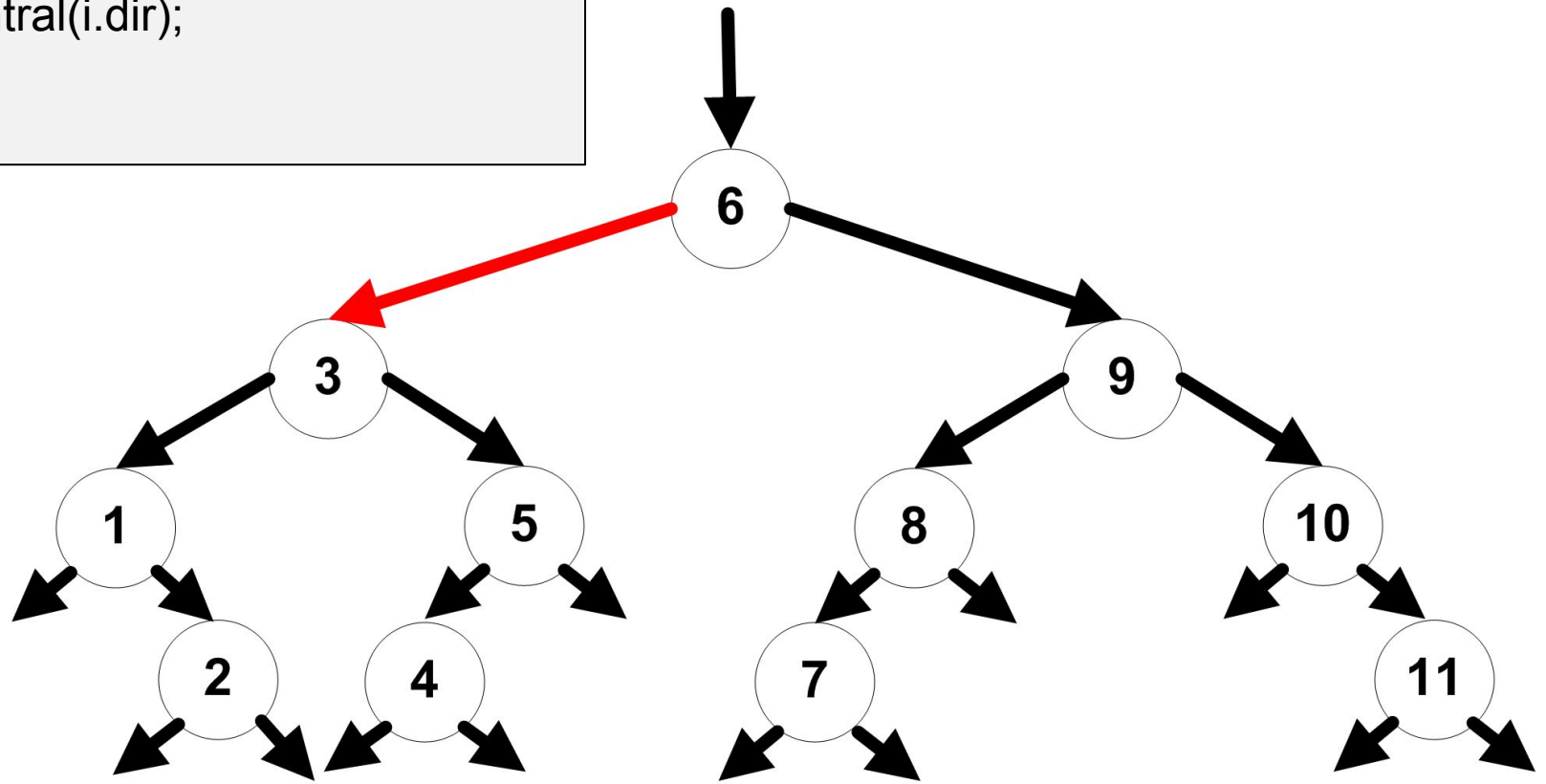
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	

Caminhamento Central ou Em Ordem

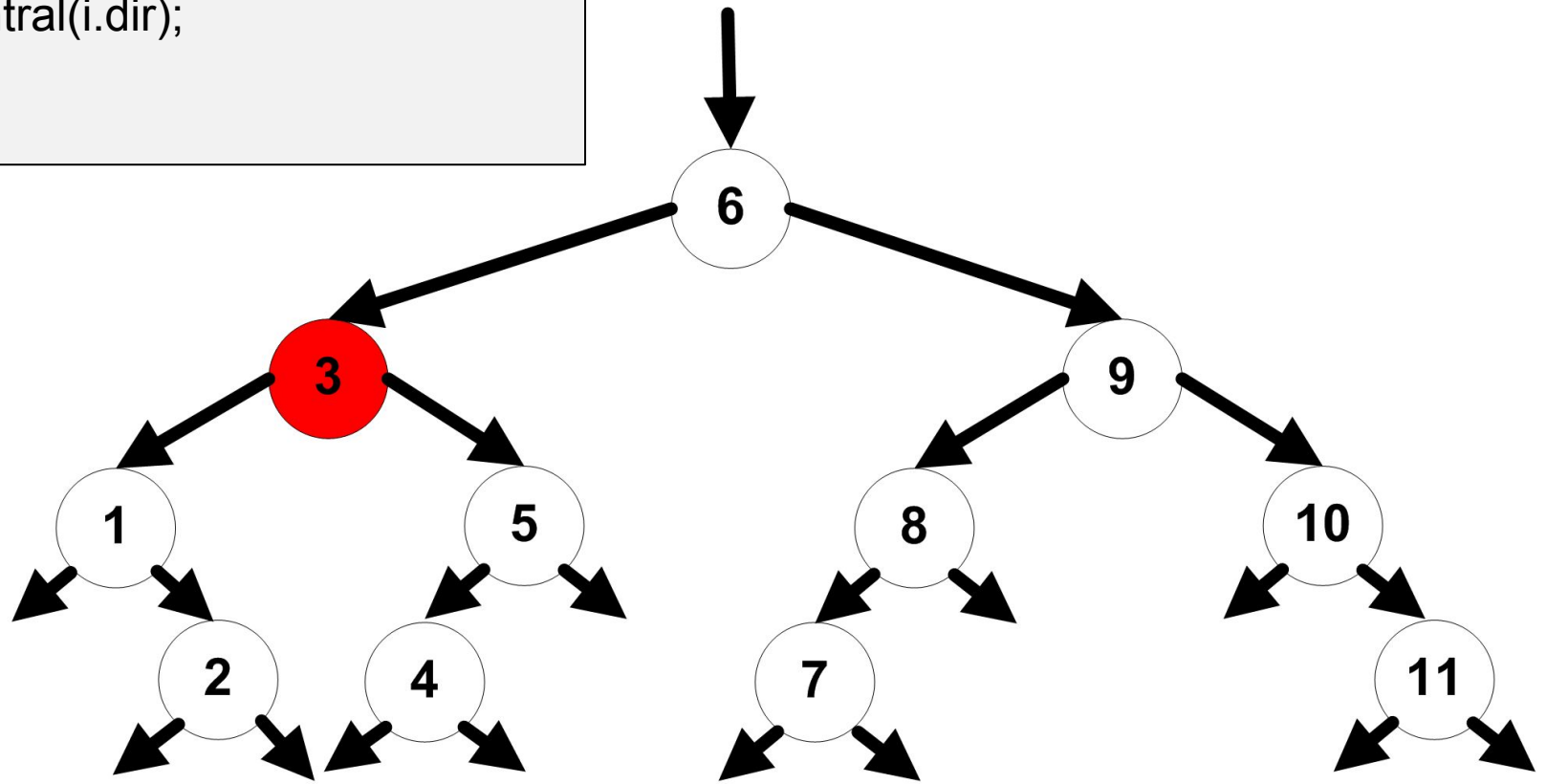
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	

Caminhamento Central ou Em Ordem

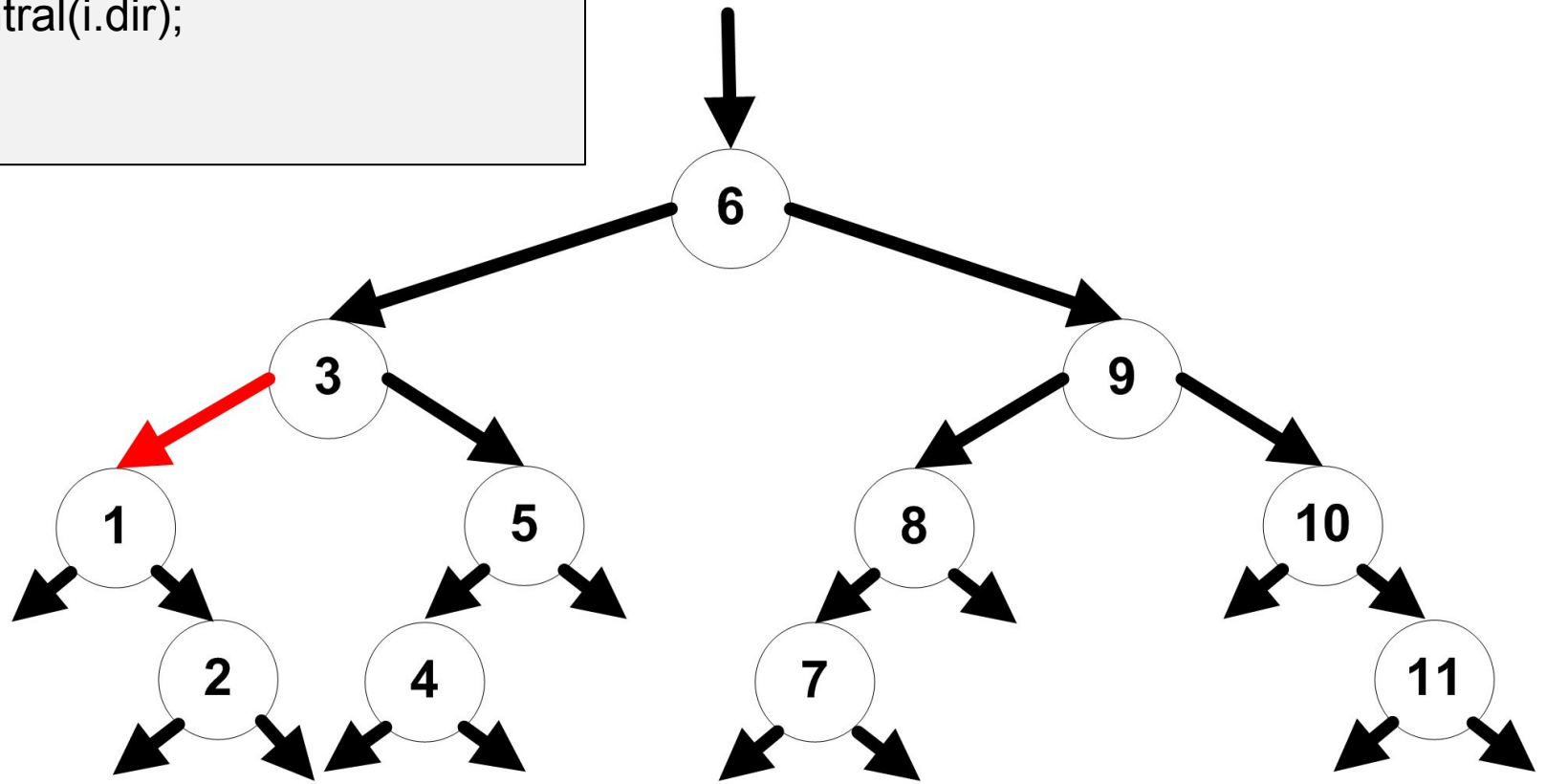
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	

Caminhamento Central ou Em Ordem

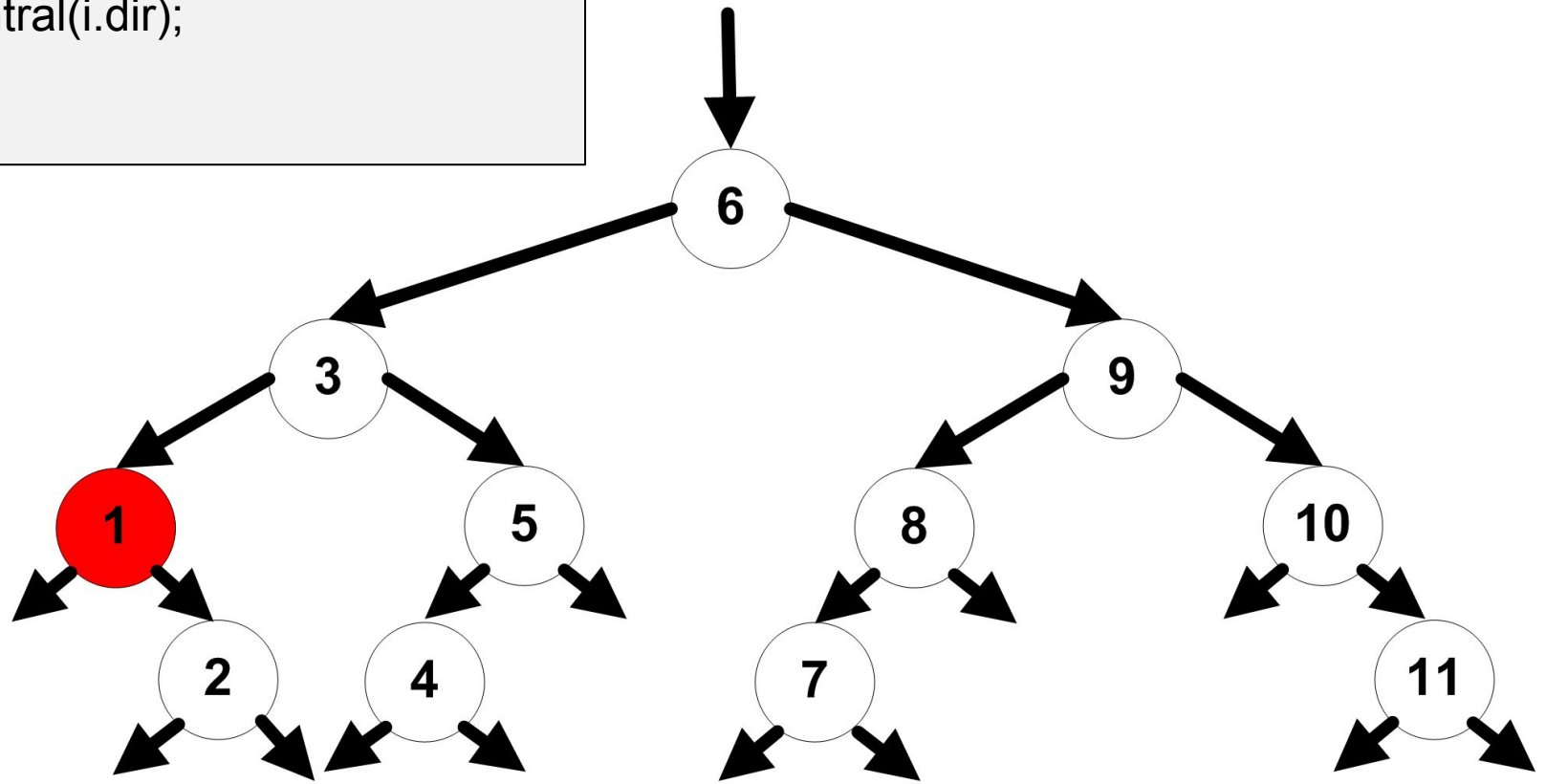
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	

Caminhamento Central ou Em Ordem

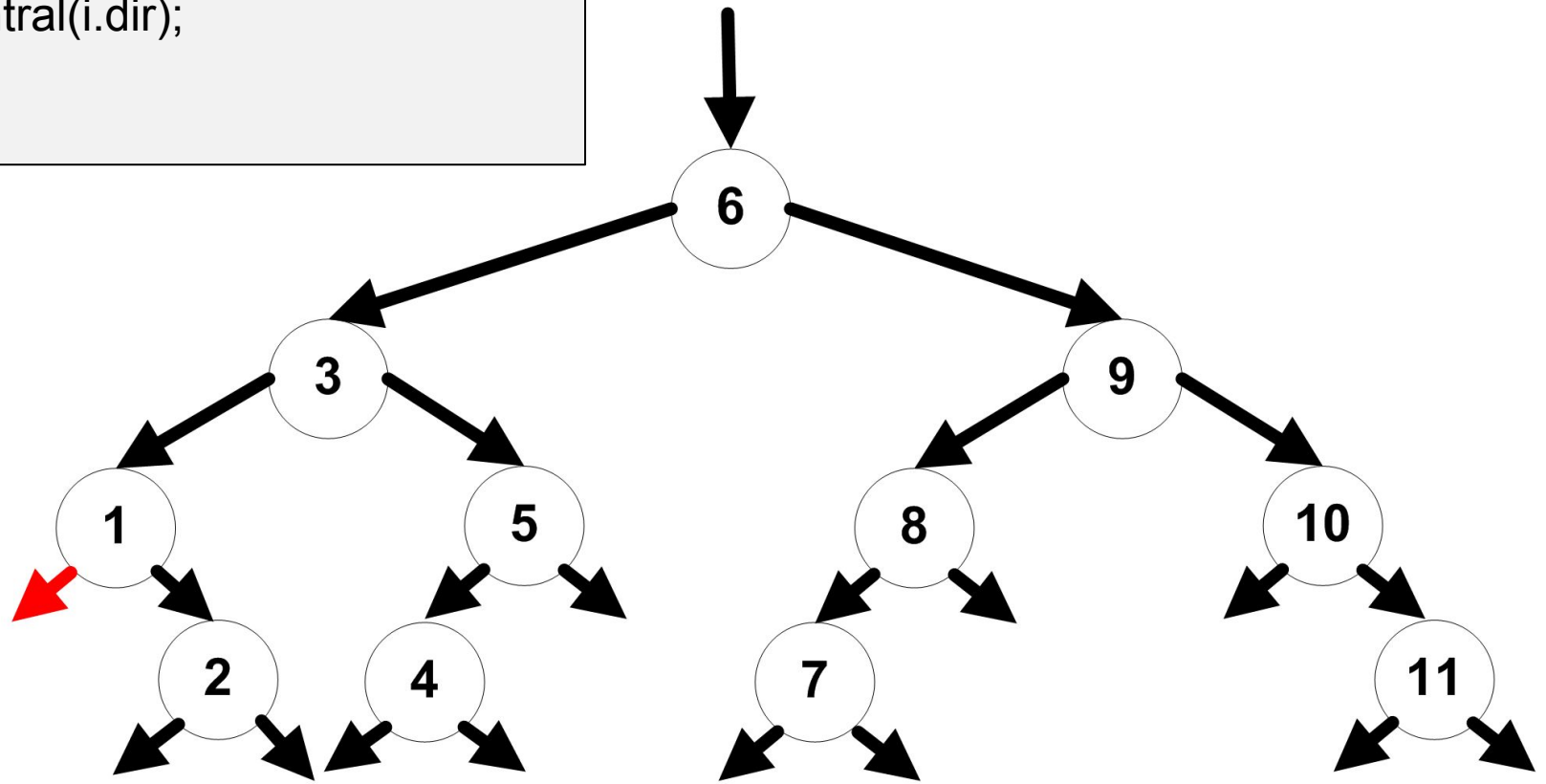
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	

Caminhamento Central ou Em Ordem

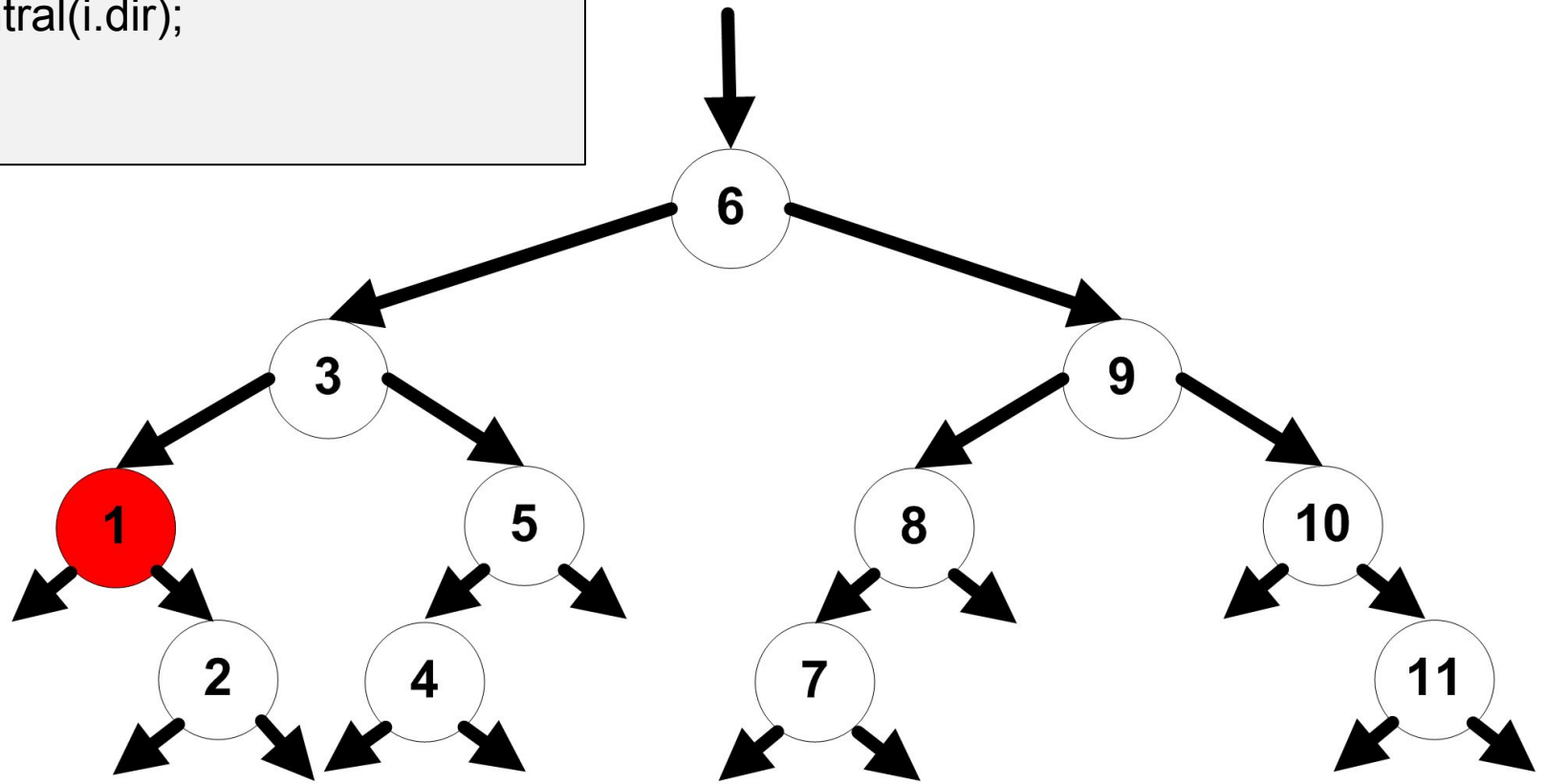
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	

