

# Unidade VI: Balanceamento de Árvores Binárias



Instituto de Ciências Exatas e Informática  
Departamento de Ciência da Computação

# Balanceamento de Árvores

- Qual é a vantagem de uma árvore balanceada?

# Balanceamento de Árvores

- Qual é a vantagem de uma árvore balanceada?
- Resposta: eficiência em termos de pesquisa, inserção e remoção
- Inicialmente, todas árvores são balanceadas e elas podem desbalancear após as operações de inserção e remoção

# Ideia Básica do Balanceamento de Árvores

- As árvores desbalanceadas para a esquerda devem ser rotacionadas para a direita e as para a direita, para a esquerda

# Tipos de Rotação

- Rotação simples à esquerda
- Rotação simples à direita
- Rotação dupla direita - esquerda
- Rotação dupla esquerda - direita

# Tipos de Rotação

- **Rotação simples à esquerda** 

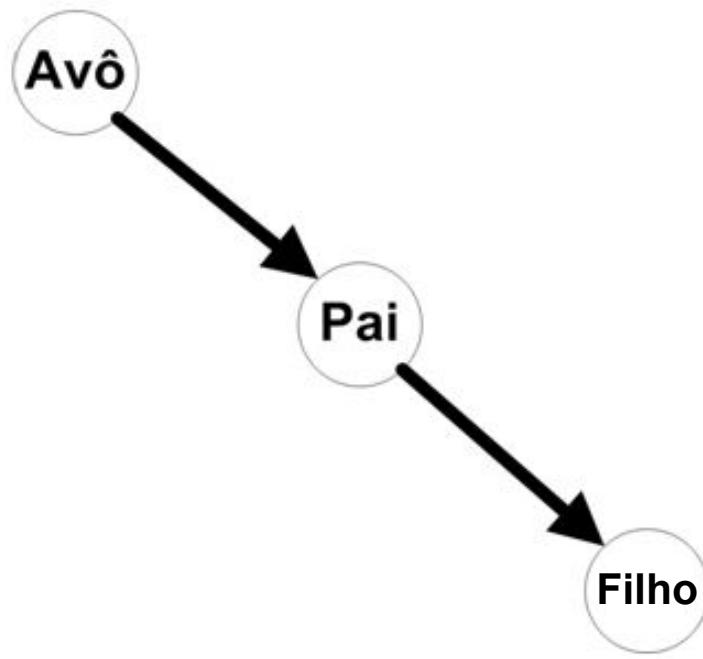
- Rotação simples à direita