Caminhamento Central ou Em Ordem

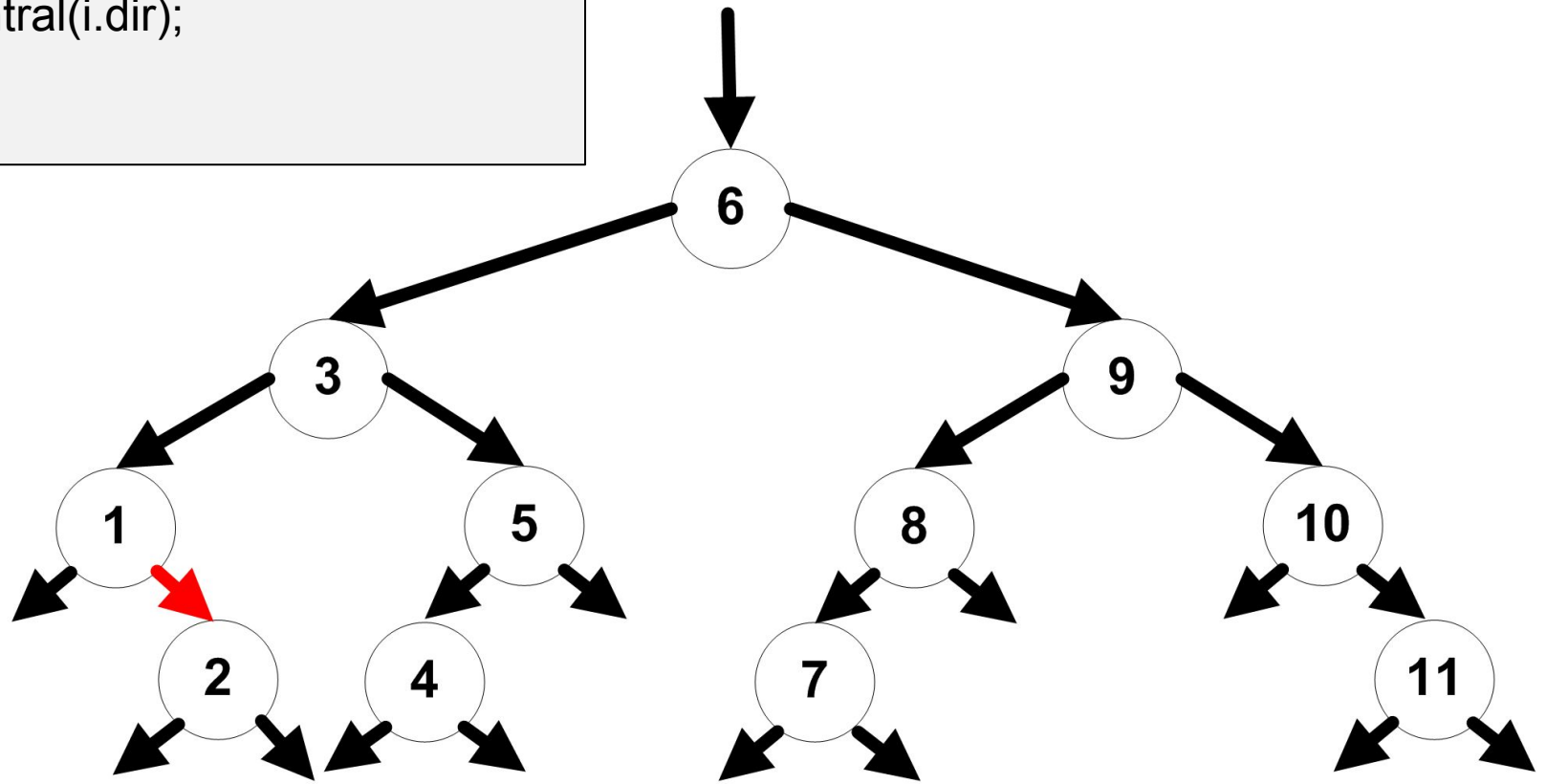
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1

Caminhamento Central ou Em Ordem

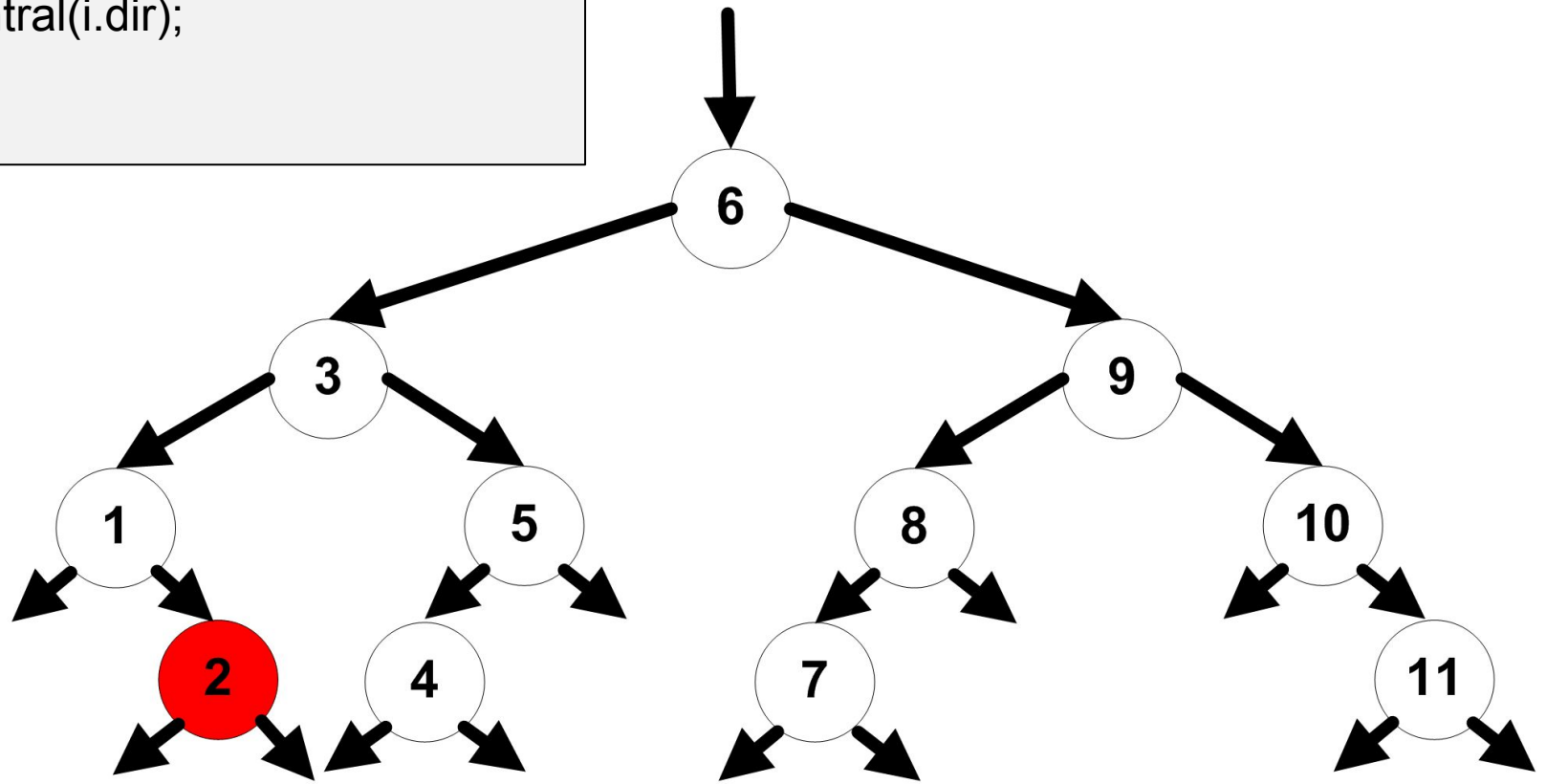
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1

Caminhamento Central ou Em Ordem

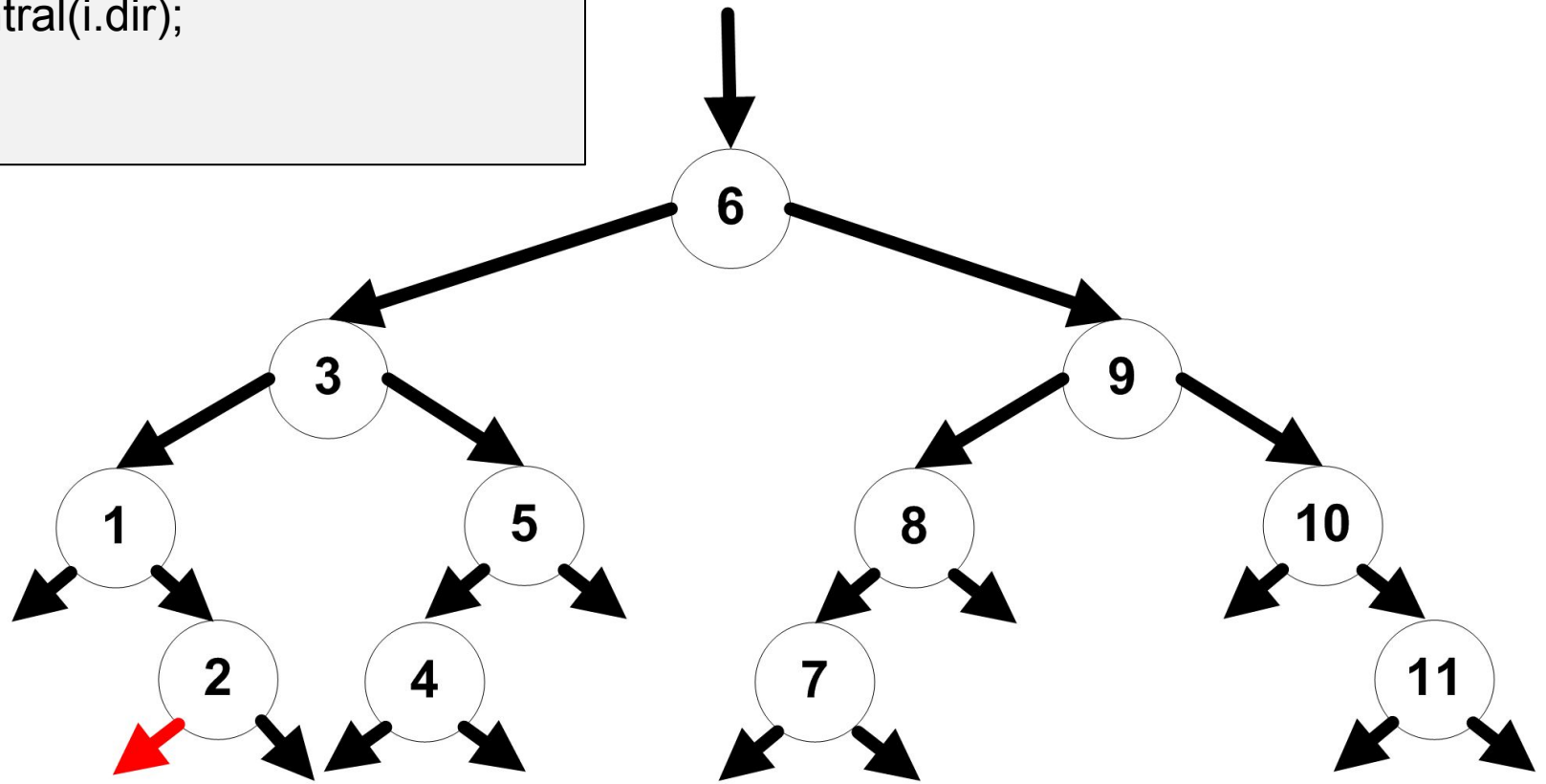
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1

Caminhamento Central ou Em Ordem

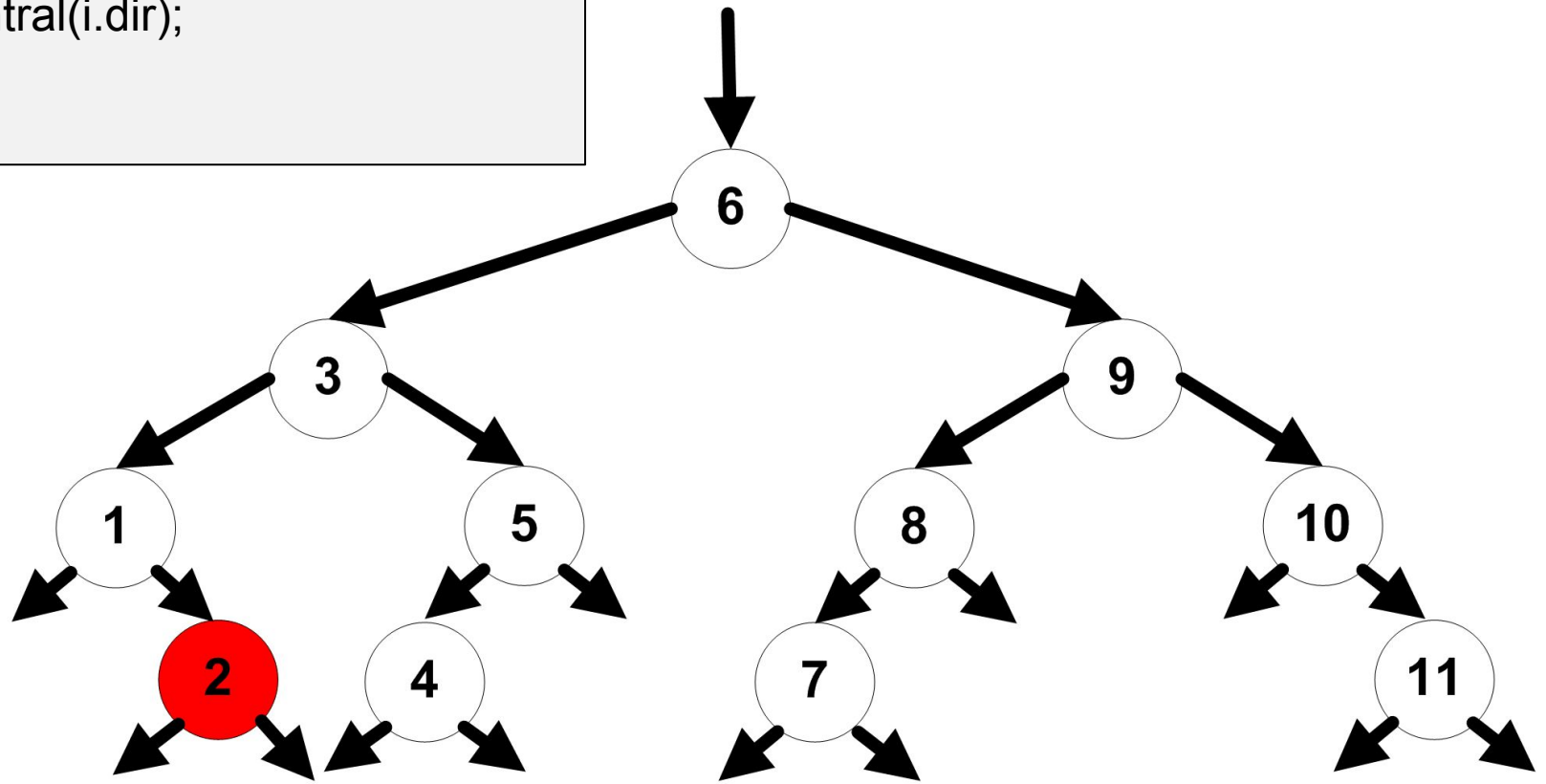
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1

Caminhamento Central ou Em Ordem

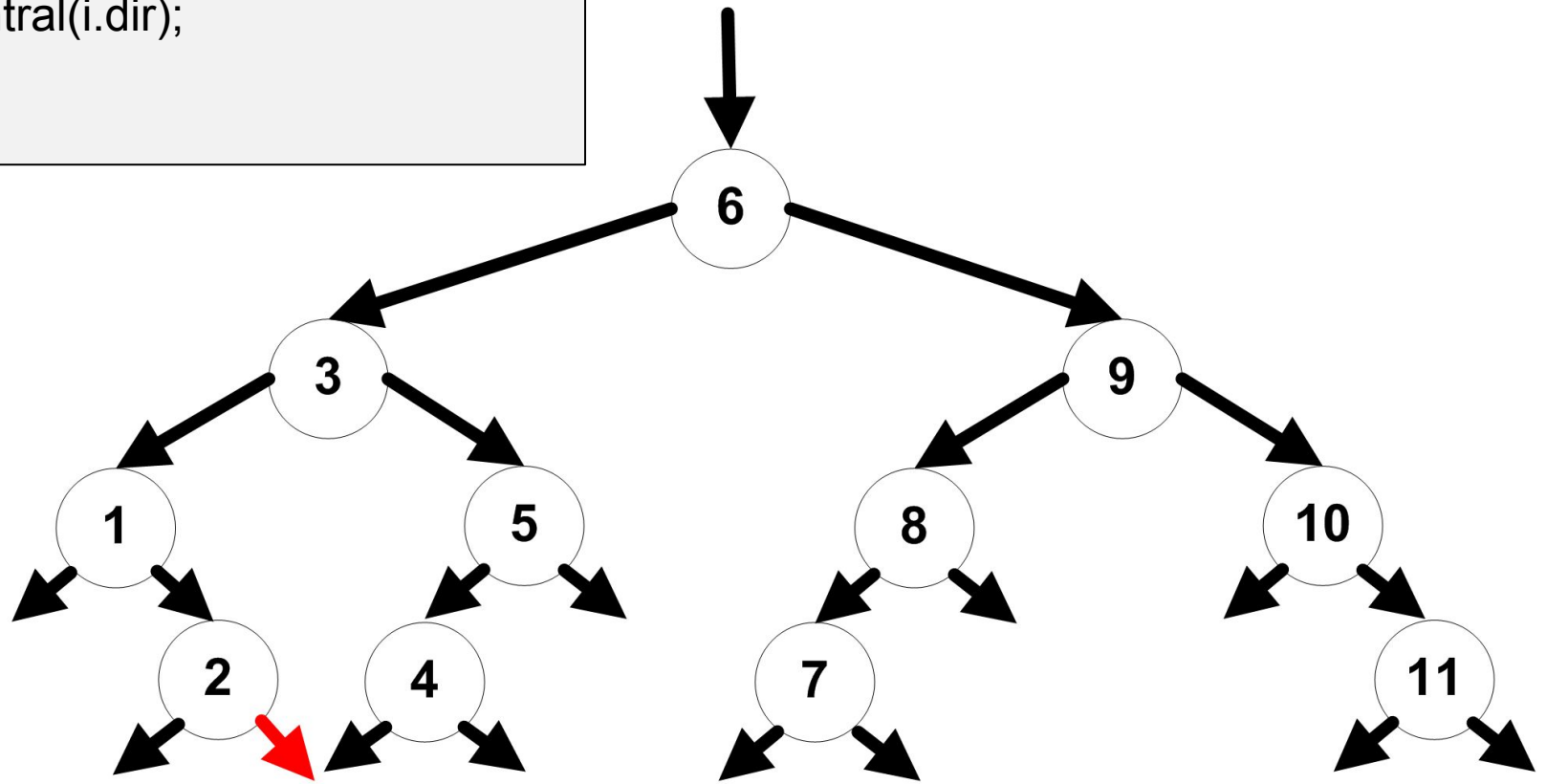
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2

Caminhamento Central ou Em Ordem

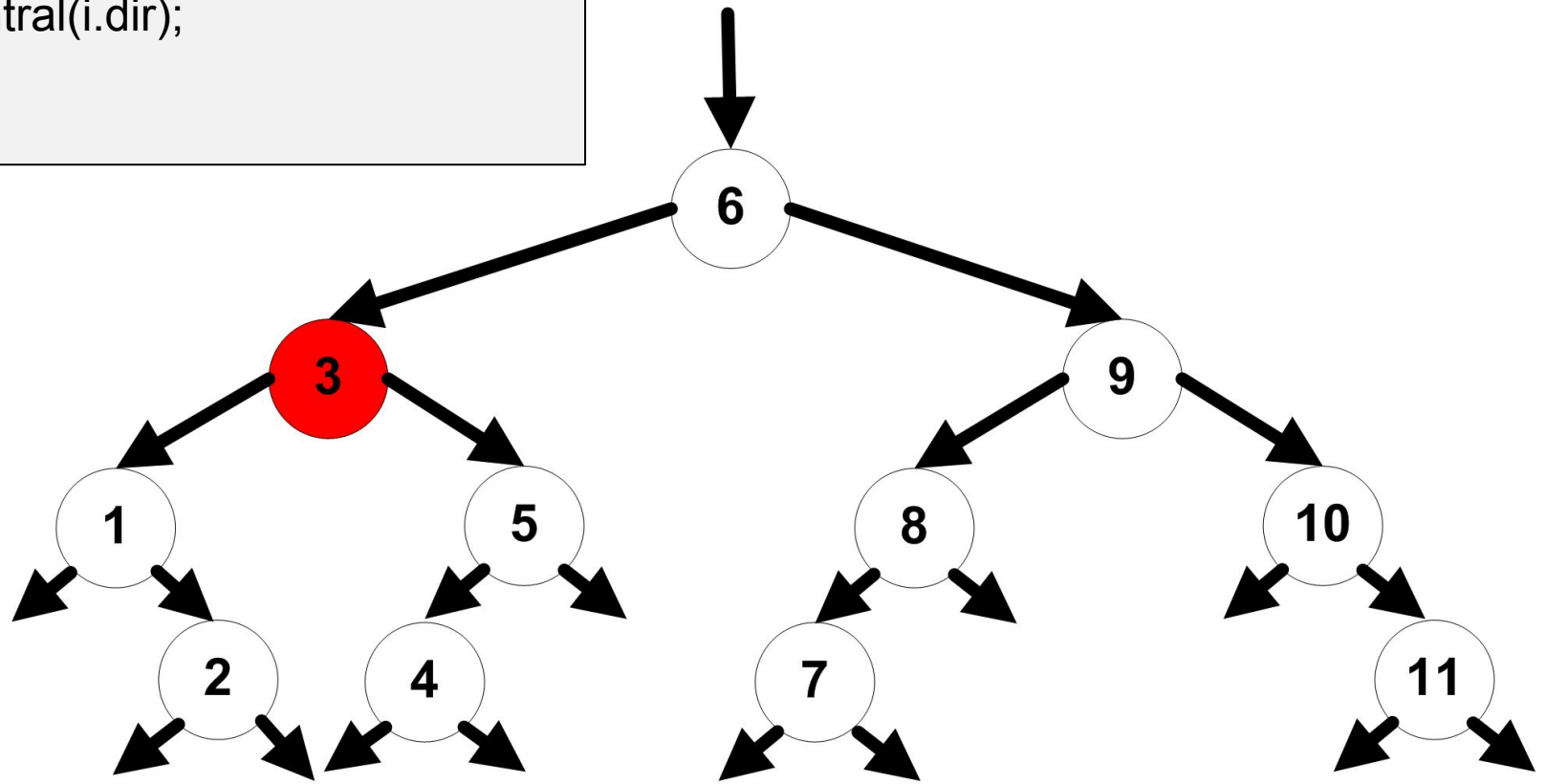
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2