- Rotação dupla direita - esquerda

- Rotação dupla esquerda - direita

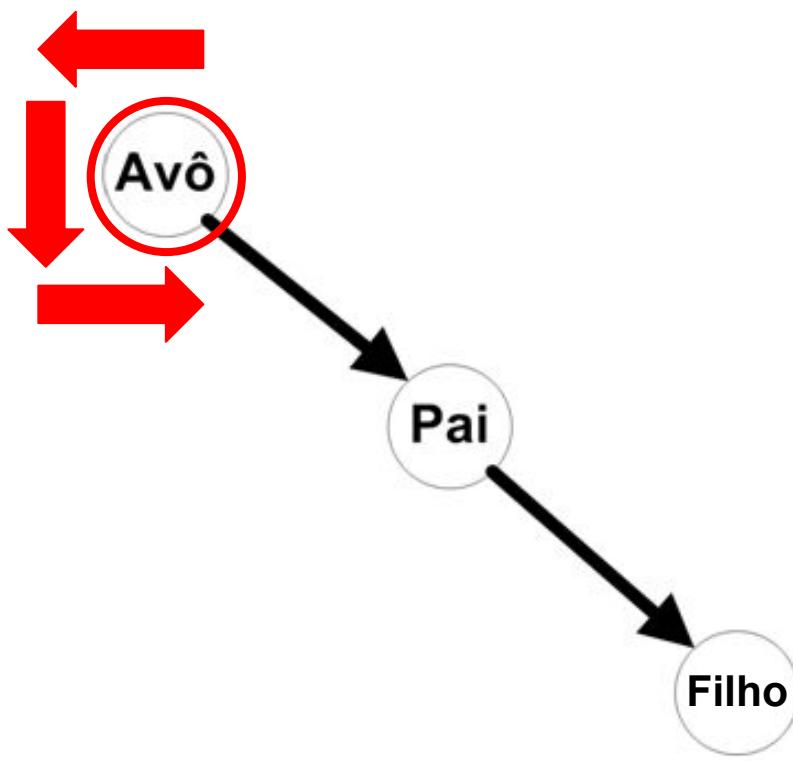
# Rotação Simples à Esquerda

- Usada em subárvores em que o pai e o filho estão desbalanceados para a direita



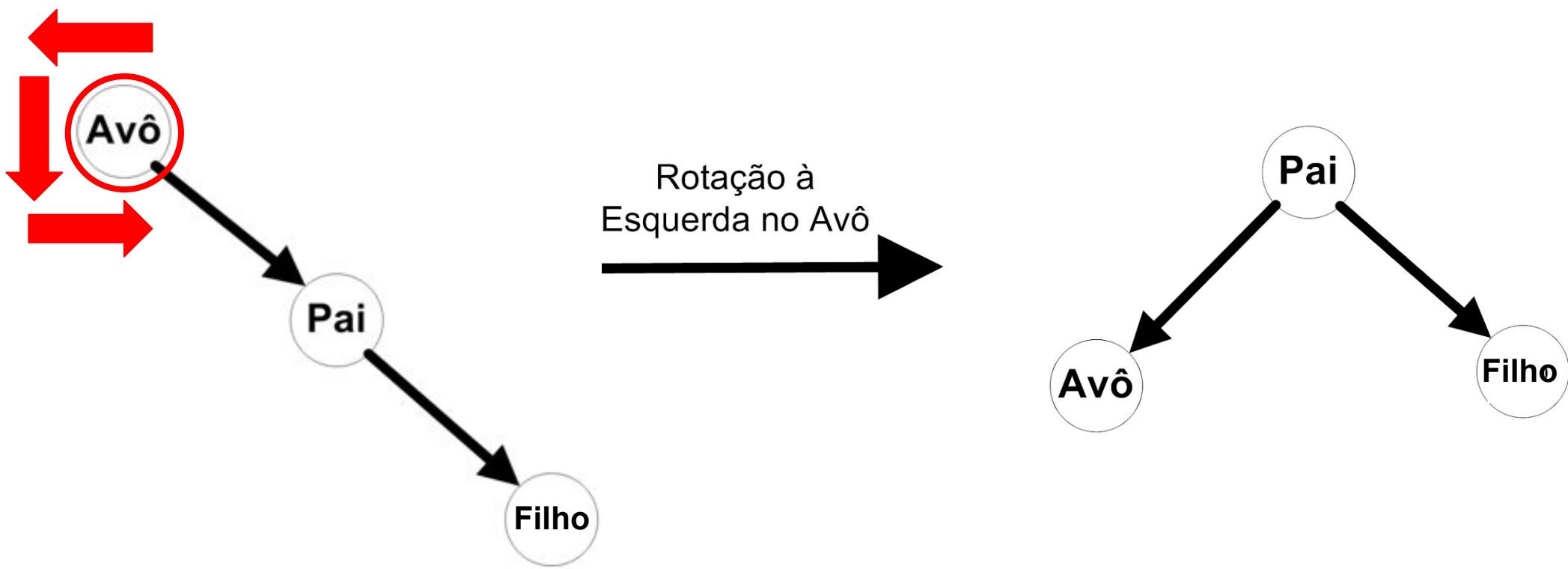
# Rotação Simples à Esquerda

- Usada em subárvores em que o pai e o filho estão desbalanceados para a direita

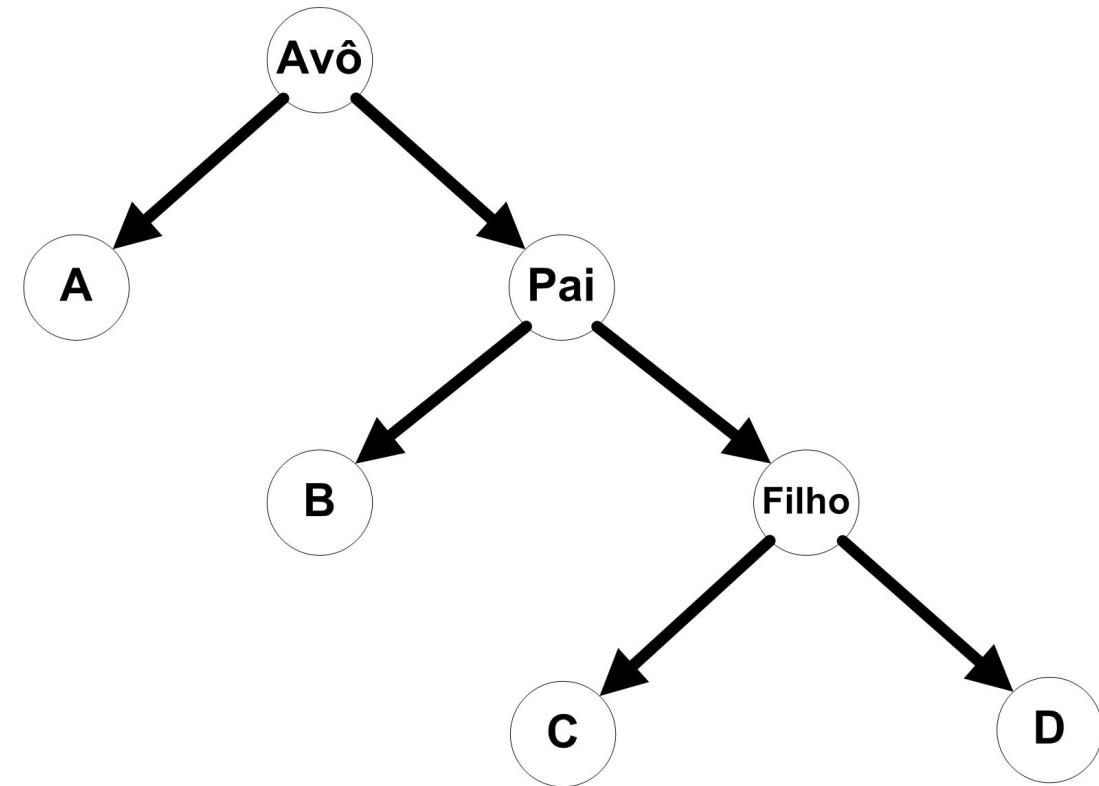


# Rotação Simples à Esquerda

- Usada em subárvores em que o pai e o filho estão desbalanceados para a direita

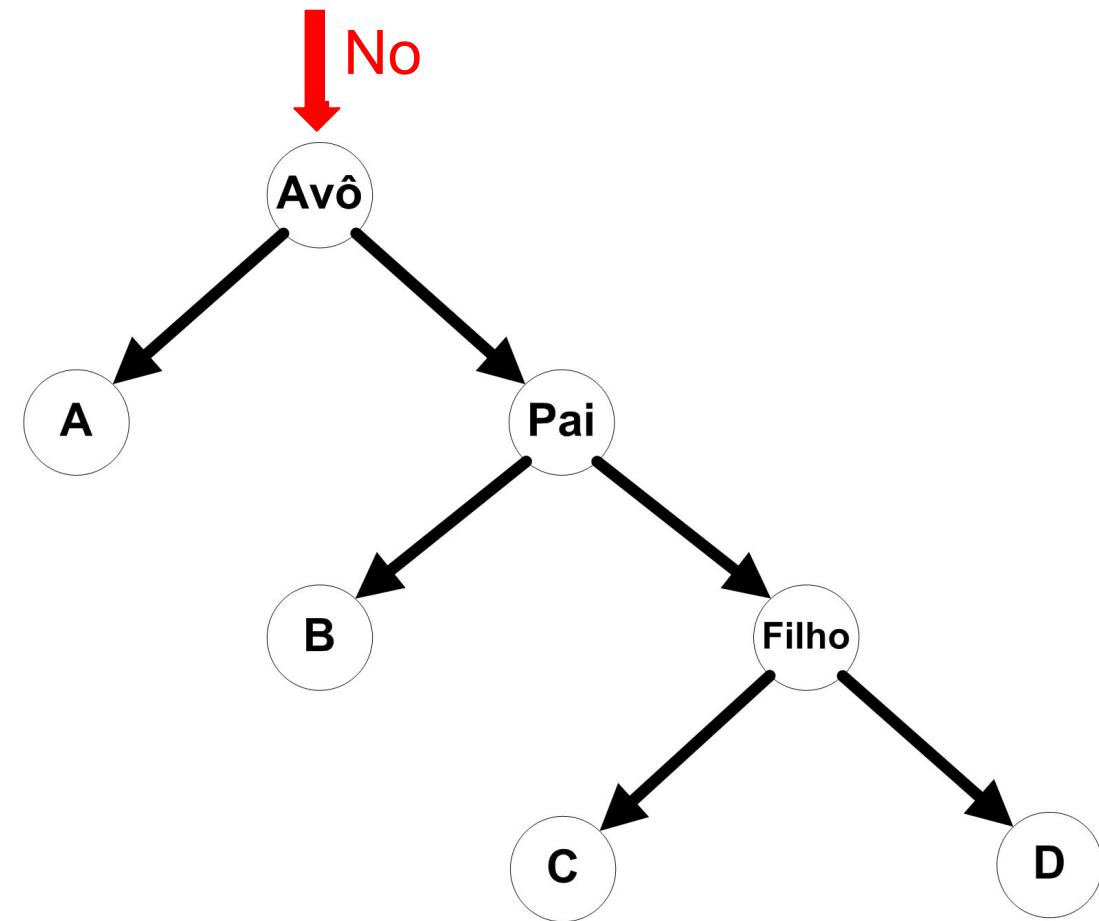


# Implementação da Rotação à Esquerda



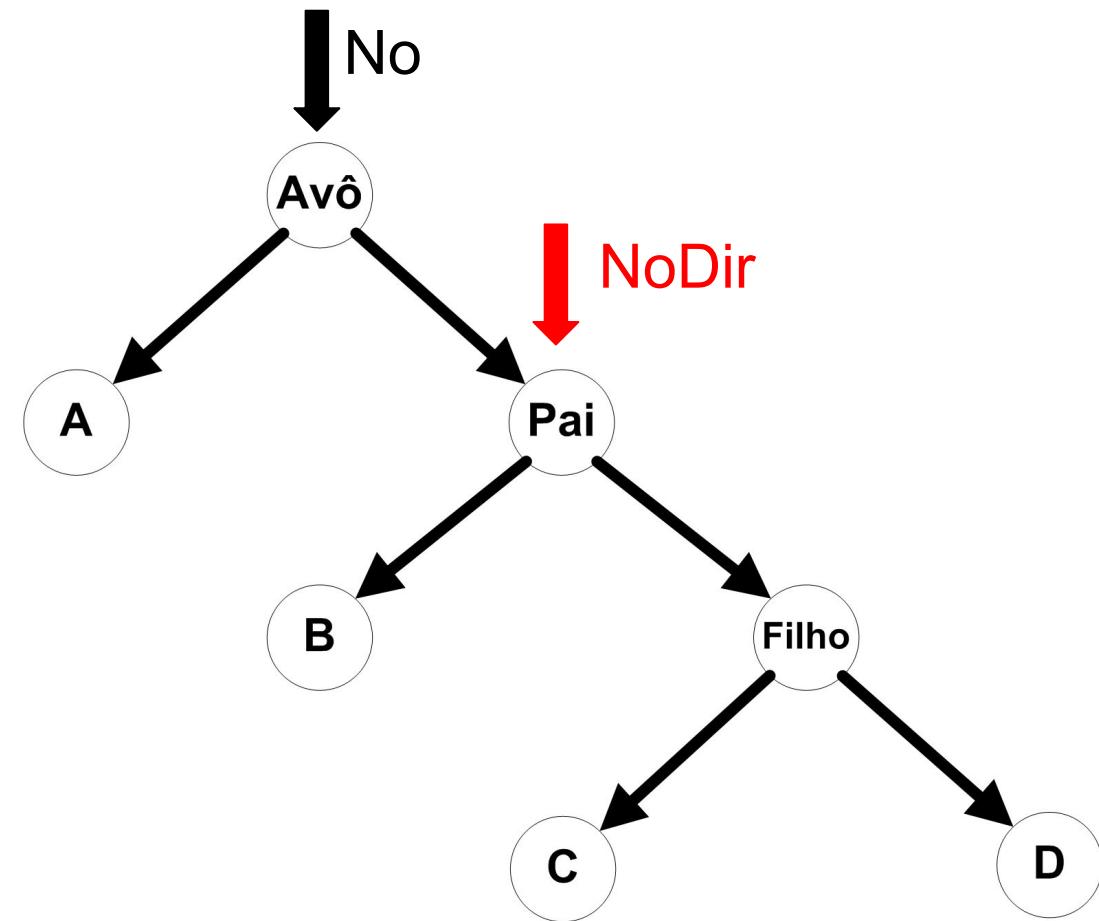
```
void metodo(){  
    ...  
    no = rotacionarEsq(no);  
    ...  
}  
No rotacionarEsq (No no) {  
    No noDir = no.dir;  
    No noDirEsq = noDir.esq;  
  
    noDir.esq = no;  
    no.dir = noDirEsq;  
  
    return noDir;  
}
```