Caminhamento Central ou Em Ordem

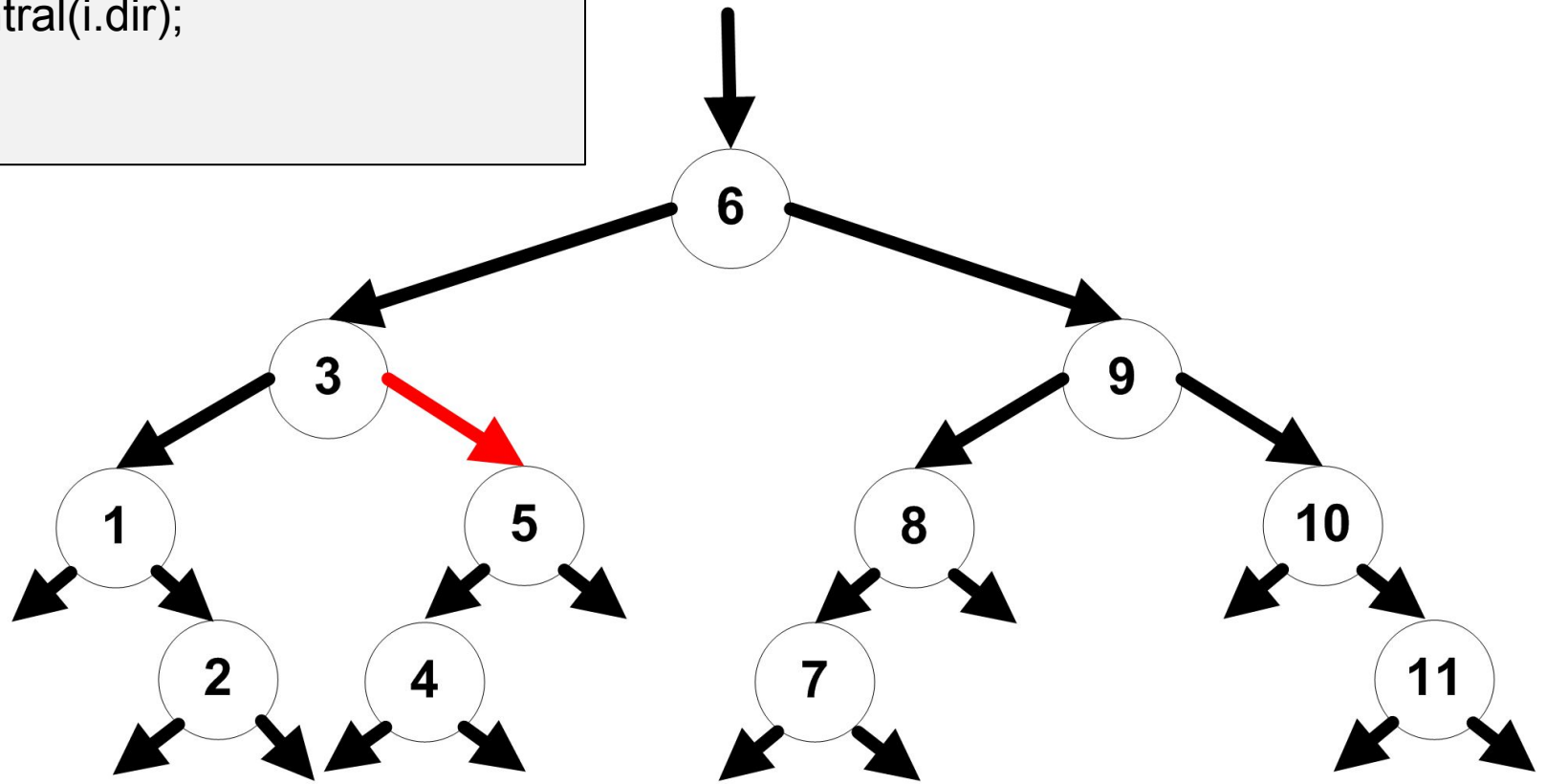
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3

Caminhamento Central ou Em Ordem

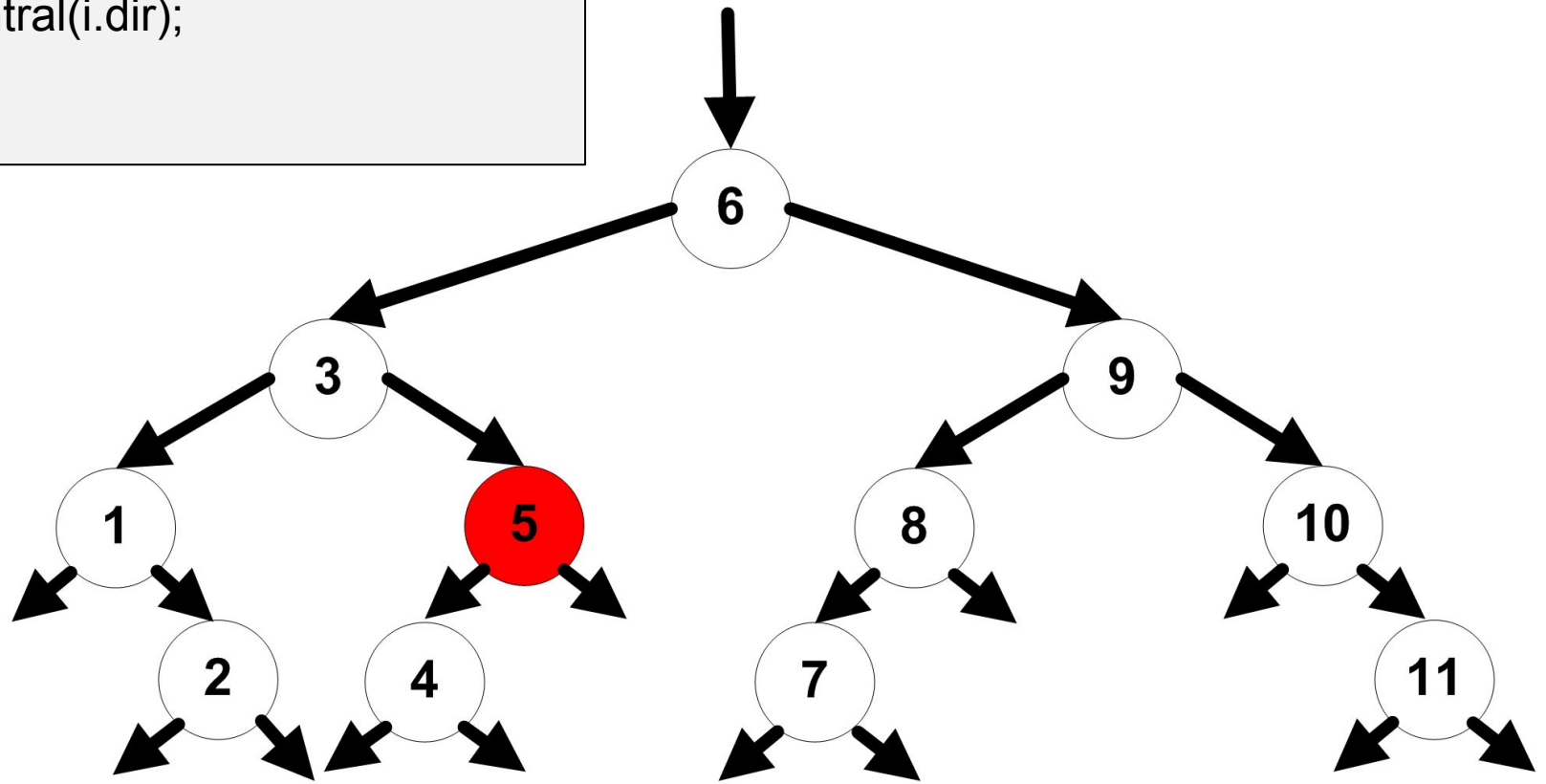
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3

Caminhamento Central ou Em Ordem

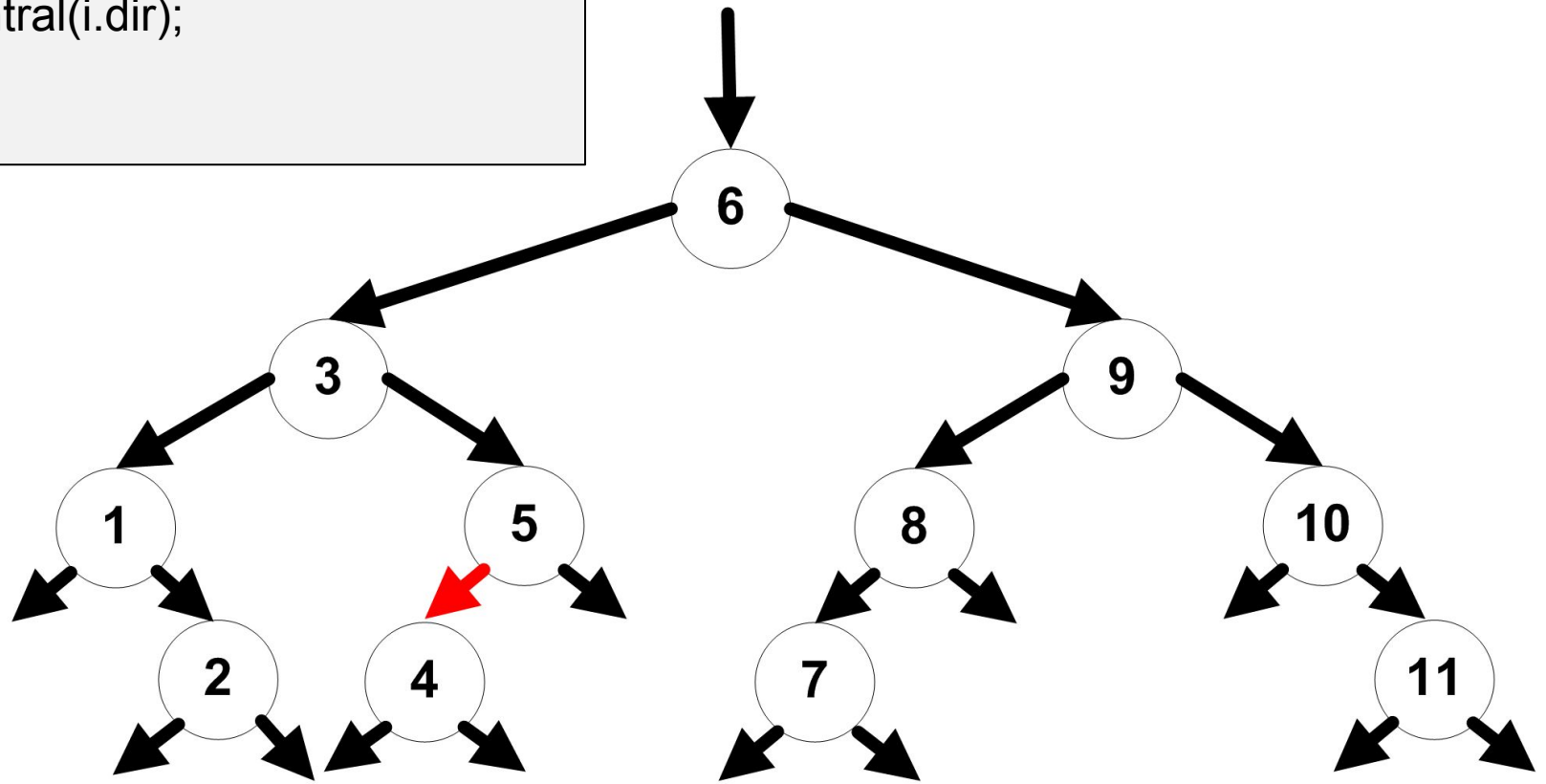
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3

Caminhamento Central ou Em Ordem

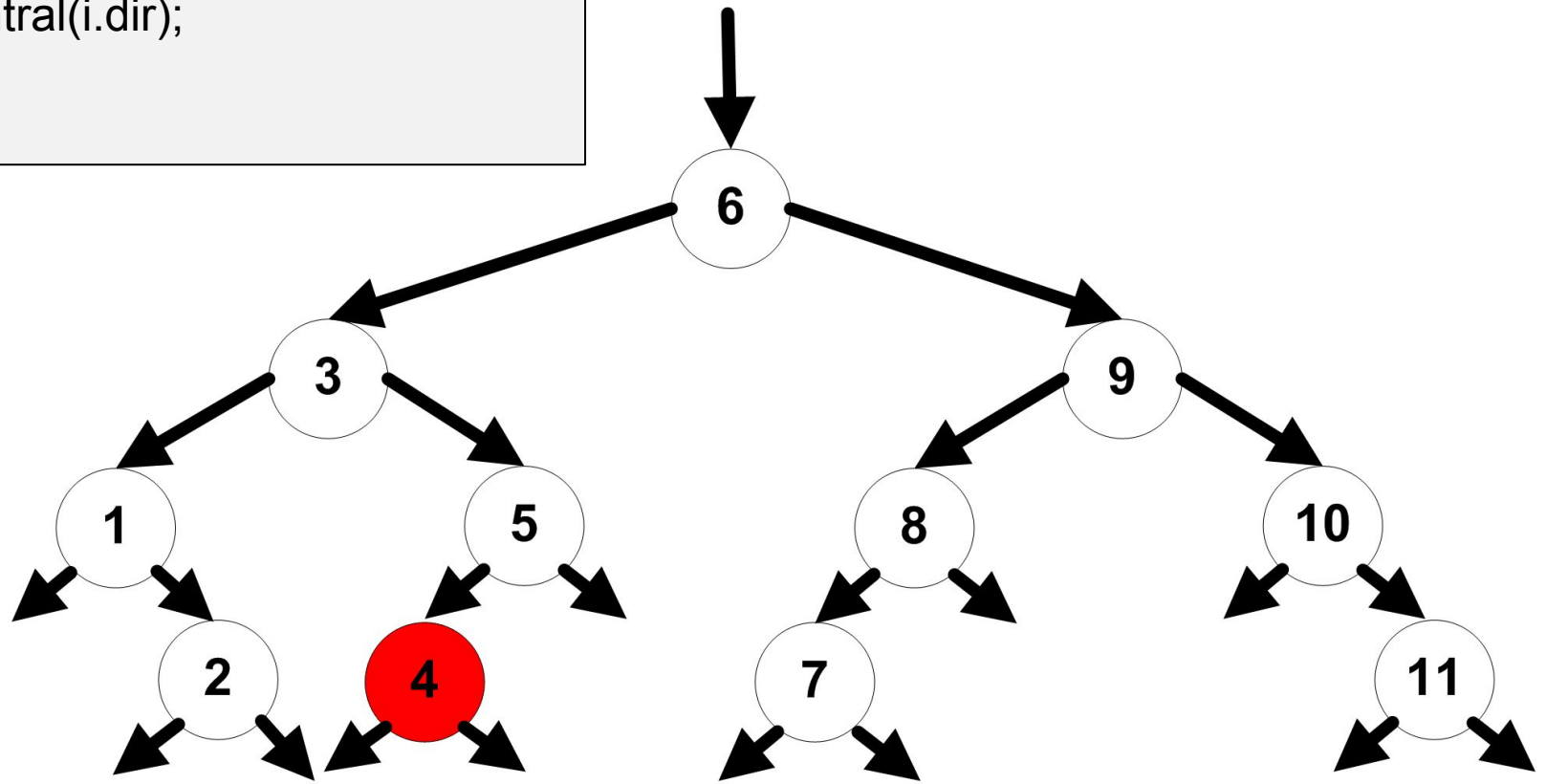
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3

Caminhamento Central ou Em Ordem

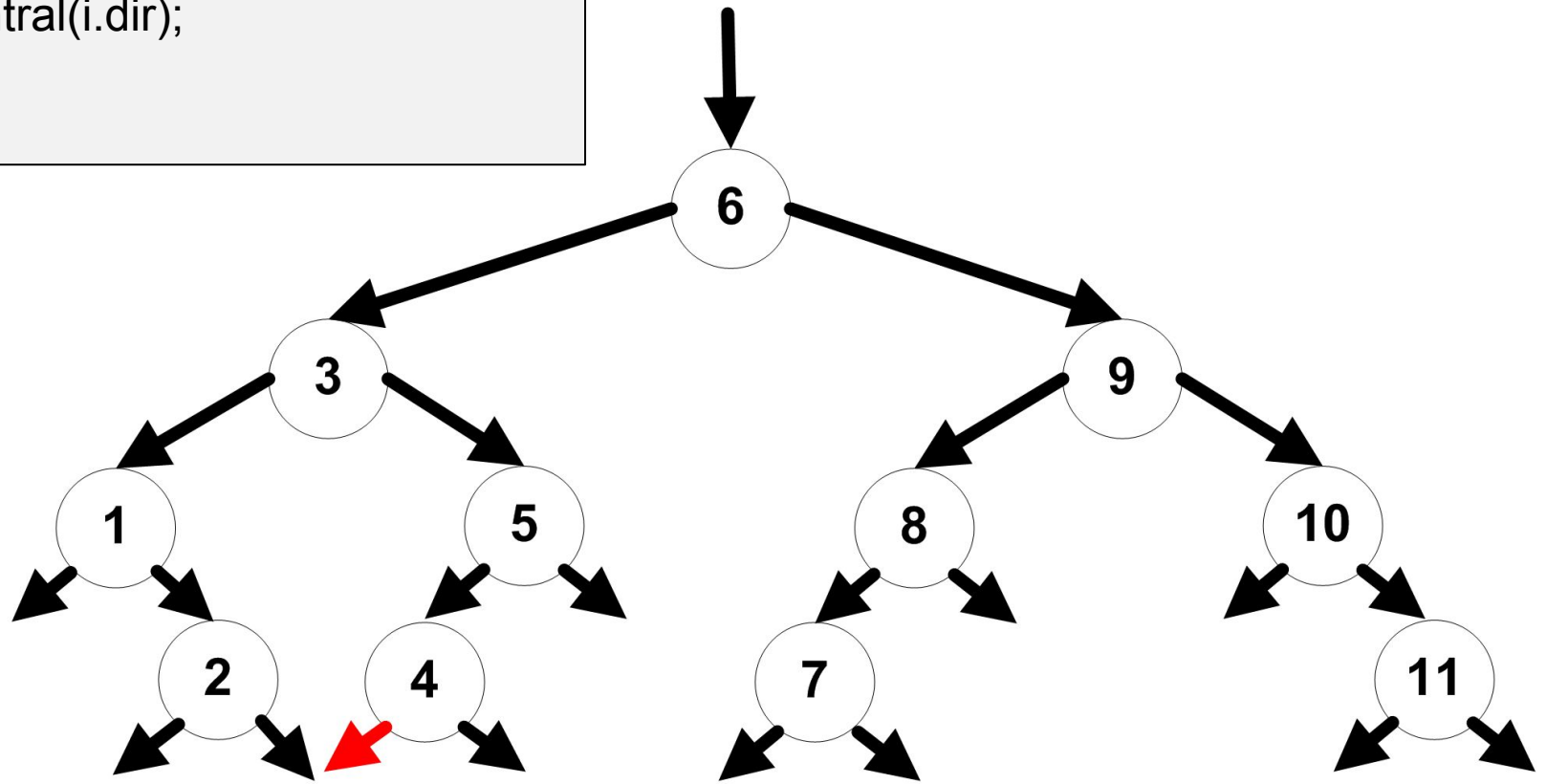
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3

Caminhamento Central ou Em Ordem

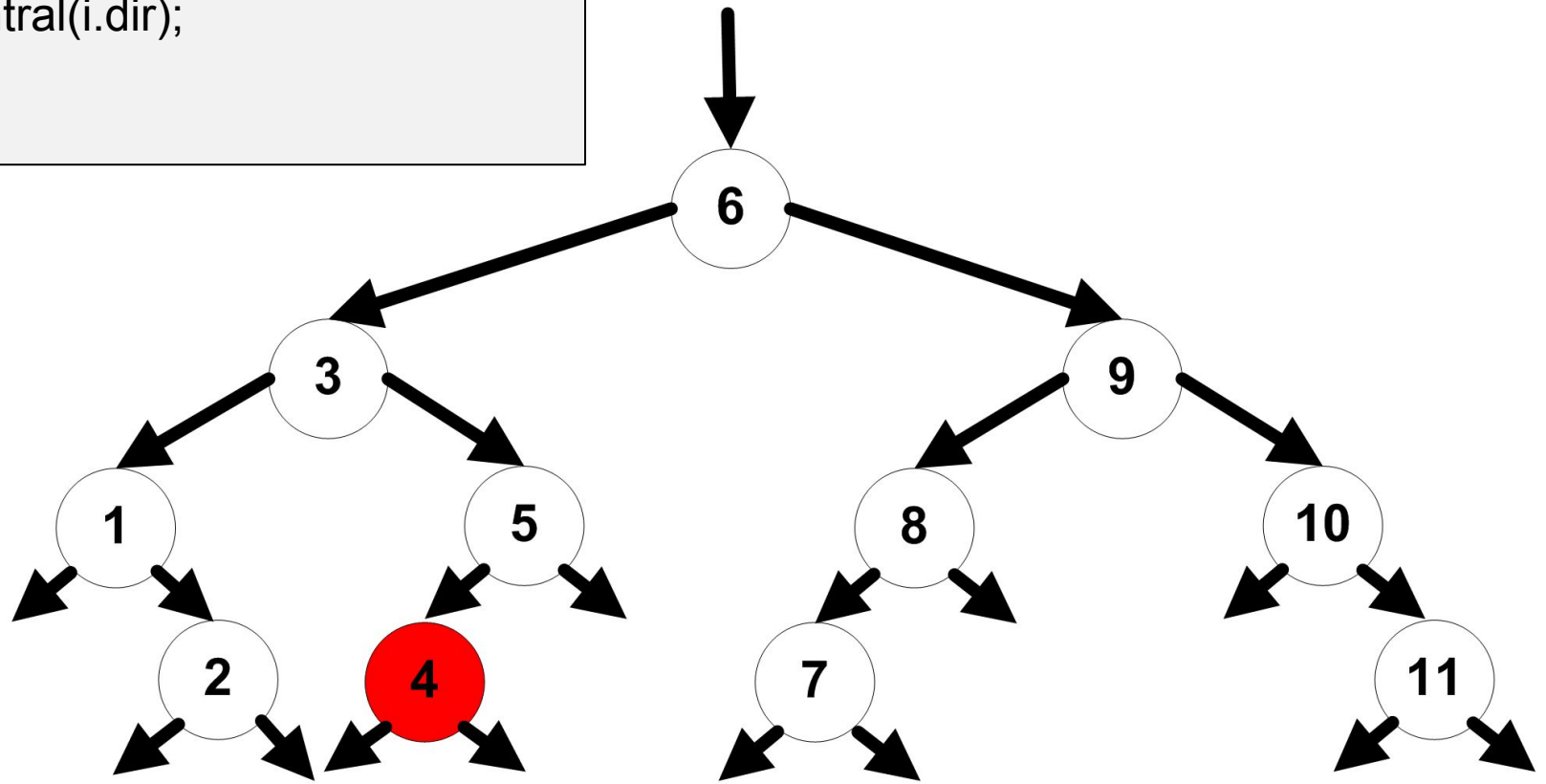
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3