## Implementação da Rotação à Esquerda



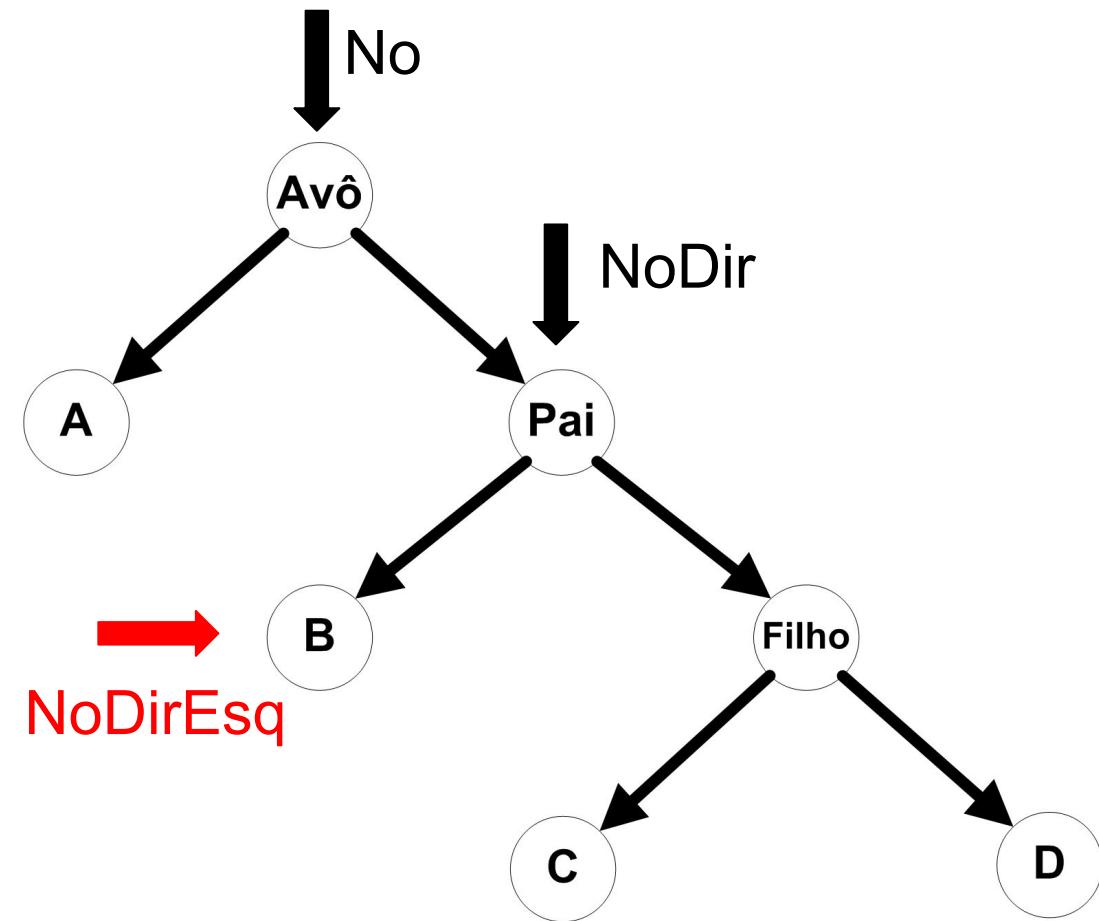
```
void metodo(){  
    ...  
    no = rotacionarEsq(no);  
    ...  
}  
No rotacionarEsq (No no) {  
    No noDir = no.dir;  
    No noDirEsq = noDir.esq;  
  
    noDir.esq = no;  
    no.dir = noDirEsq;  
  
    return noDir;  
}
```

## Implementação da Rotação à Esquerda



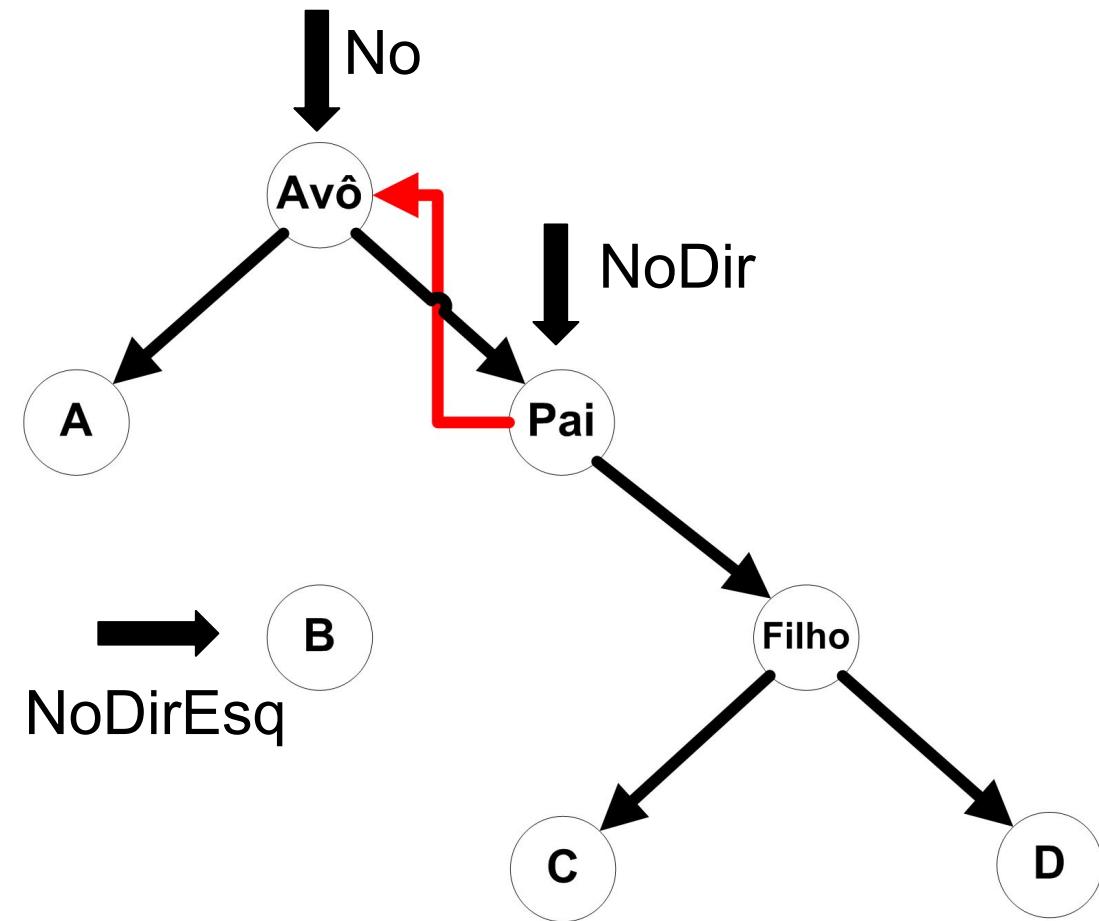
```
void metodo(){  
    ...  
    no = rotacionarEsq(no);  
    ...  
}  
NoNo rotacionarEsq (No no) {  
    No noDir = no.dir;  
    No noDirEsq = noDir.esq;  
  
    noDir.esq = no;  
    no.dir = noDirEsq;  
  
    return noDir;  
}
```