Caminhamento Central ou Em Ordem

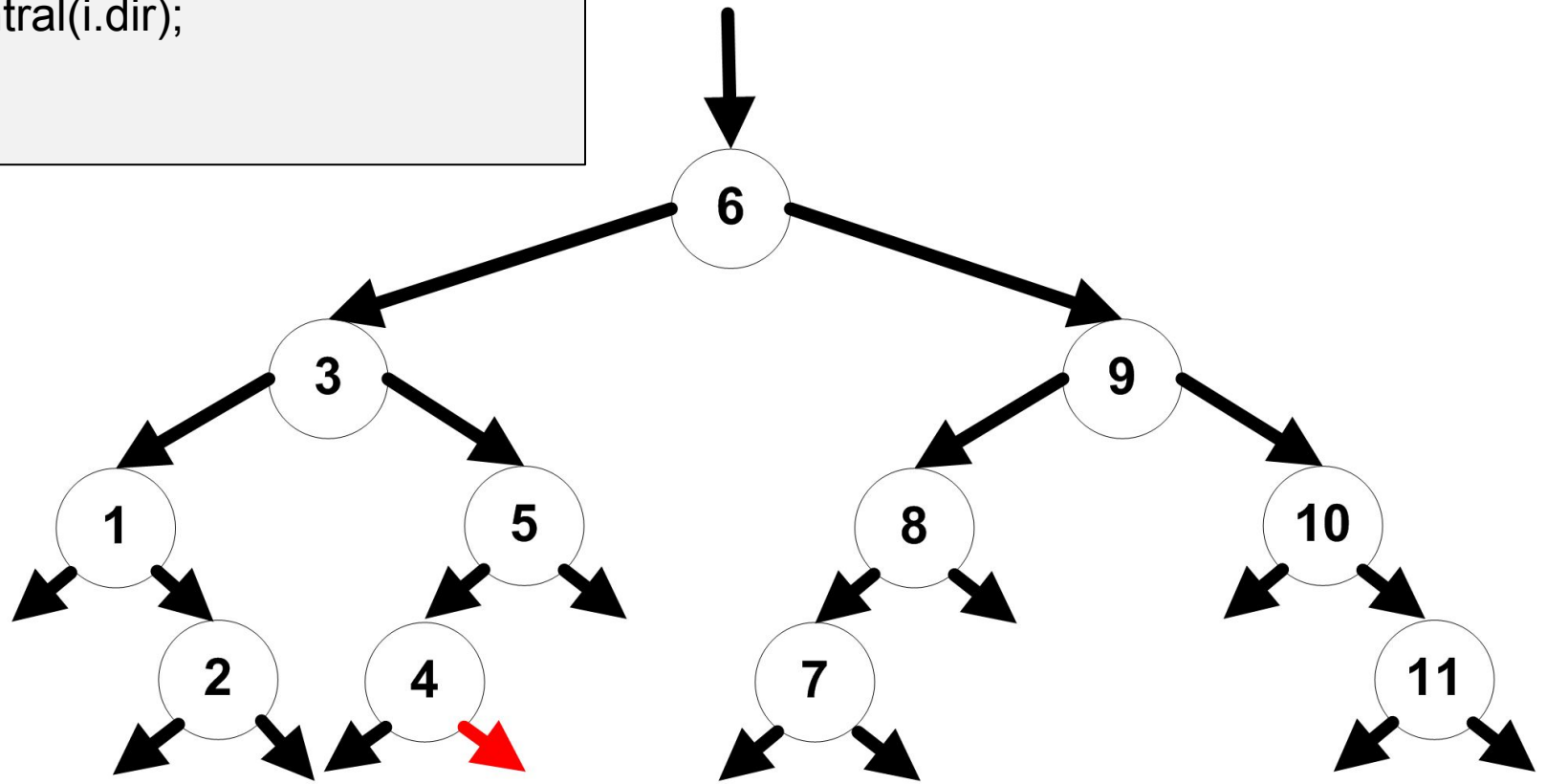
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3 4

Caminhamento Central ou Em Ordem

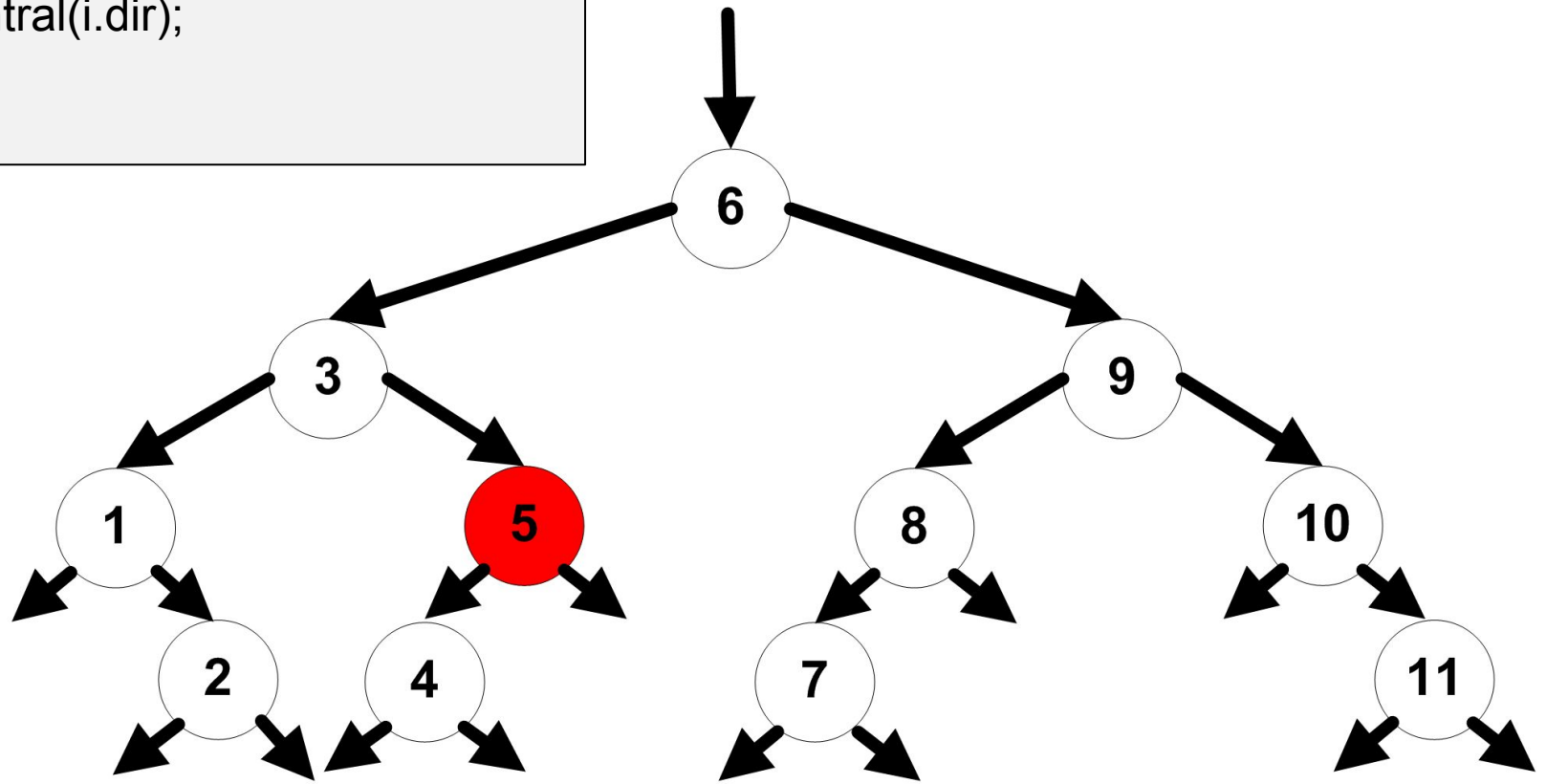
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3 4

Caminhamento Central ou Em Ordem

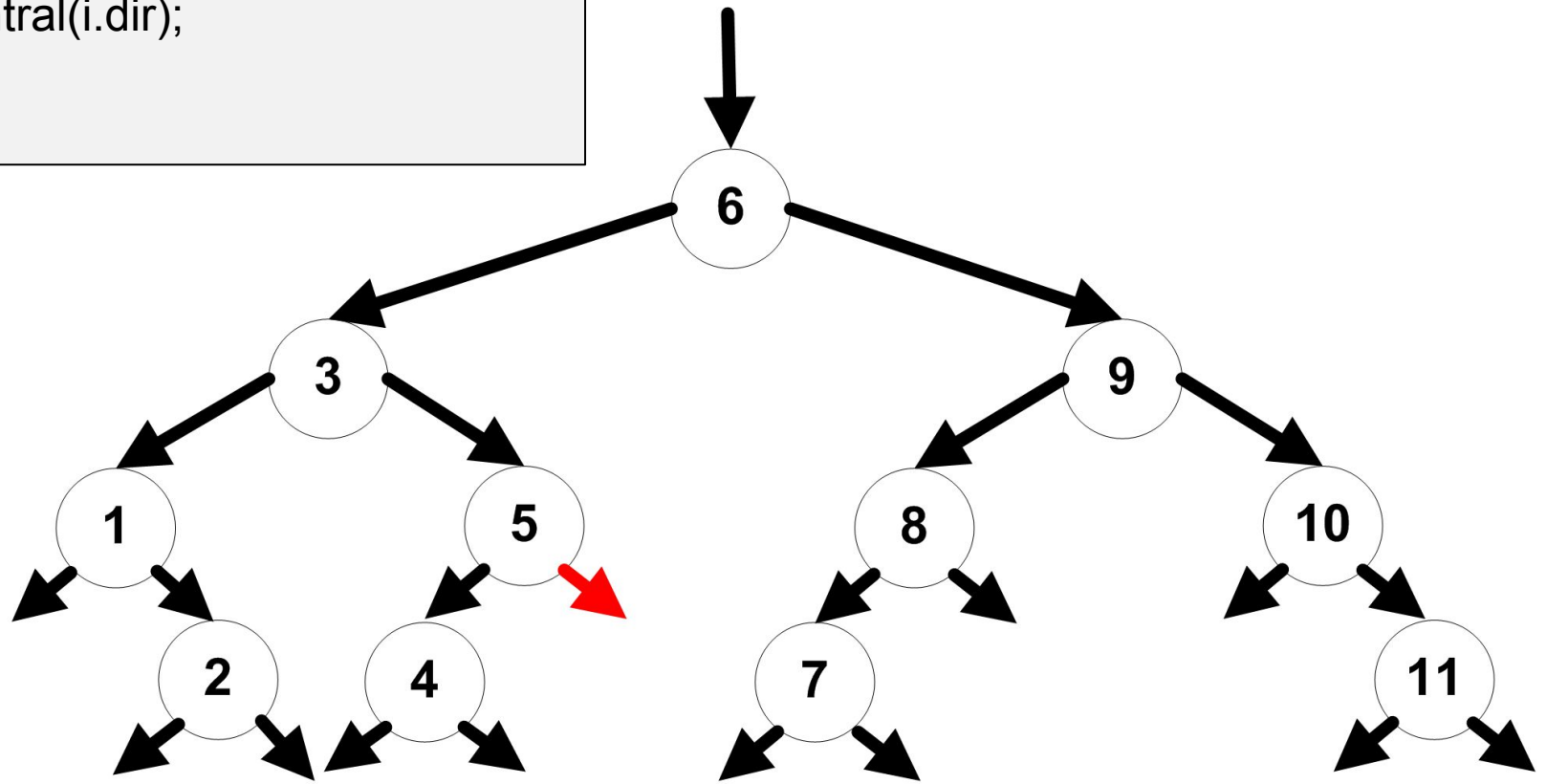
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3 4 5

Caminhamento Central ou Em Ordem

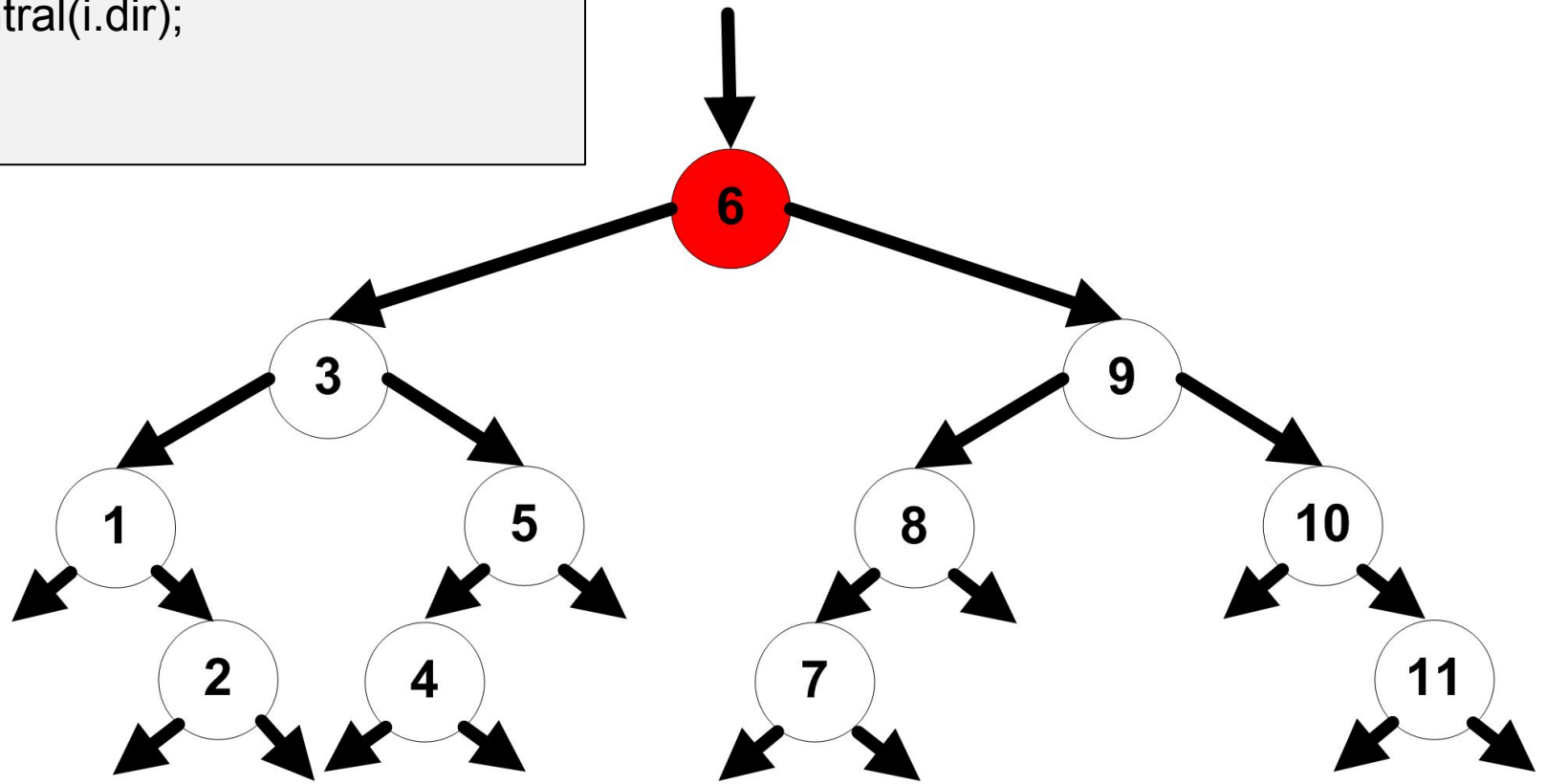
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3 4 5

Caminhamento Central ou Em Ordem

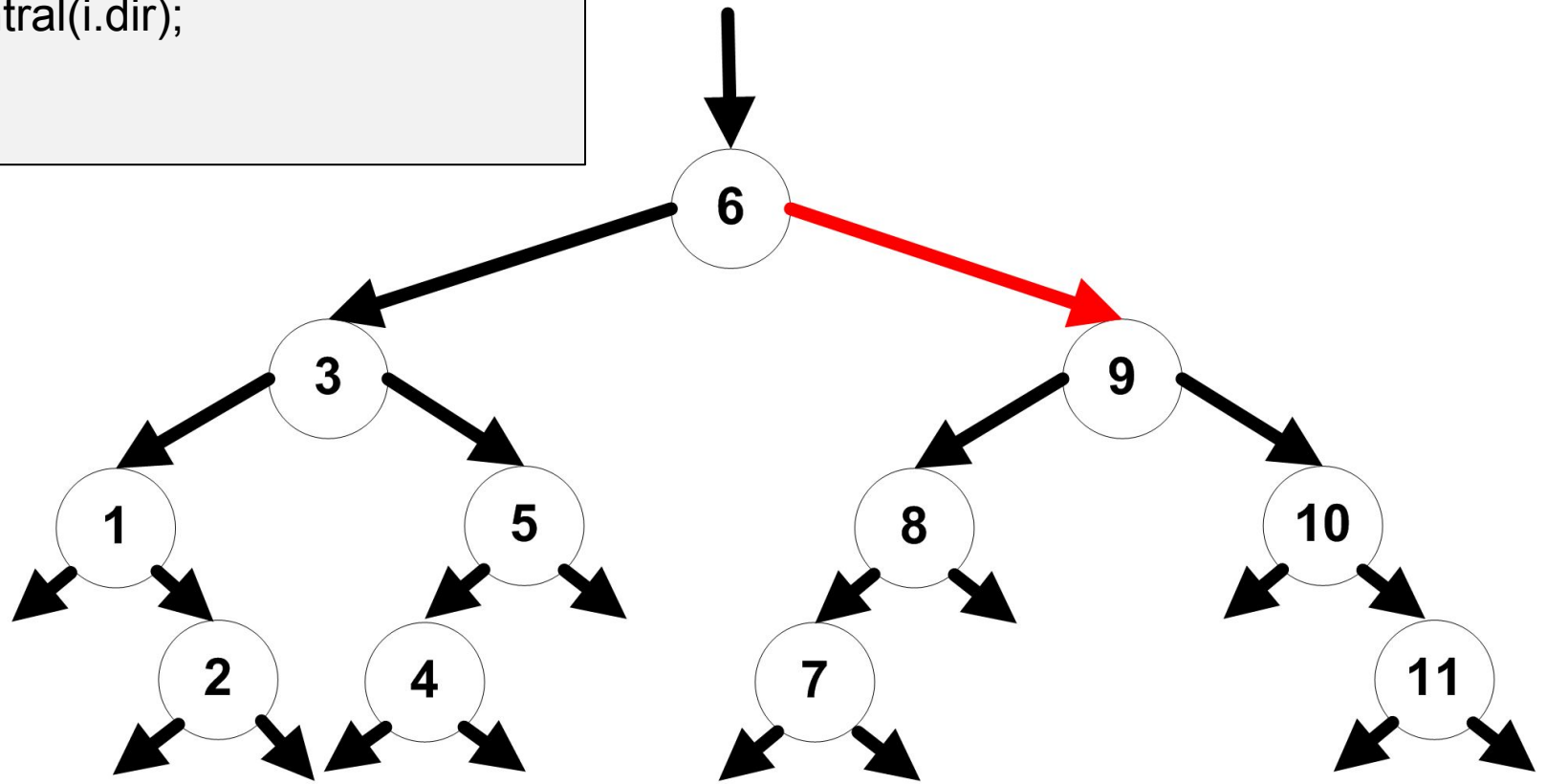
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3 4 5 6

Caminhamento Central ou Em Ordem

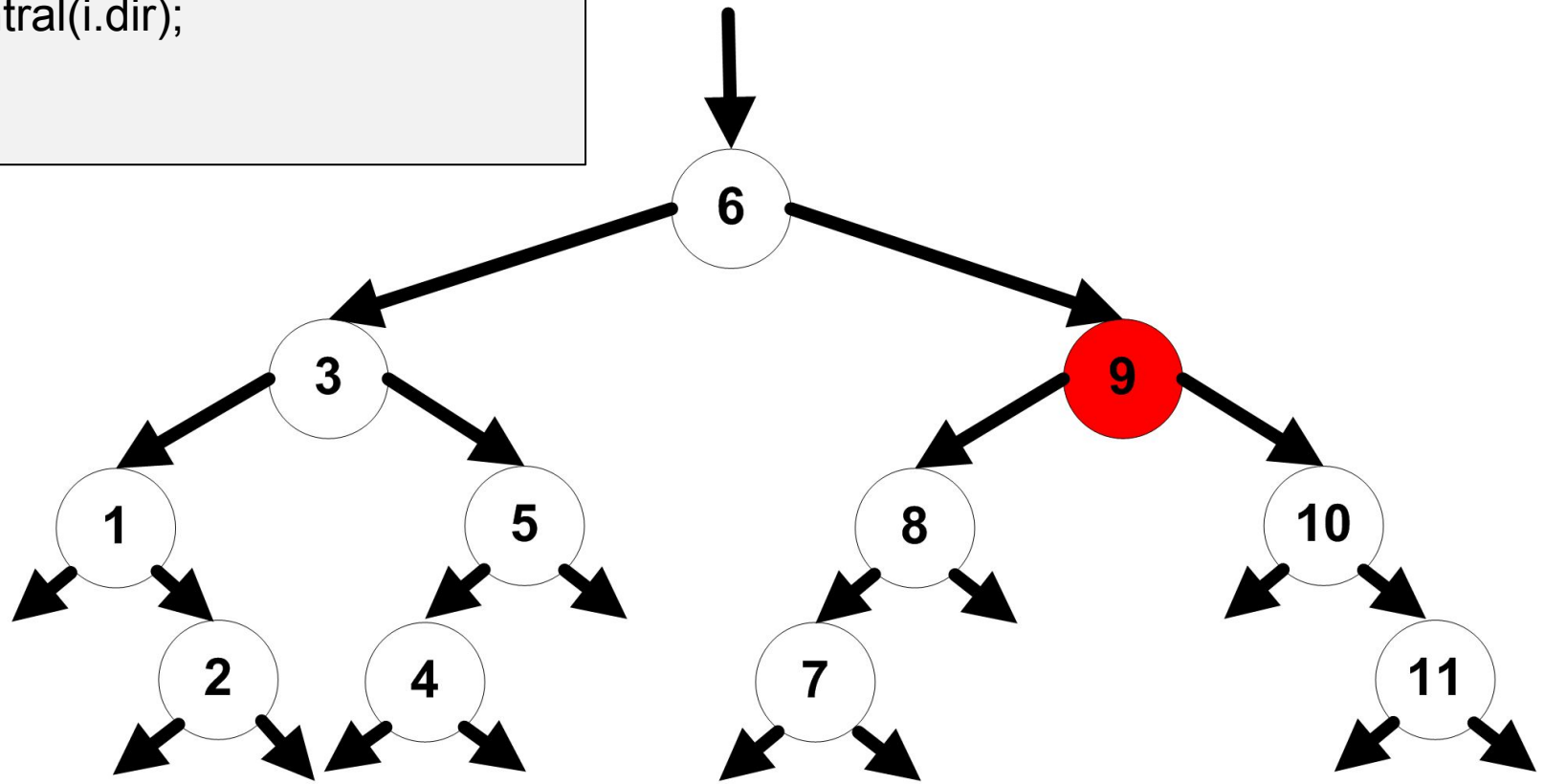
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3 4 5 6