## Implementação da Rotação à Esquerda



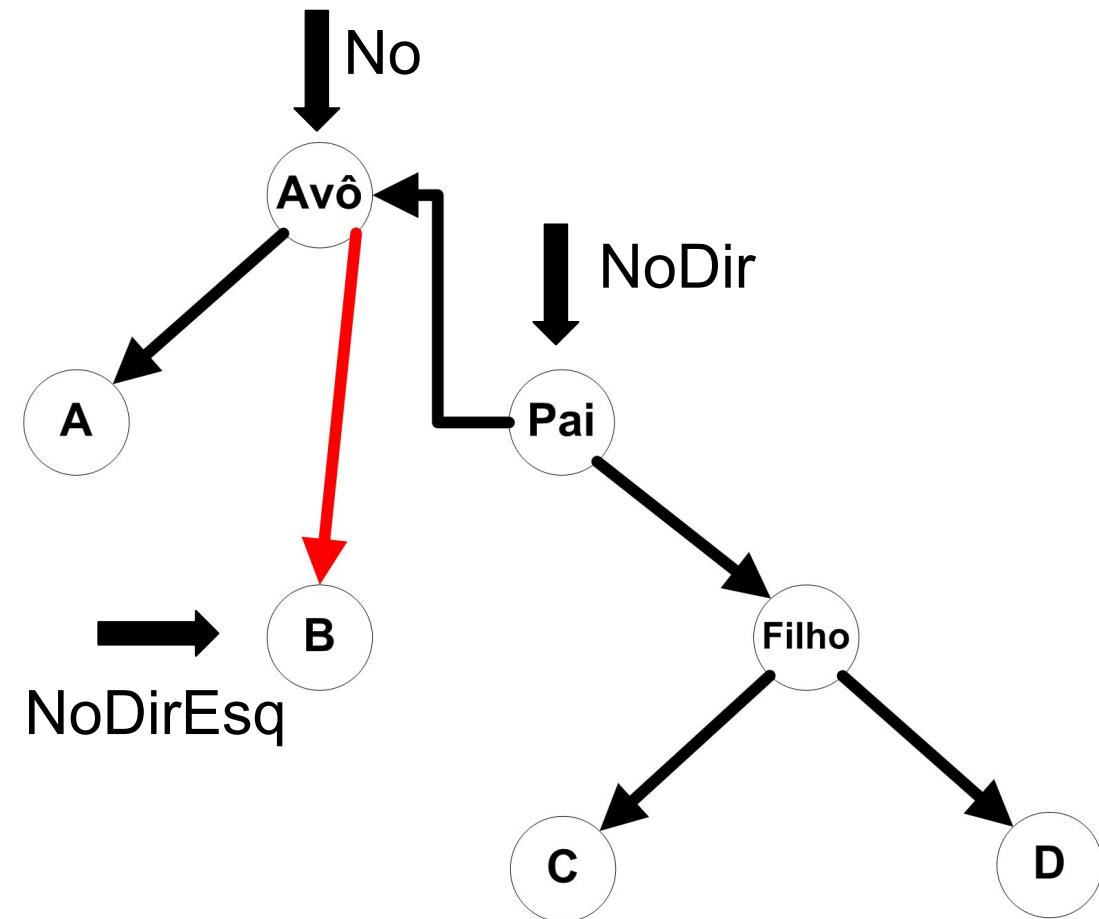
```
void metodo(){  
    ...  
    no = rotacionarEsq(no);  
    ...  
}  
NoNo rotacionarEsq (No no) {  
    No noDir = no.dir;  
    No noDirEsq = noDir.esq;  
  
    noDir.esq = no;  
    no.dir = noDirEsq;  
  
    return noDir;  
}
```