Caminhamento Central ou Em Ordem

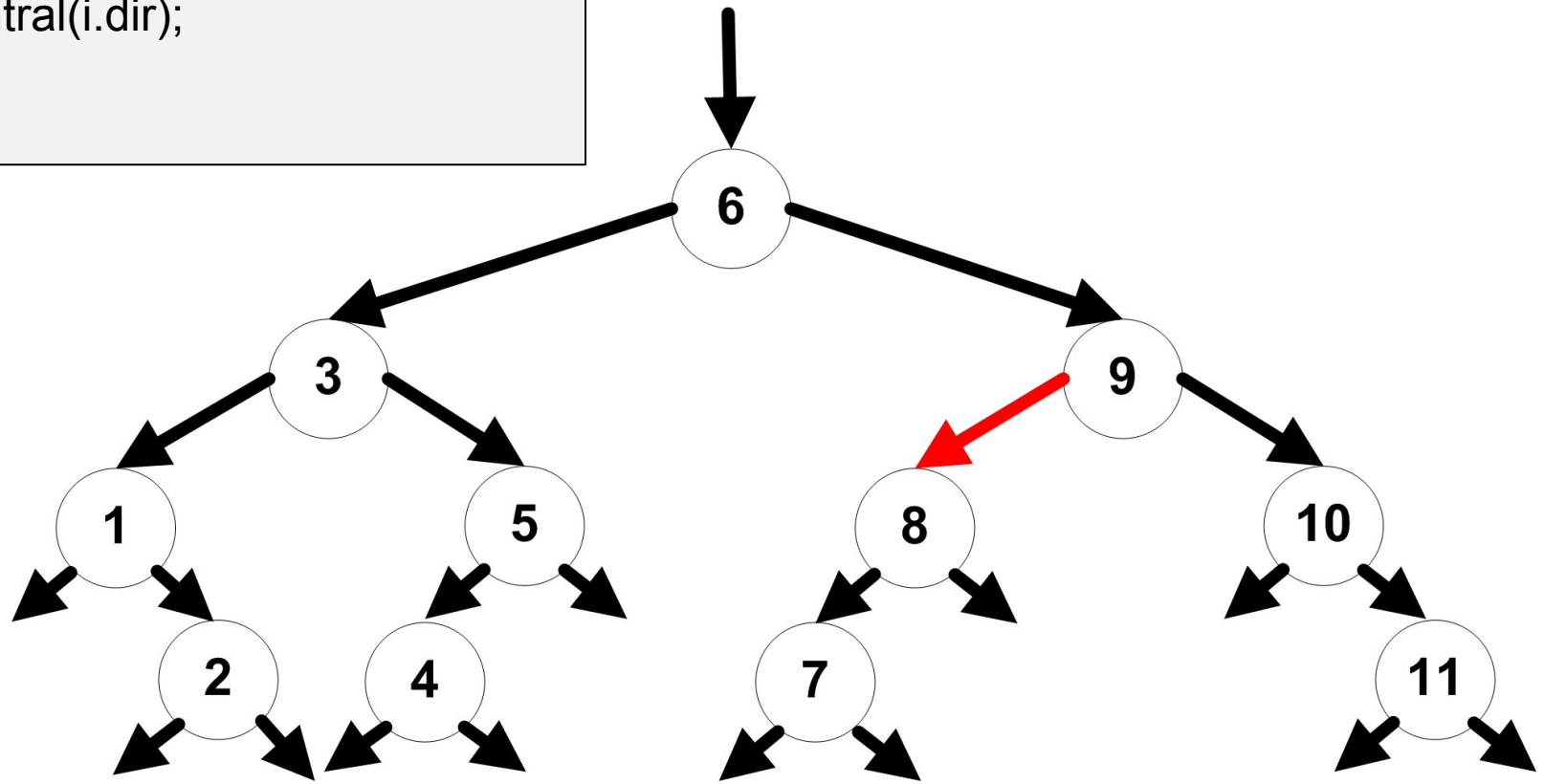
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3 4 5 6

Caminhamento Central ou Em Ordem

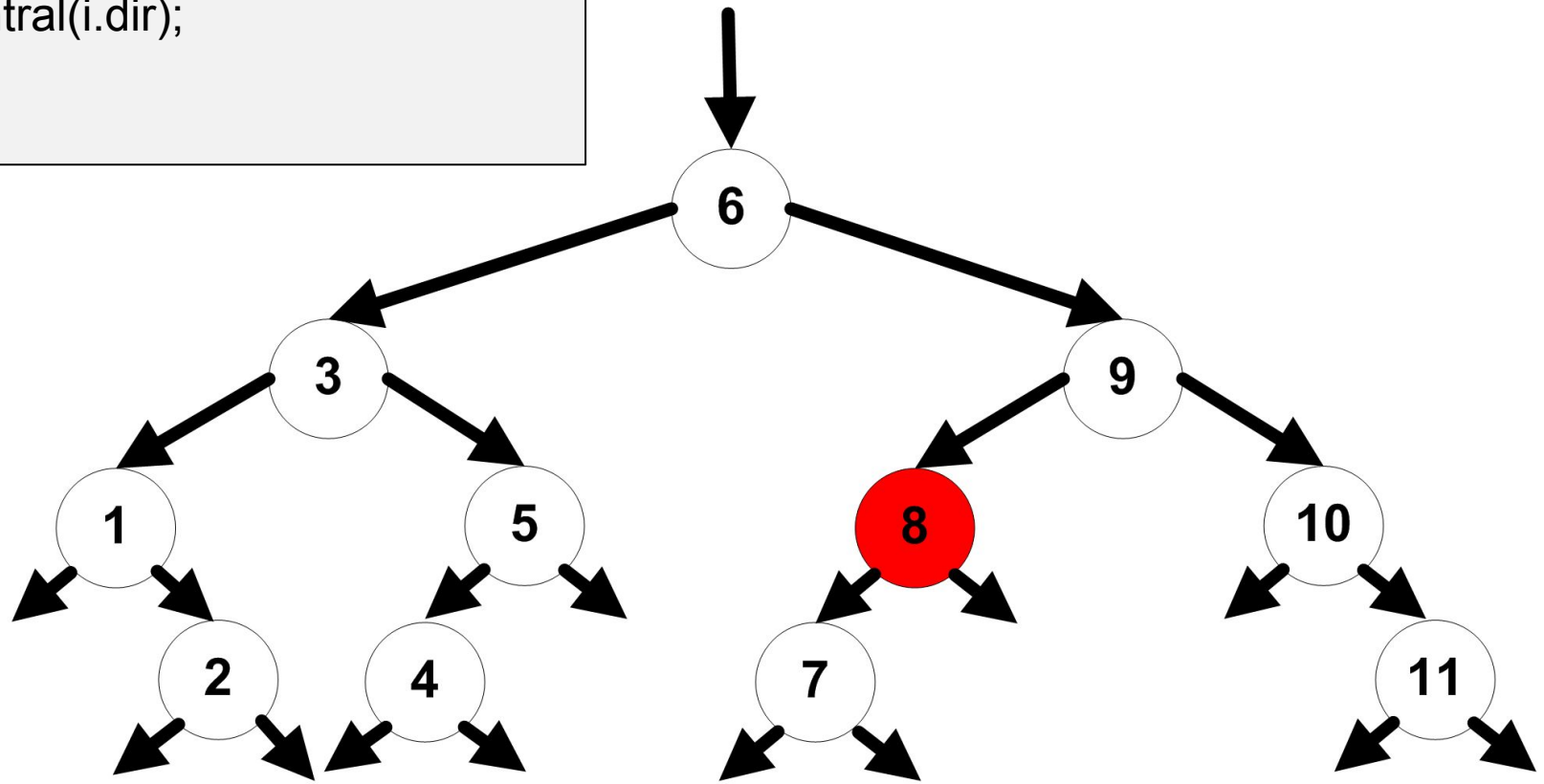
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6

Caminhamento Central ou Em Ordem

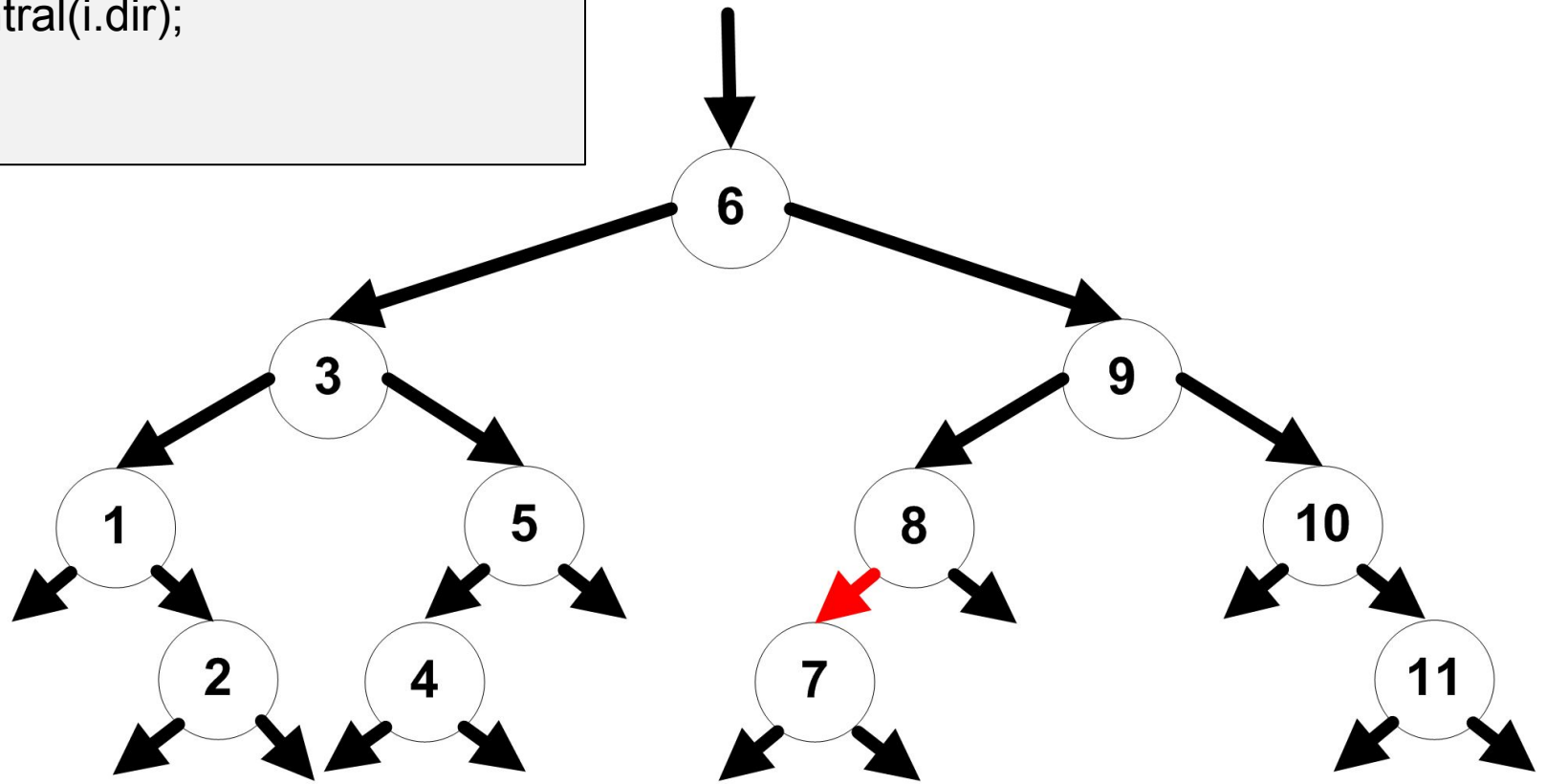
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3 4 5 6

Caminhamento Central ou Em Ordem

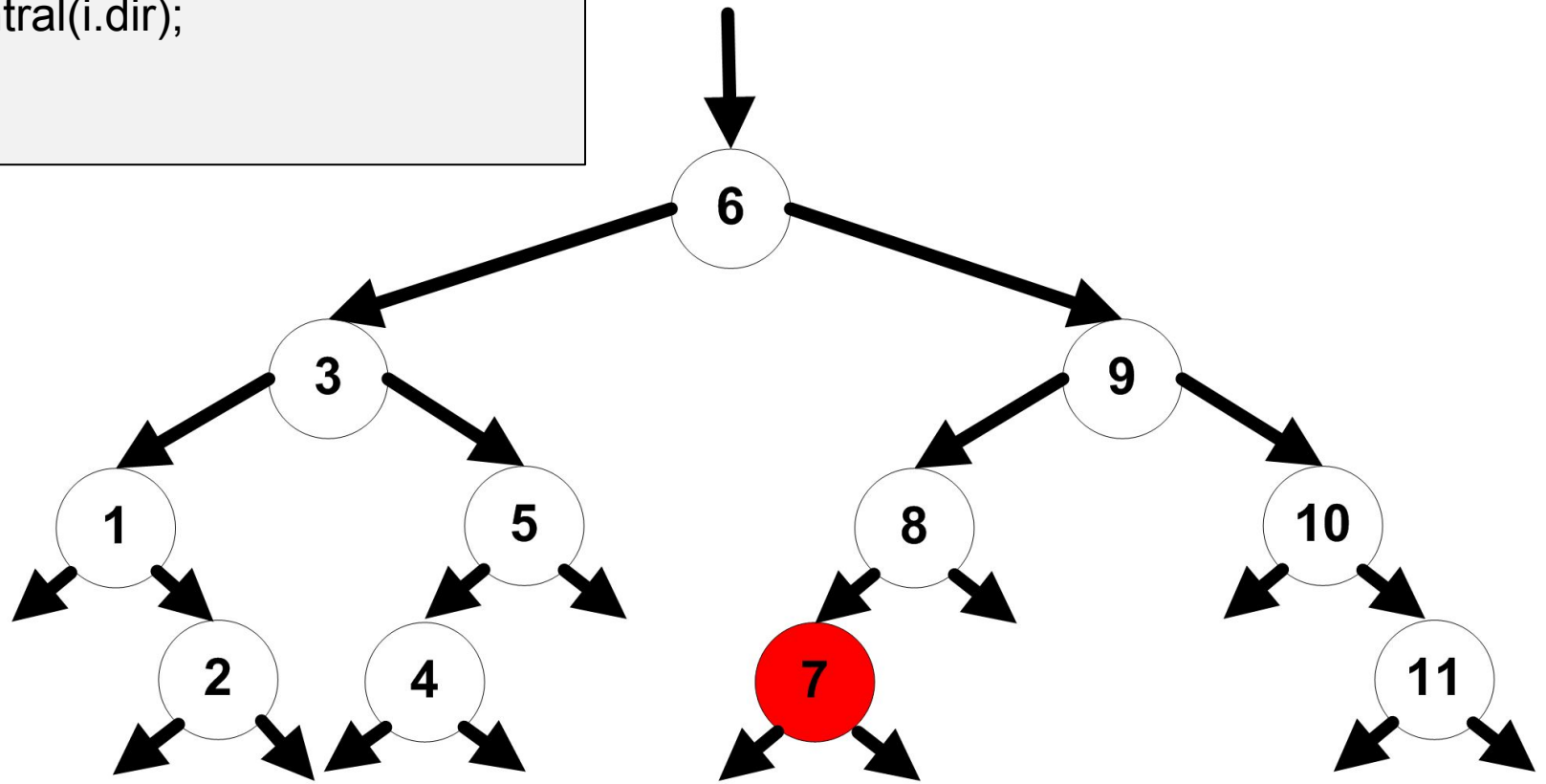
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1 2 3 4 5 6

Caminhamento Central ou Em Ordem

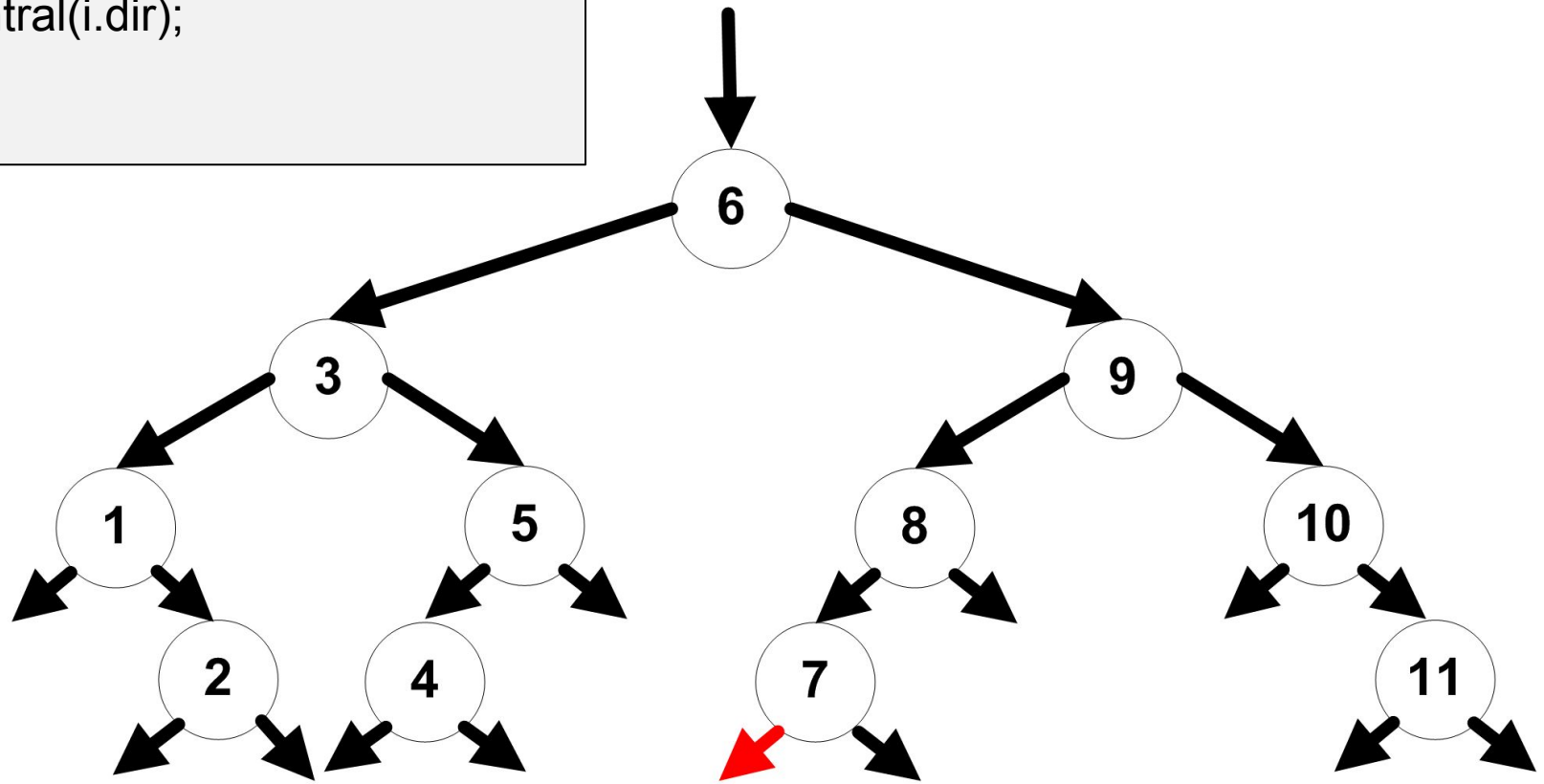
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6

Caminhamento Central ou Em Ordem

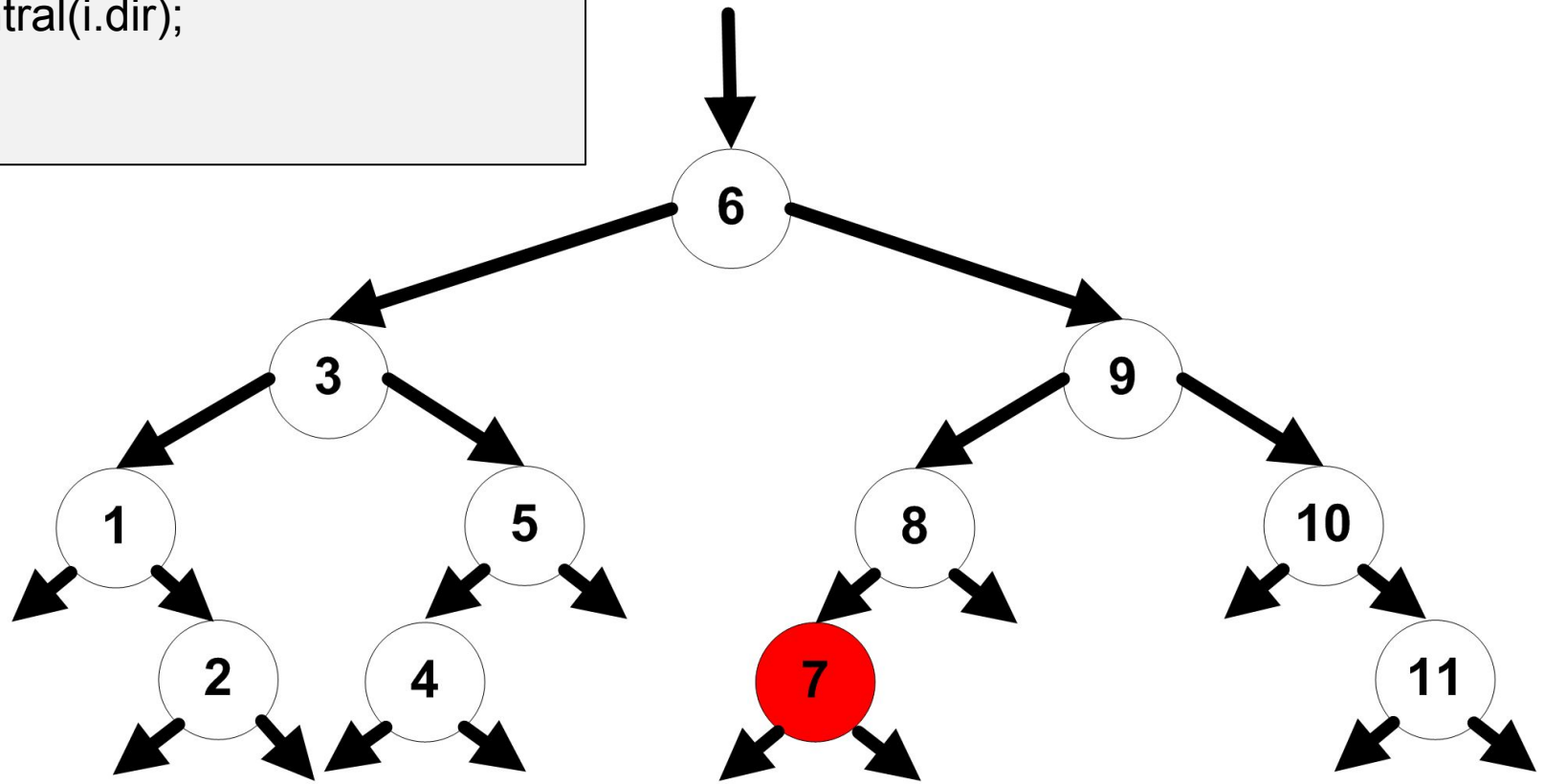
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6

Caminhamento Central ou Em Ordem

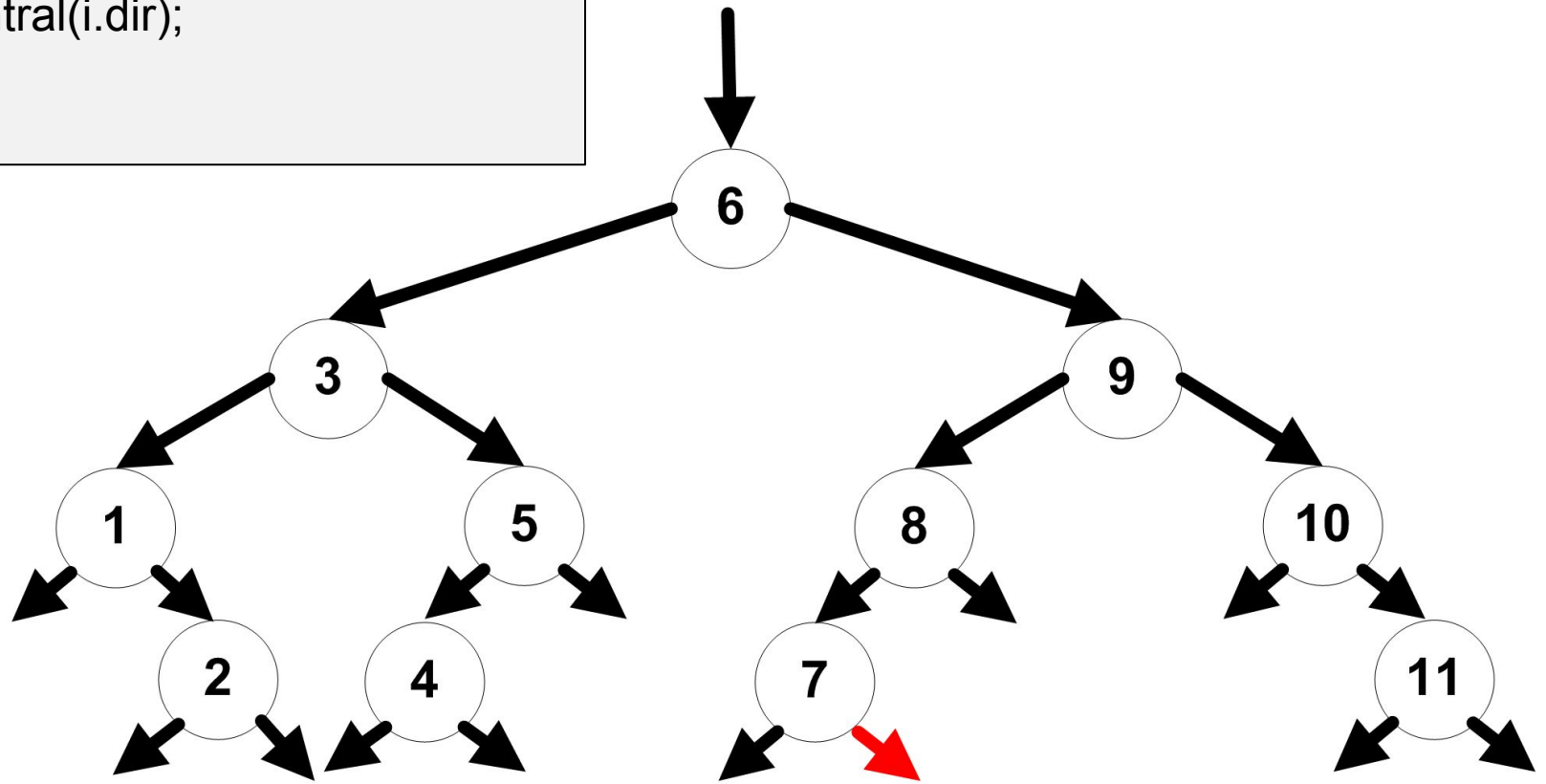
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6	7

Caminhamento Central ou Em Ordem

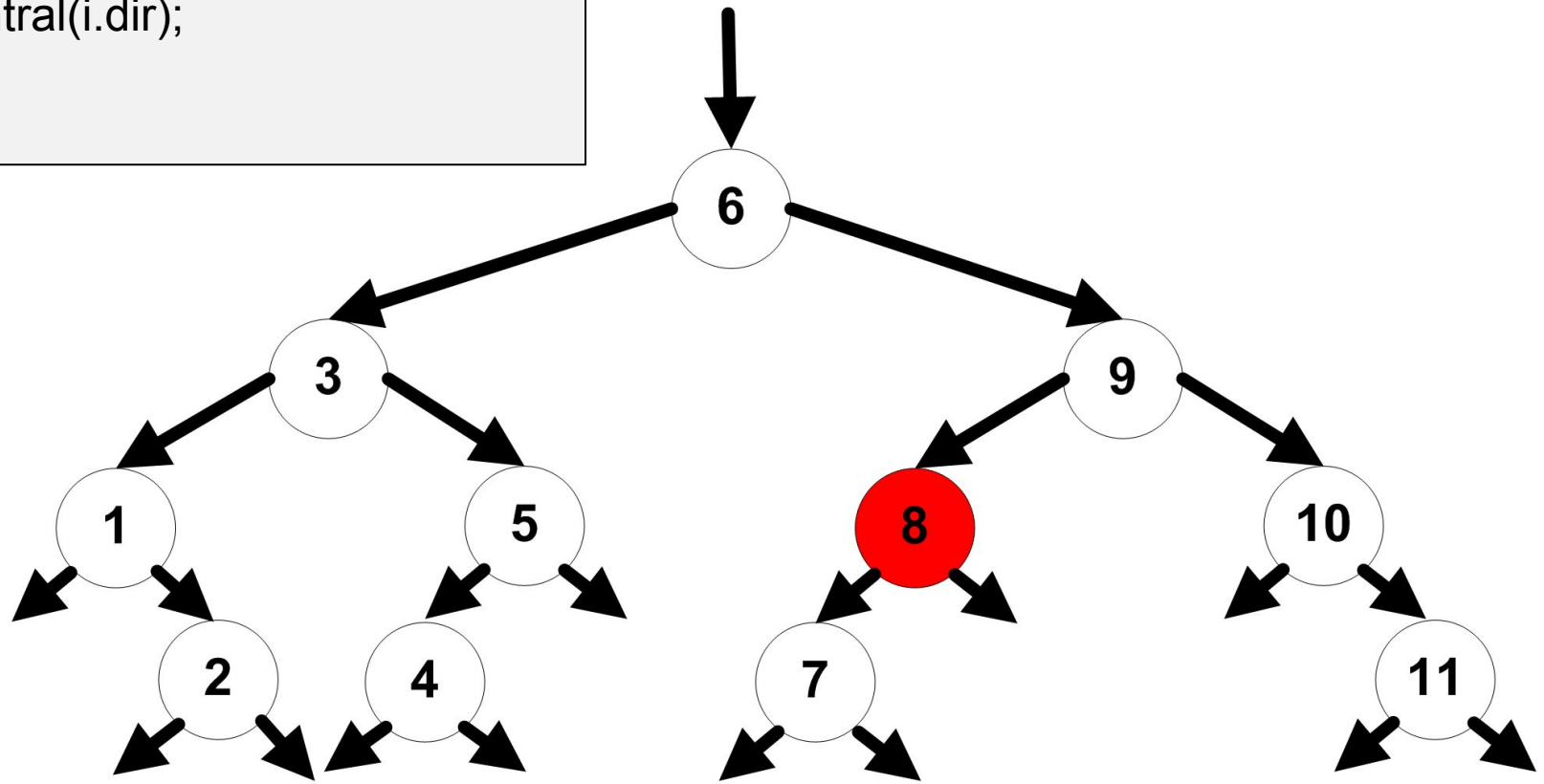
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6	7

Caminhamento Central ou Em Ordem

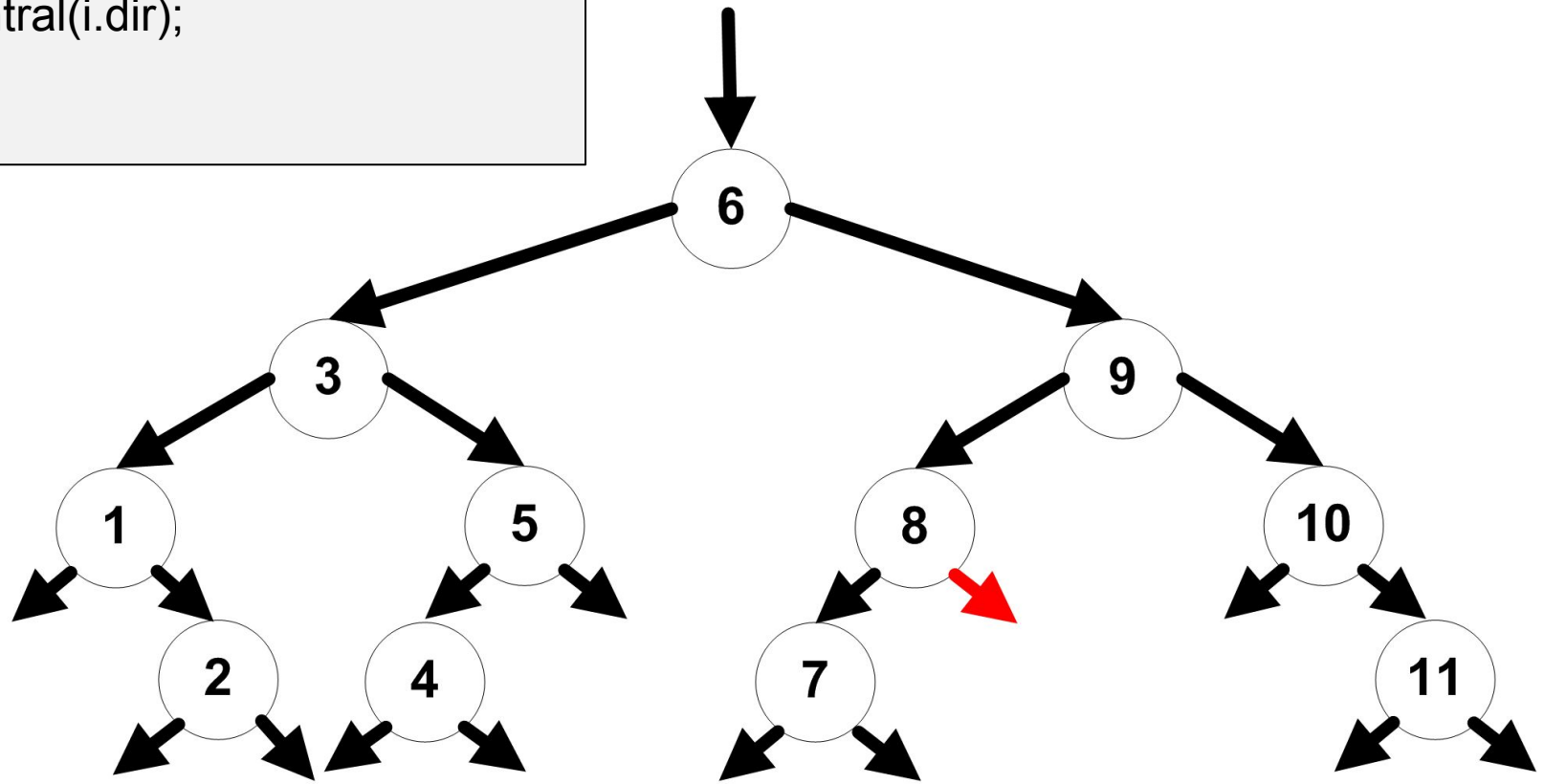
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6	7	8

Caminhamento Central ou Em Ordem

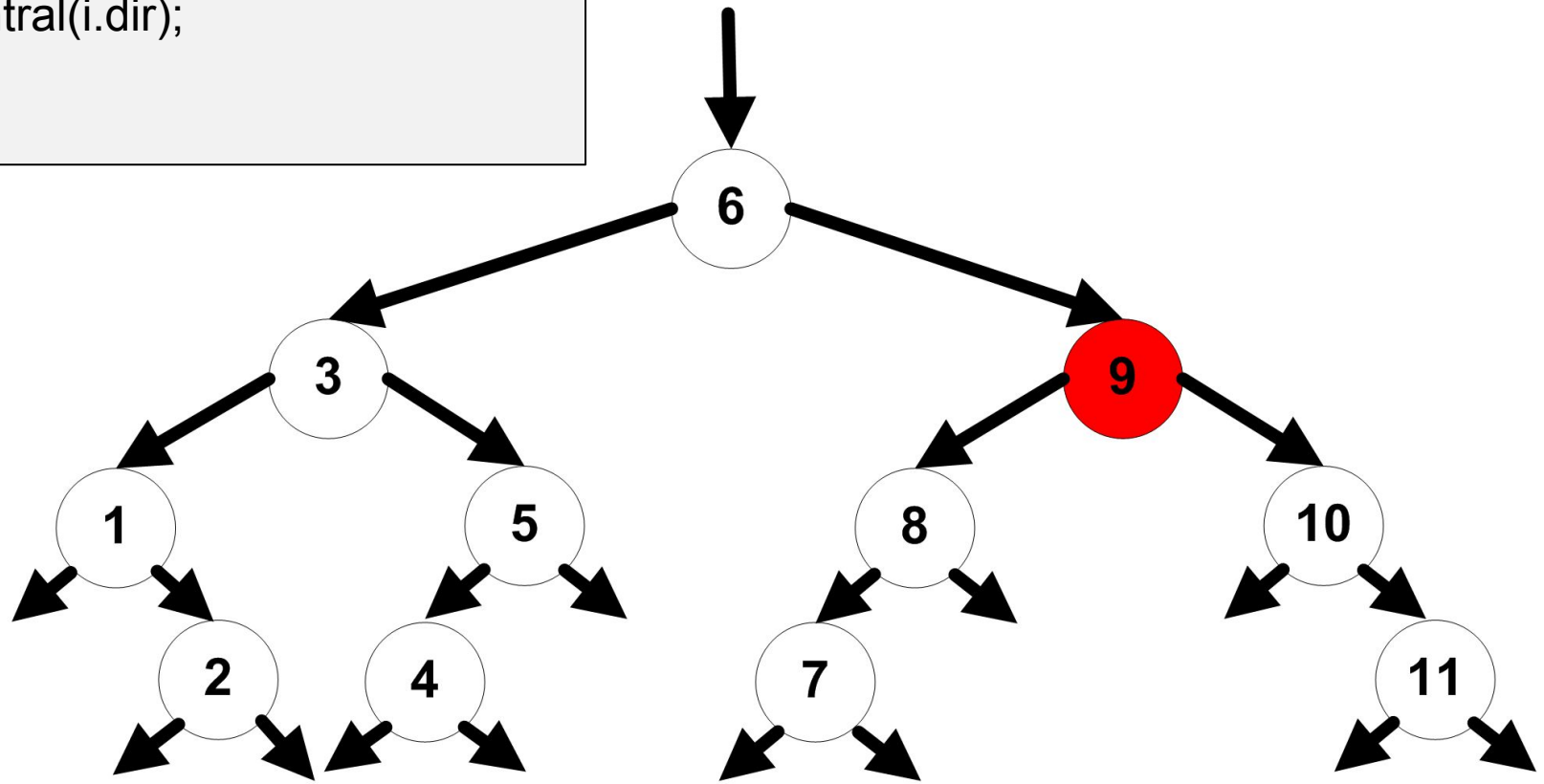
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6	7	8

Caminhamento Central ou Em Ordem

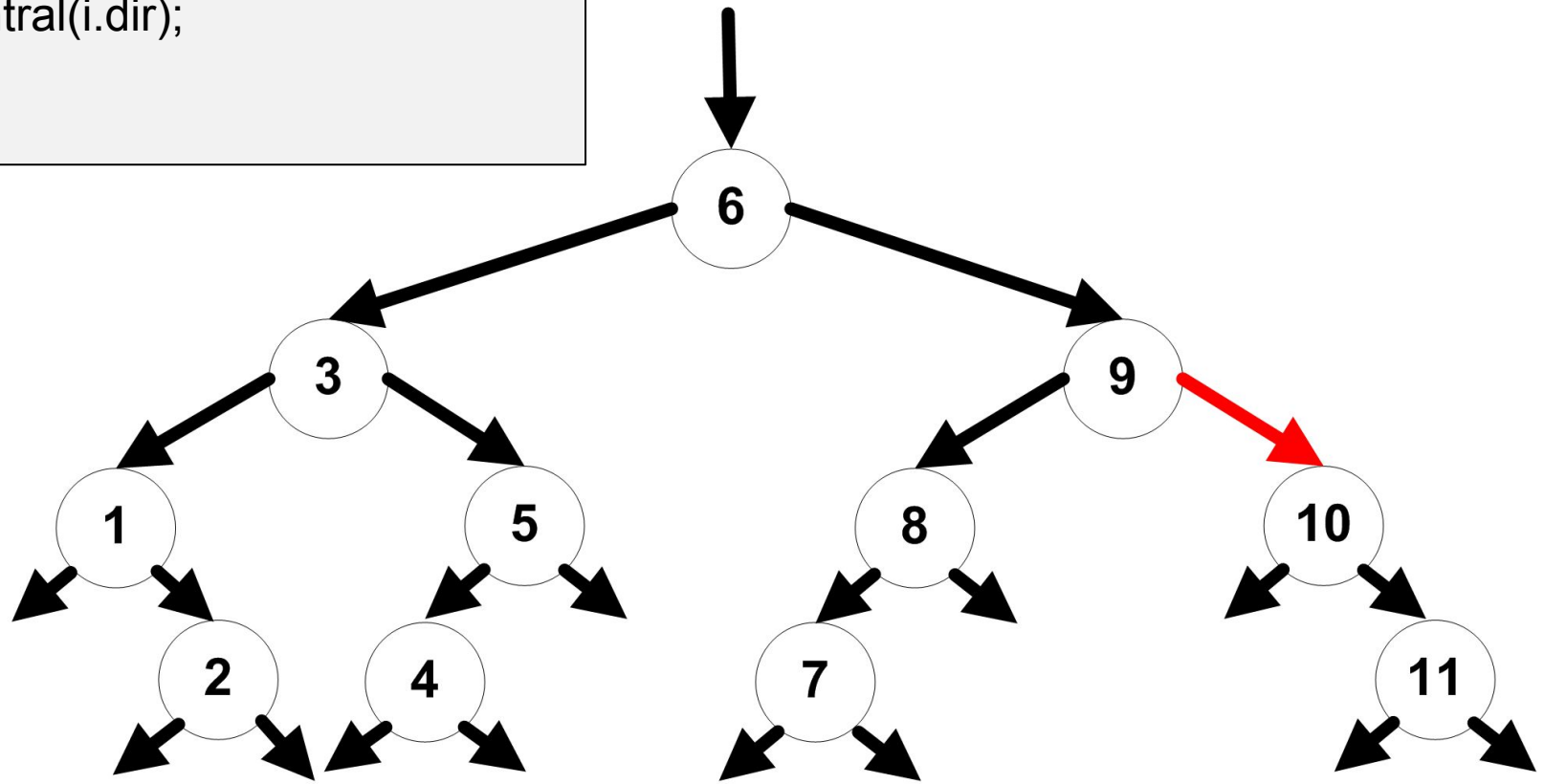
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6	7	8	9

Caminhamento Central ou Em Ordem

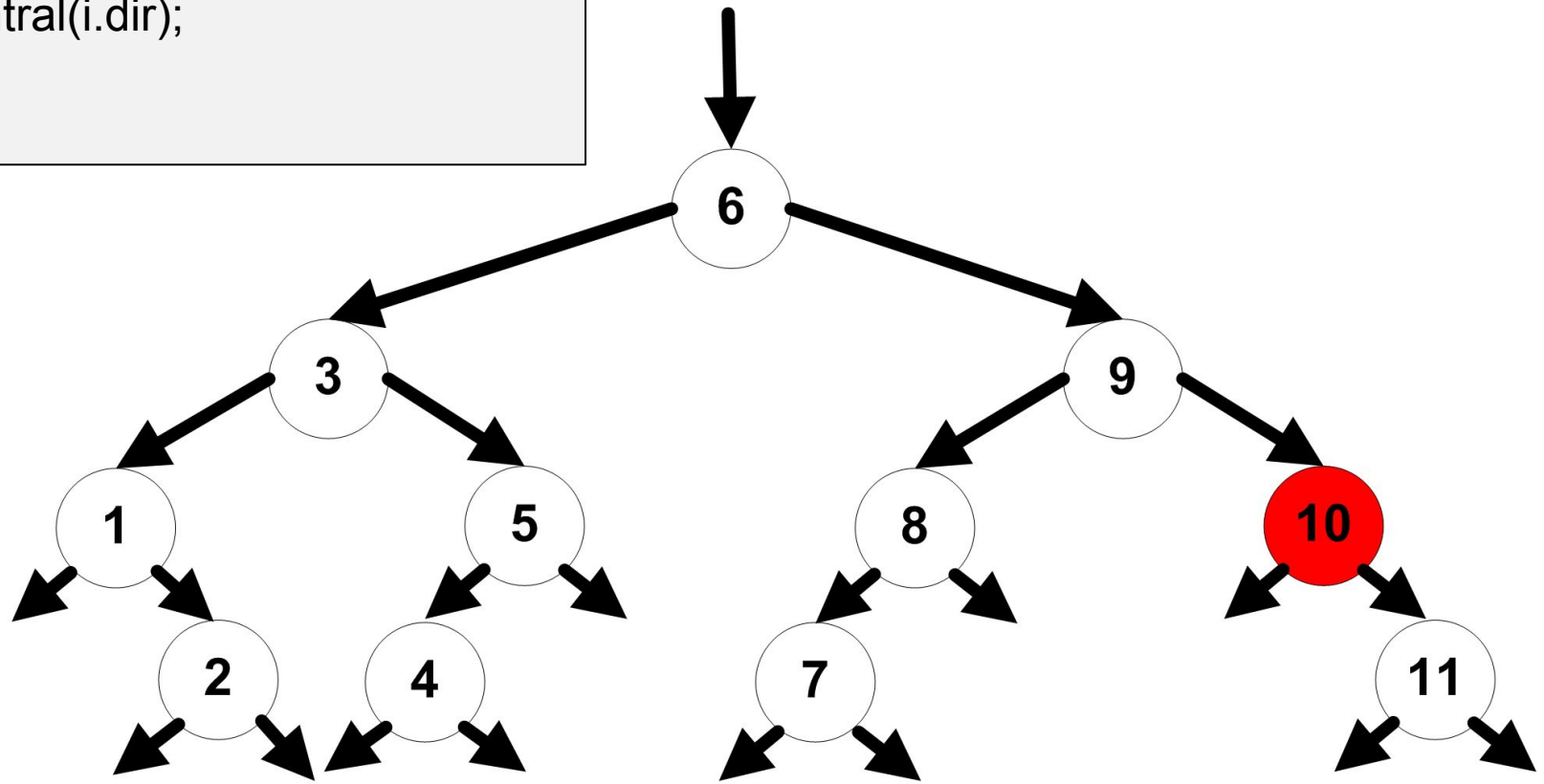
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6	7	8	9

Caminhamento Central ou Em Ordem

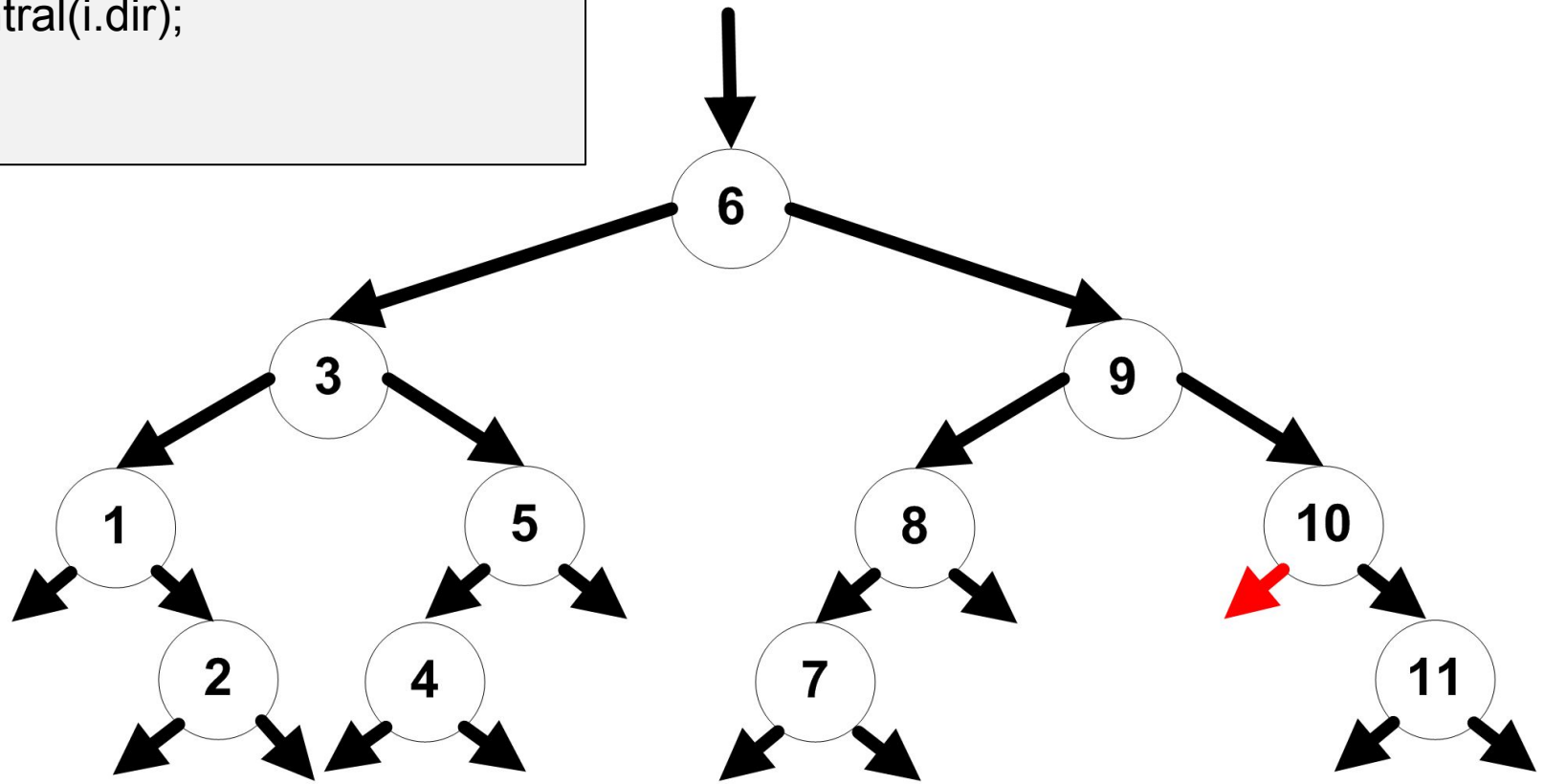
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6	7	8	9

Caminhamento Central ou Em Ordem

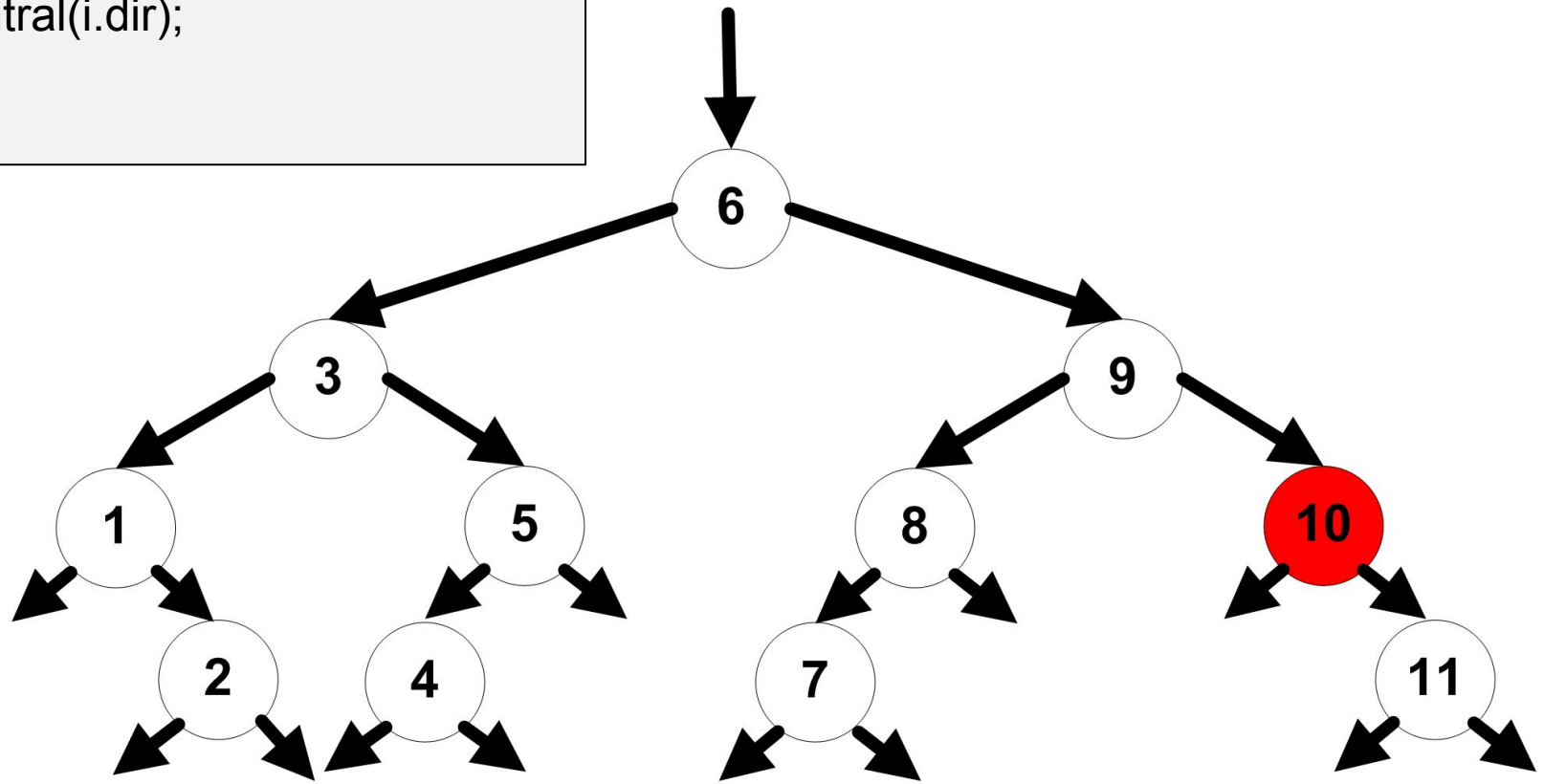
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6	7	8	9

Caminhamento Central ou Em Ordem

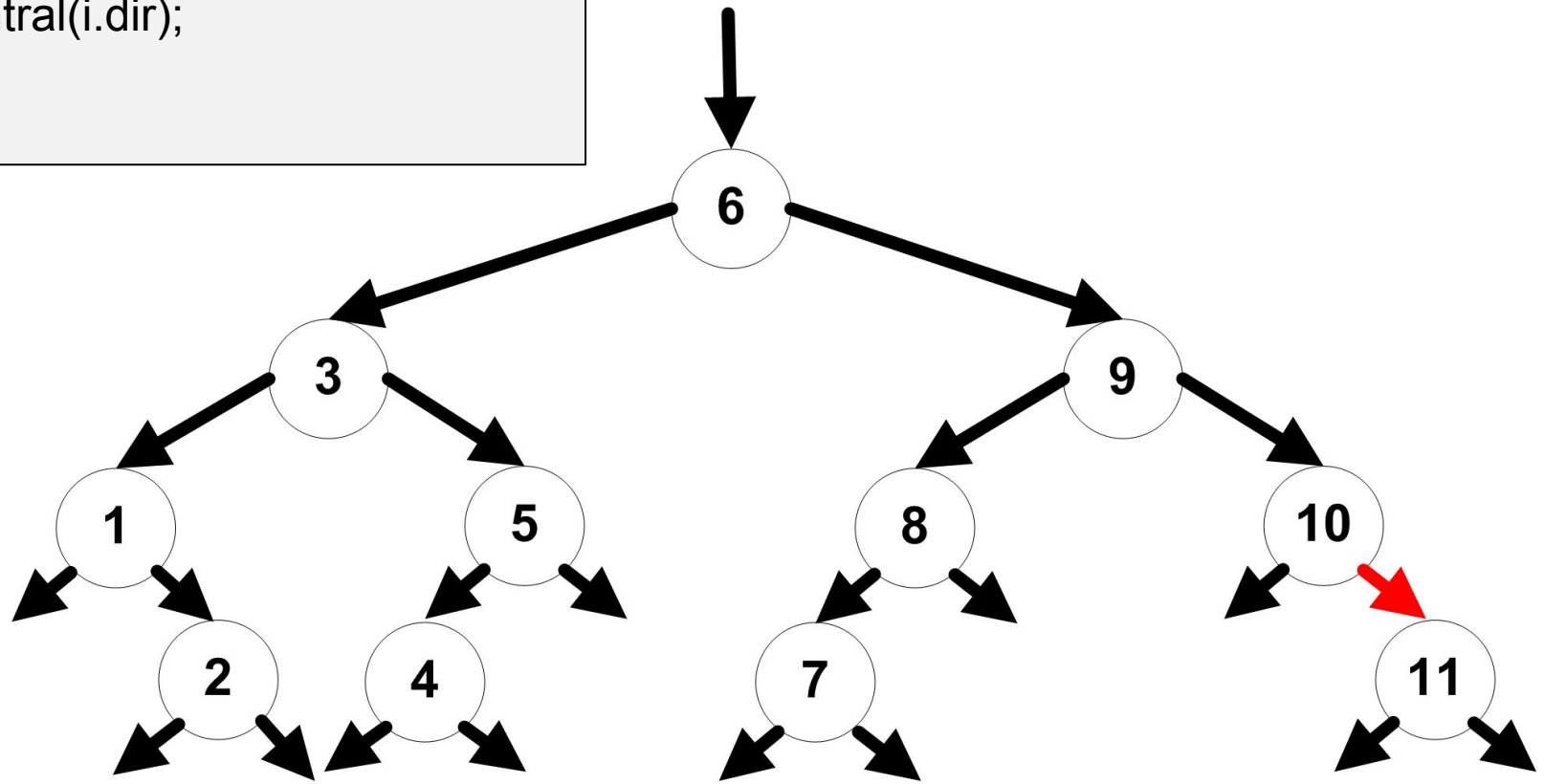
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6	7	8	9	10

Caminhamento Central ou Em Ordem

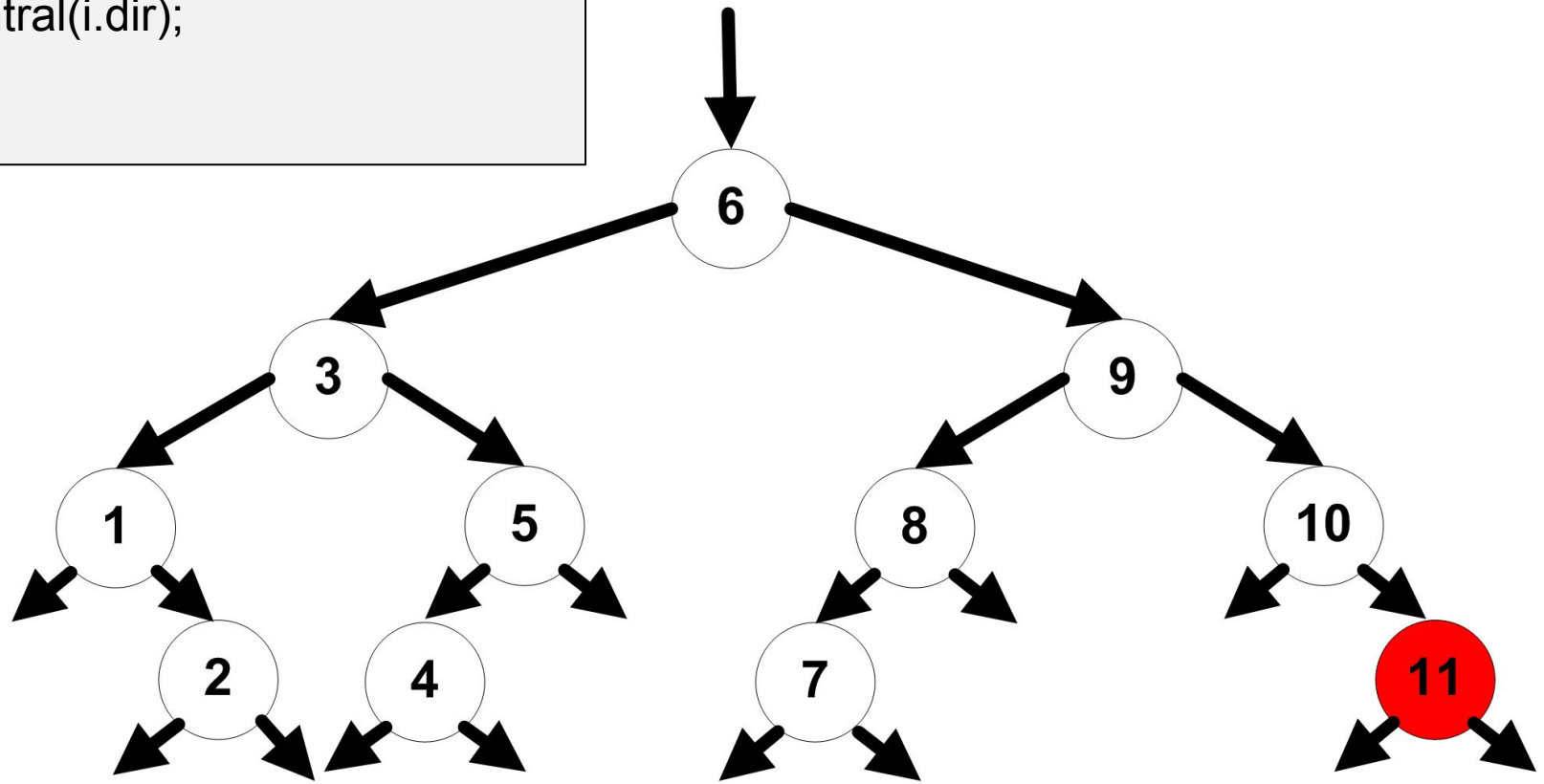
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6	7	8	9	10

Caminhamento Central ou Em Ordem

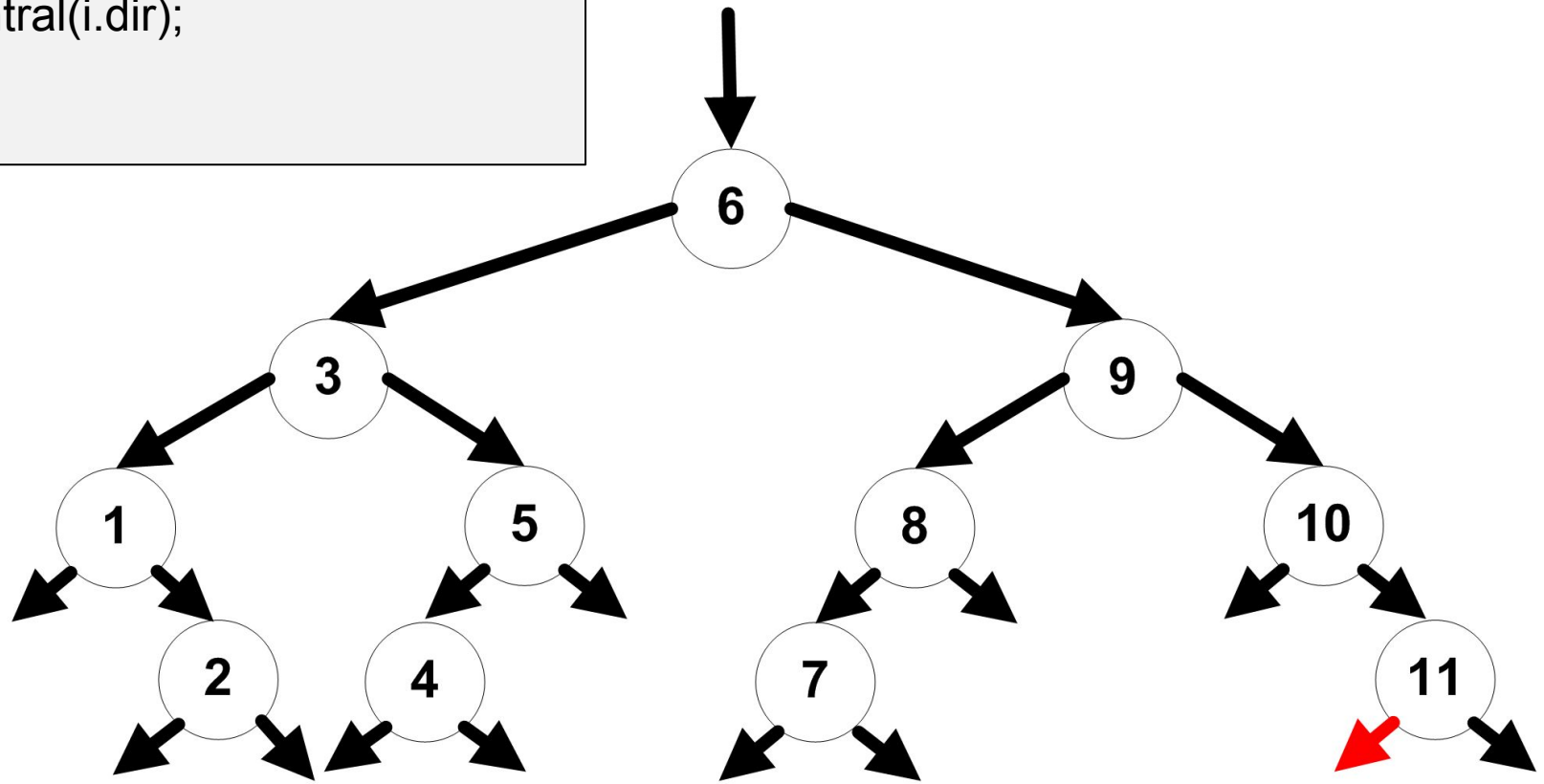
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6	7	8	9	10

Caminhamento Central ou Em Ordem

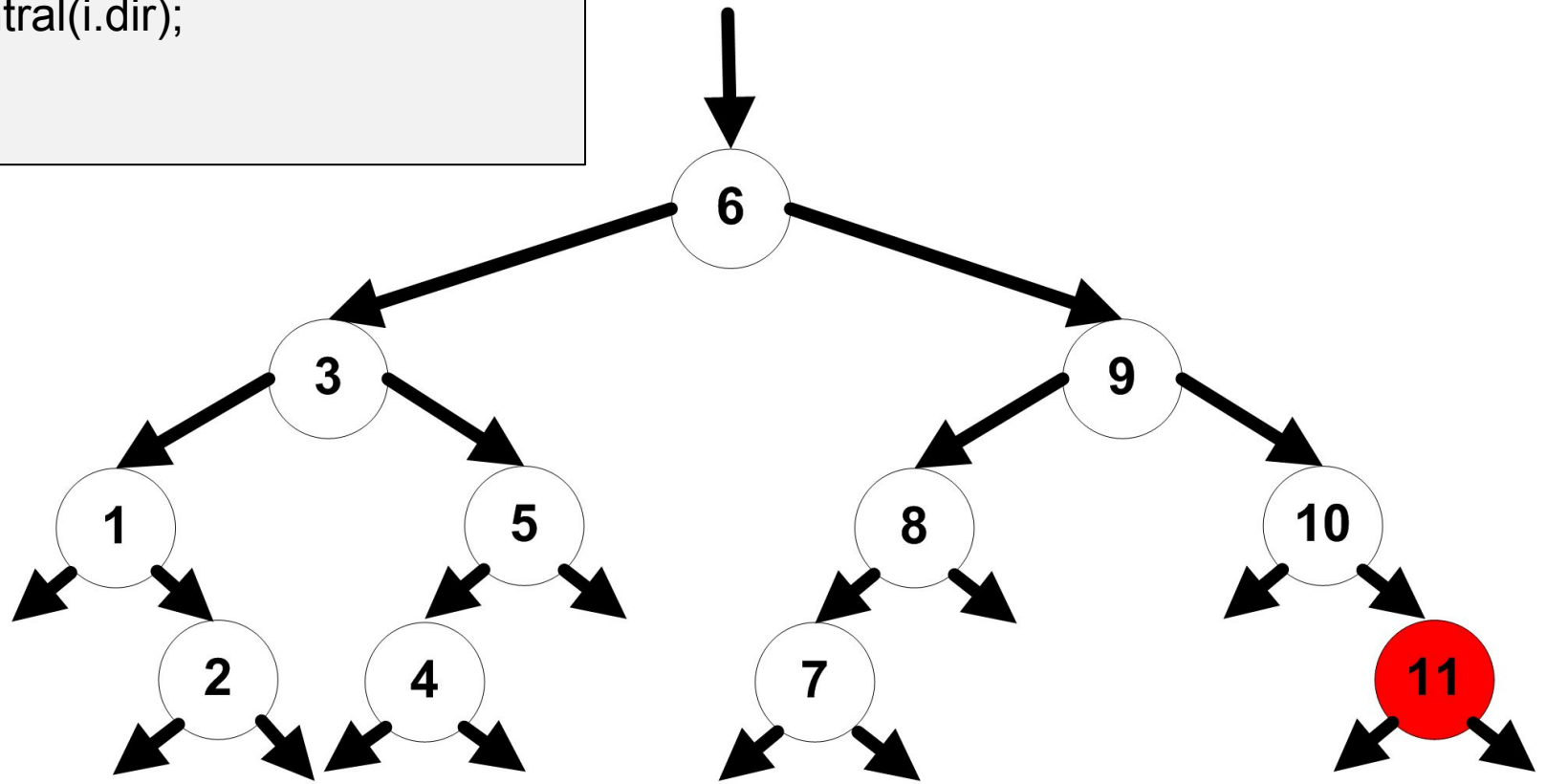
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



Tela	1	2	3	4	5	6	7	8	9	10

Caminhamento Central ou Em Ordem

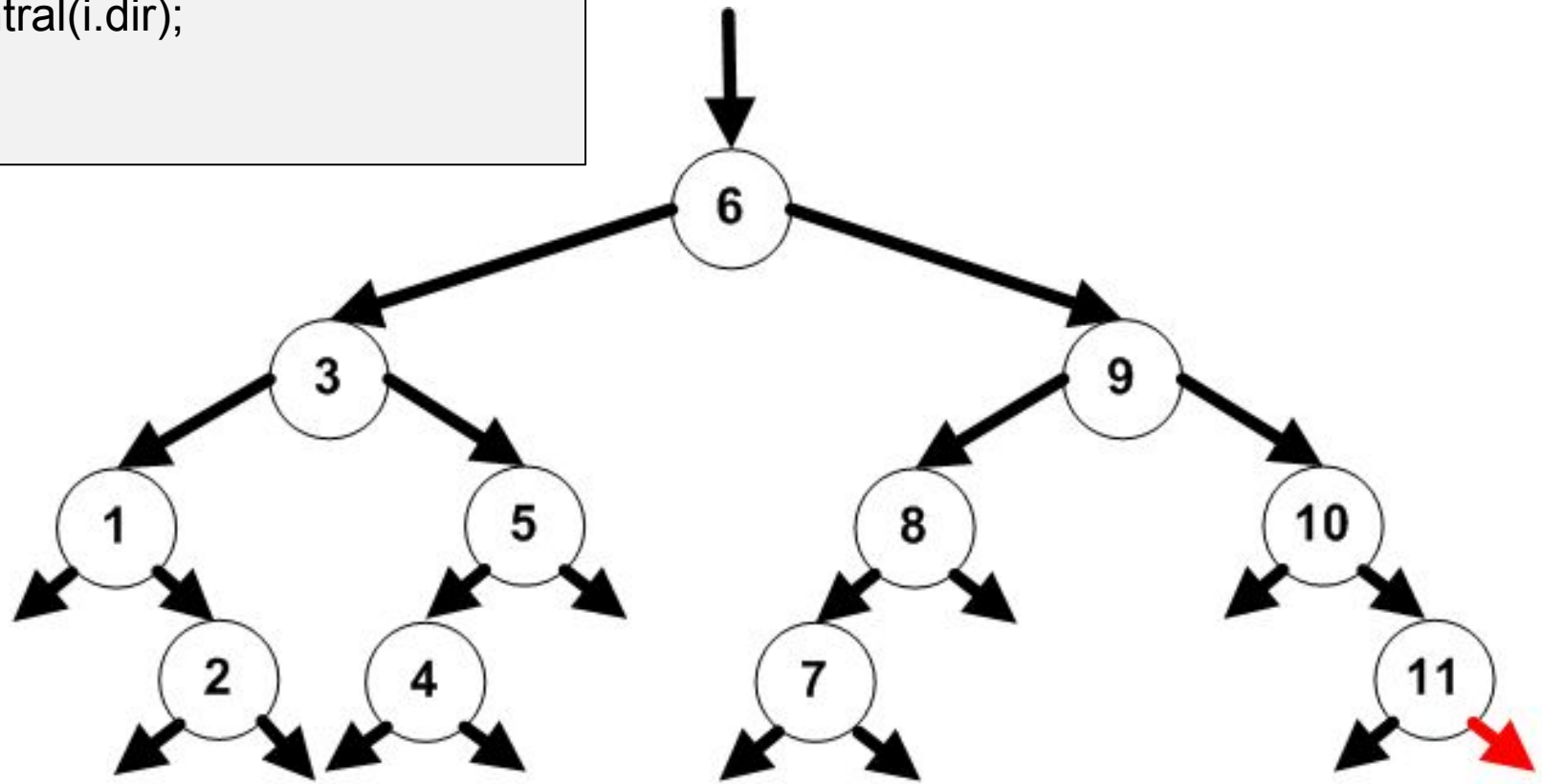
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



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

Caminhamento Central ou Em Ordem

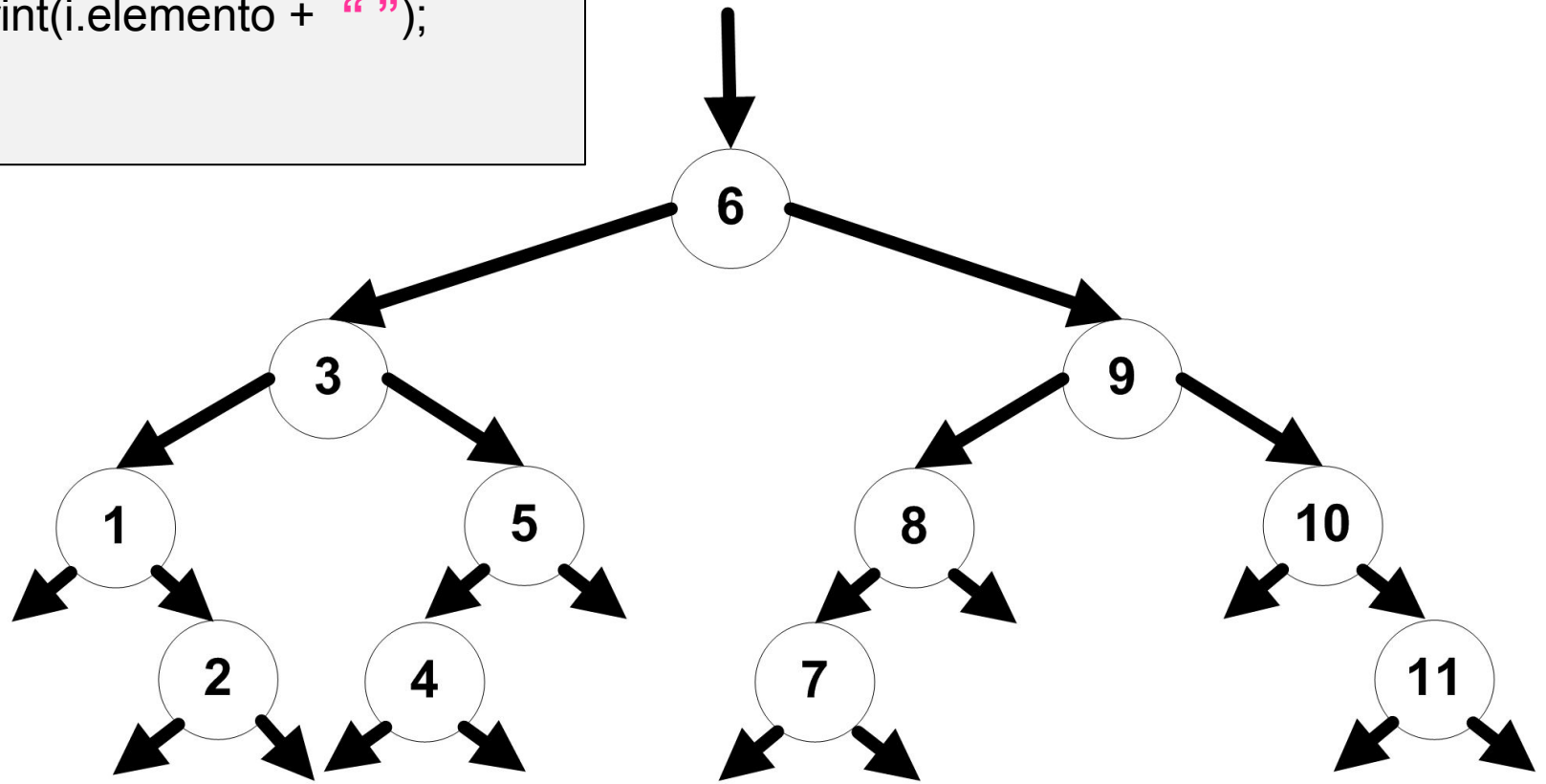
```
void caminharCentral(No i) {
   if (i != null) {
      caminharCentral(i.esq);
      System.out.print(i.elemento + " ");
      caminharCentral(i.dir);
   }
}
```



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

Caminhamento Pós-fixado ou Pós-ordem

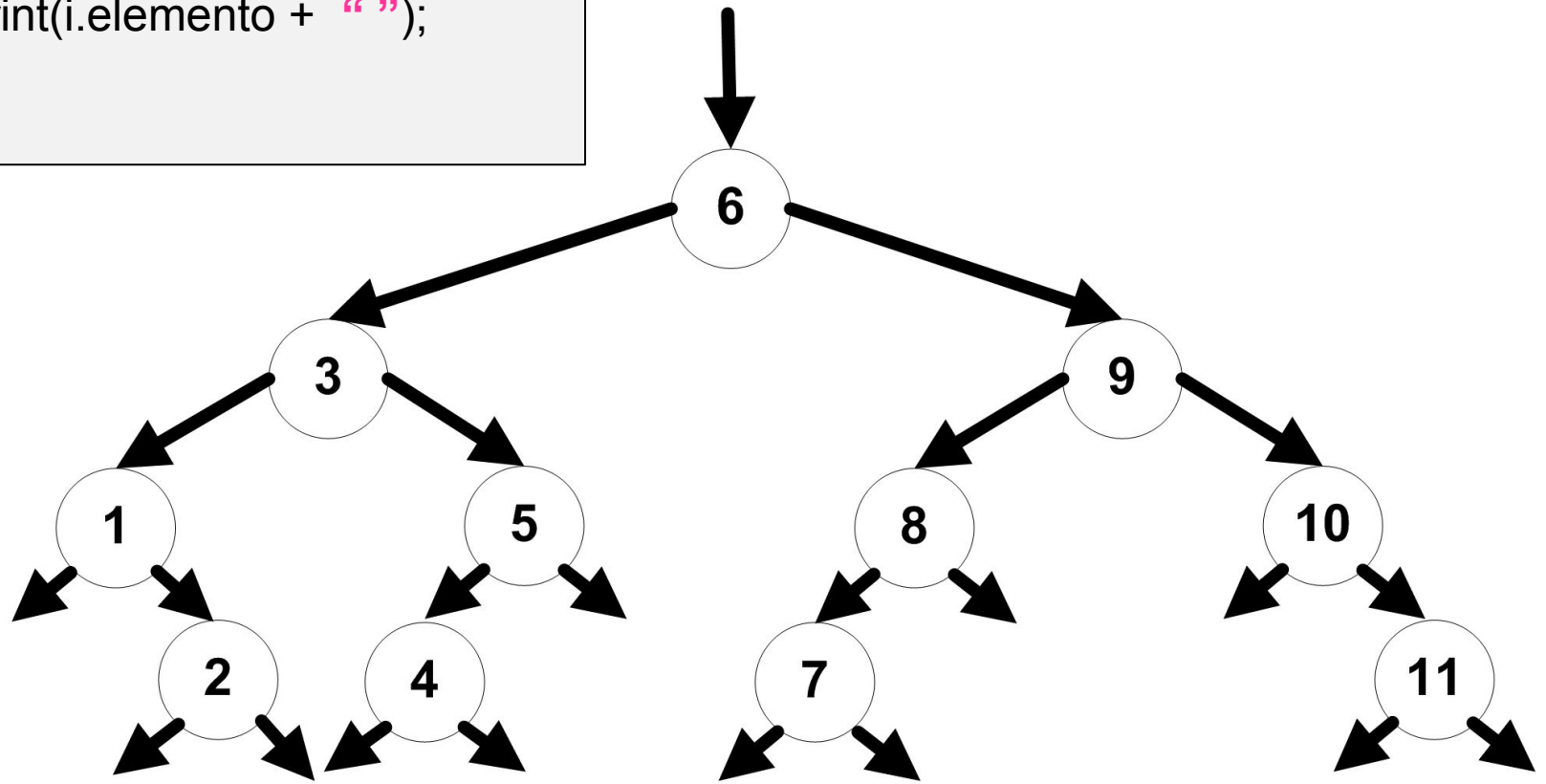
```
void caminharPos(No i) {
  if (i != null) {
    caminharPos(i.esq);
    caminharPos(i.dir);
    System.out.print(i.elemento + " ");
  }
}
```



Tela	

Caminhamento Pós-fixado ou Pós-ordem

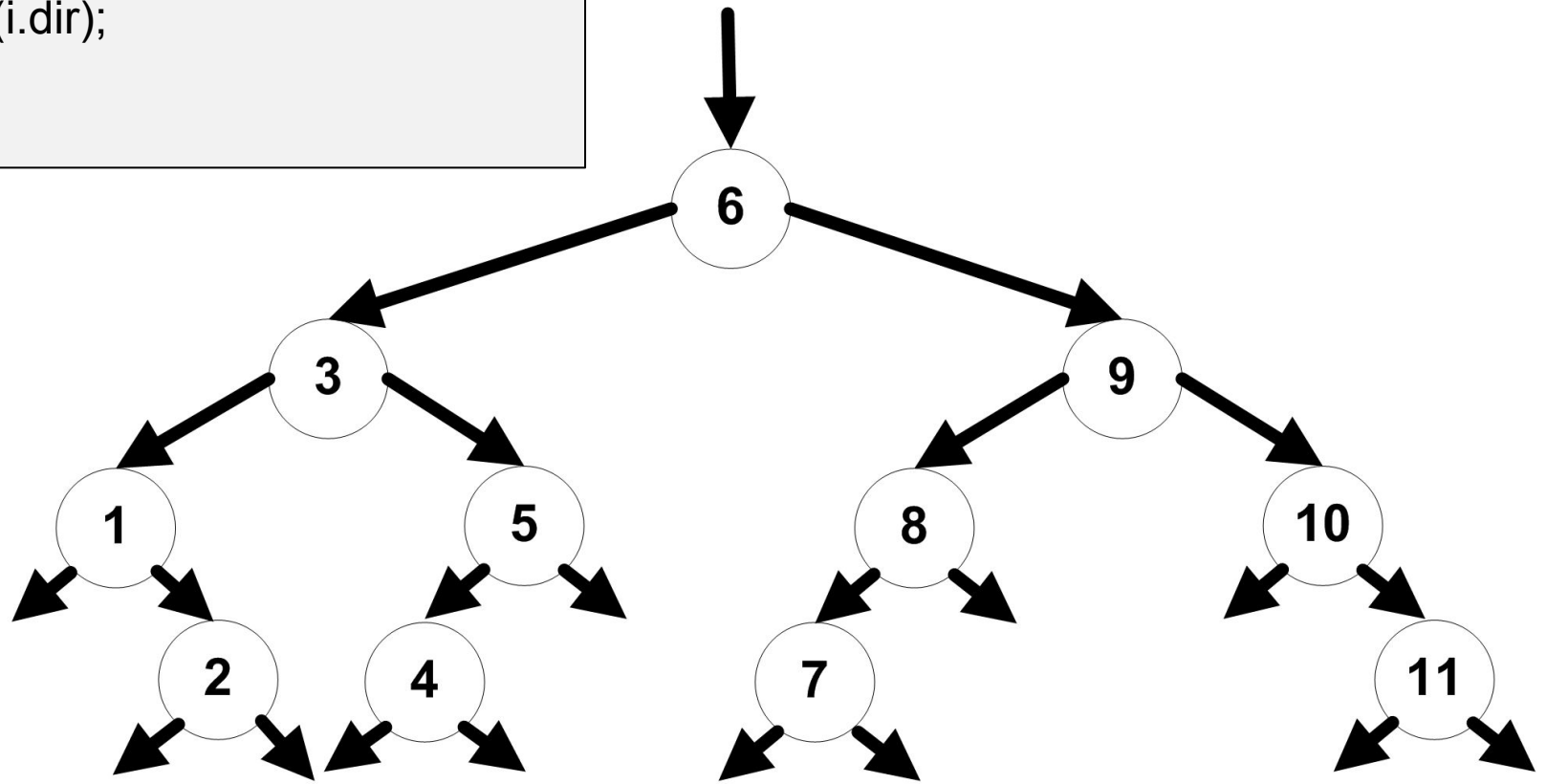
```
void caminharPos(No i) {
  if (i != null) {
    caminharPos(i.esq);
    caminharPos(i.dir);
    System.out.print(i.elemento + " ");
  }
}
```



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

Caminhamento Pré-fixado ou Pré-ordem

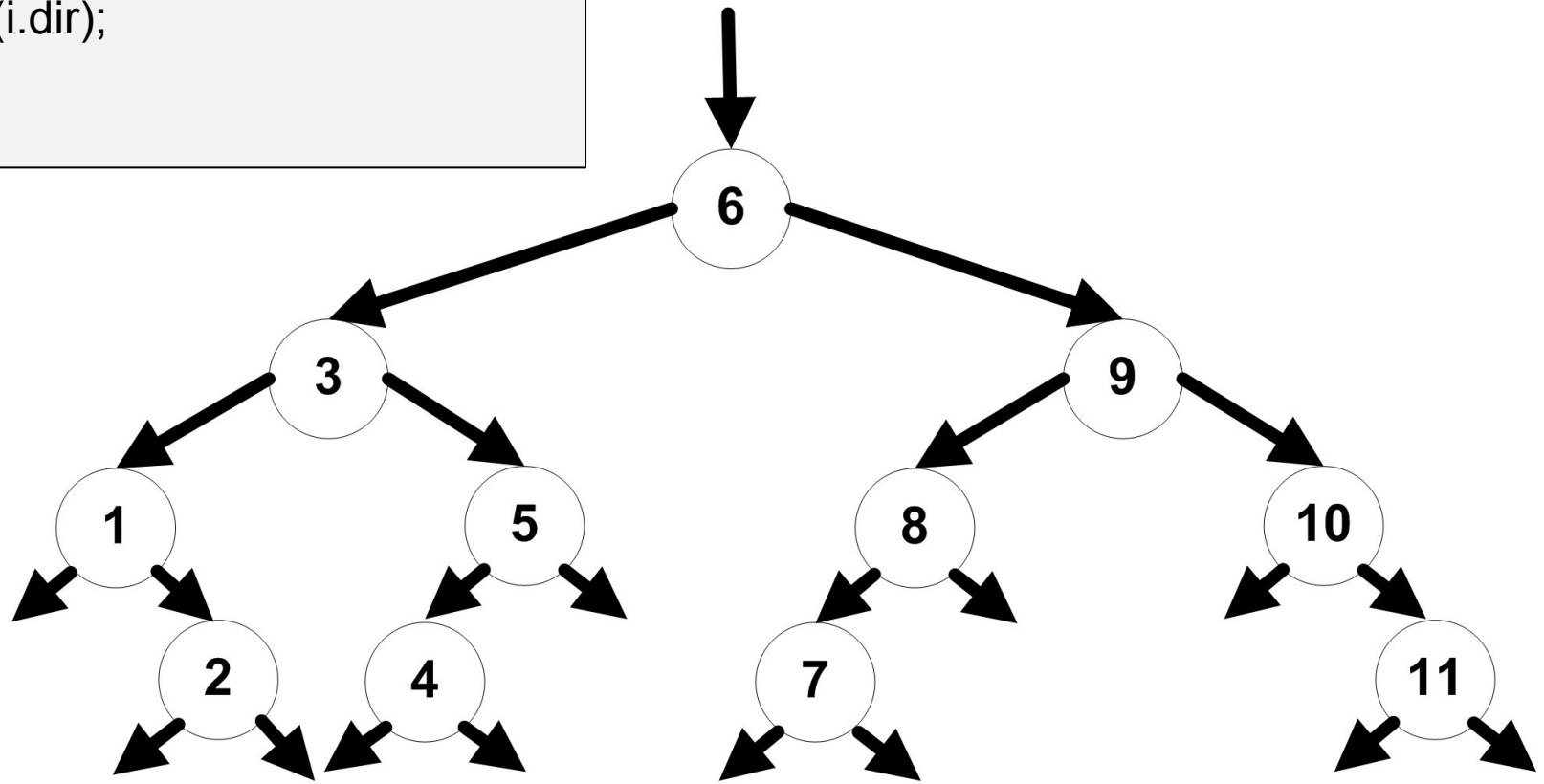
```
void caminharPre(No i) {
  if (i != null) {
    System.out.print(i.elemento + " ");
    caminharPre(i.esq);
    caminharPre(i.dir);
  }
}
```



Tela	

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

Exercício Resolvido (1)

- Faça um método que retorna a altura da árvore

Exercício Resolvido (1)

- Faça um método que retorna a altura da árvore

```
public int getAltura(No i, int altura){
    if(i == null){
        altura--;
    } else {
        int alturaEsq = getAltura(i.esq, altura + 1);
        int alturaDir = getAltura(i.dir, altura + 1);
        altura = (alturaEsq > alturaDir) ? alturaEsq : alturaDir;
    }
    return altura;
}
```

Exercício Resolvido (2)

- Insira 100000 elementos de forma aleatória. Para cada inserção, mostre na tela o número de elementos da árvore, o logaritmo (base 2) desse número e a altura da árvore

Exercício Resolvido (2)

- Insira 100000 elementos de forma aleatória. Para cada inserção, mostre na tela o número de elementos da árvore, o logaritmo (base 2) desse número e a altura da árvore

```
ArvoreBinaria a = new ArvoreBinaria();

Random gerador = new Random();
gerador.setSeed(0);
for(int i = 1; i <= 100000; i++){
   int valor;
   do {
      valor = Math.abs(gerador.nextInt());
   } while (a.pesquisar(valor) == true);

   a.inserir(valor);

   if(i % 1000 == 0){
      double log2 = (Math.log(i) / Math.log(2));
      log2 *= 1.39;
      System.out.println("Número de nós = " + i + " --- log(i,2) = " + log2 + " --- h = " + a.getAltura());
   }
}
```

Exercício Resolvido (3)

- Faça um método que retorne a soma dos elementos existentes na árvore

Exercício Resolvido (3)

- Faça um método que retorne a soma dos elementos existentes na árvore

```
public int soma(){
   return soma(raiz);
}

public int soma(No i){
   int resp = 0;
   if(i != null){
      resp = i.elemento + soma(i.esq) + soma(i.dir);
   }
   return resp;
}
```

Exercício Resolvido (4)

- Faça um método que retorne o número de elementos pares existentes na árvore

Exercício Resolvido (4)

- Faça um método que retorne o número de elementos pares existentes na árvore

```
public int numPares(){
   return numPares(raiz);
}

public int numPares(No i){
   int resp = 0;
   if(i != null){
      resp = ((i.elemento % 2 == 0) ? 1 : 0) + numPares(i.esq) + numPares(i.dir);
   }
   return resp;
}
```

Exercício Resolvido (5)

- Faça um método estático que recebe dois objetos do tipo árvore binária e retorne um booleano indicando se as duas árvores são iguais

Exercício Resolvido (5)

- Faça um método estático que recebe dois objetos do tipo árvore binária e retorne um booleano indicando se as duas árvores são iguais

```
public static boolean igual (ArvoreBinaria a1, ArvoreBinaria a2){
   return igual(a1.raiz, a2.raiz);
}

private static boolean igual (No i1, No i2){
   boolean resp;
   if(i1 != null && i2 != null){
      resp = (i1.elemento == i2.elemento) && igual(i1.esq, i2.esq) && igual(i1.dir, i2.dir);
   } else if(i1 == null && i2 == null){
      resp = true;
   } else {
      resp = false;
   }
   return resp;
}
```

Exercício Resolvido (6)

- Faça um método que retorna ***true*** se a árvore contém algum número divisível por onze

Exercício Resolvido (6)

- Faça um método que retorna ***true*** se a árvore contém algum número divisível por onze

```
public boolean hasDiv11(){
   return hasDiv11(raiz);
}

public boolean hasDiv11(No i){
   boolean resp = false;
   if(i != null){
      resp = (i.elemento % 11 == 0) || hasDiv11(i.esq) || hasDiv11(i.dir);
   }
   return resp;
}
```

Exercício Resolvido (7)

- Um algoritmo de ordenação é o *TreeSort* que insere os elementos do *array* em uma árvore binária e utiliza um "caminhar" para ordenar os elementos do *array*. Implemente o *TreeSort* e faça a análise de complexidade do mesmo

Exercício Resolvido (7)

- U*eeSort* que insere os elementos do *array* er e Caminhamento e utiliza um "caminhar" pa*y*. Implemente o *TreeSort* e faça a análise de

```
public class TreeSort {
   private No raiz;
   private int n;

   public TreeSort() {
      raiz = null;
      n = 0;
   }

   public int[] sort() {
      int[] array = new int[n];
      n = 0;
      sort(raiz, array);
      return array;
   }

   private void sort(No i, int[] array) {
      if (i != null) {
         sort(i.esq, array);
         array[n++] = i.elemento;
         sort (i.dir, array);
      }
   }

   public void inserir(int x) {
      //...
   }
}
```

Supondo que a árvore é balanceada, o custo de inserção é $\Theta(\lg(n))$ comparações. O custo de inserção dos n elementos será $\Theta(n \times \lg(n))$ comparações

Exercício Resolvido (8)

- Faça o método ***No toAB(Celula p1, CelulaDupla p2)*** que recebe o nó cabeça de uma lista simples e o de outra dupla. Em seguida, crie uma árvore binária contendo os elementos intercalados das duas listas e retorne o endereço do nó raiz da árvore criada

Exercício Resolvido (8)

```
No toAB(Celula p1, CelulaDupla p2){
   No resp = null;
   p1 = p1.prox;
   p2 = p2.prox;
   while(p1 != null && p2 != null){
      resp = inserir(resp, p1.elemento);
      resp = inserir(resp, p2.elemento);
      p1 = p1.prox;
      p2 = p2.prox;
   }
   while(p1 != null){
      resp = inserir(resp, p1.elemento);
      p1 = p1.prox;
   }
   while(p2 != null){
      resp = inserir(resp, p2.elemento);
      p2 = p2.prox;
   }
   return resp;
}
```

laDupla p2) que recebe o nó

upla. Em seguida, crie uma

calados das duas listas e retorne

```
No inserir(No i, int x) 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 ao inserir!");
   }
   return i;
}
```