## Implementação da Rotação à Esquerda



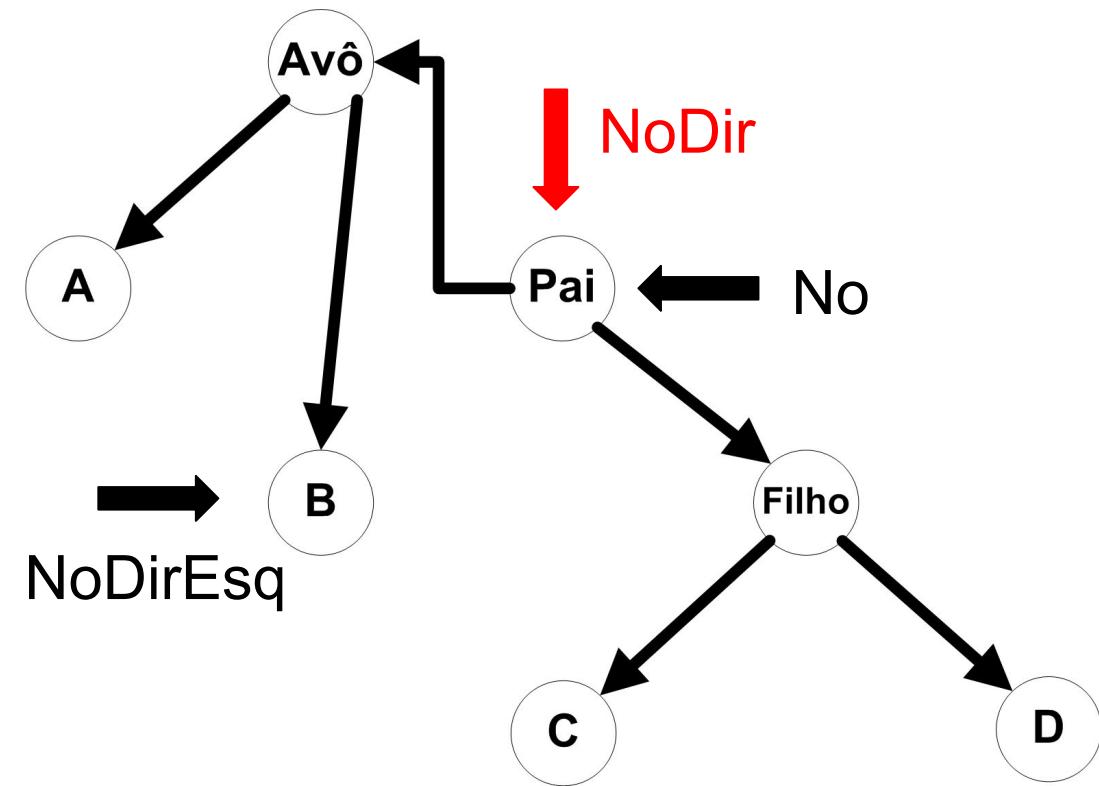
```
void metodo(){  
    ...  
    no = rotacionarEsq(no);  
    ...  
}  
No rotacionarEsq (No no) {  
    No noDir = no.dir;  
    No noDirEsq = noDir.esq;  
  
    noDir.esq = no;  
    no.dir = noDirEsq;  
  
    return noDir;  
}
```

# Implementação da Rotação à Esquerda



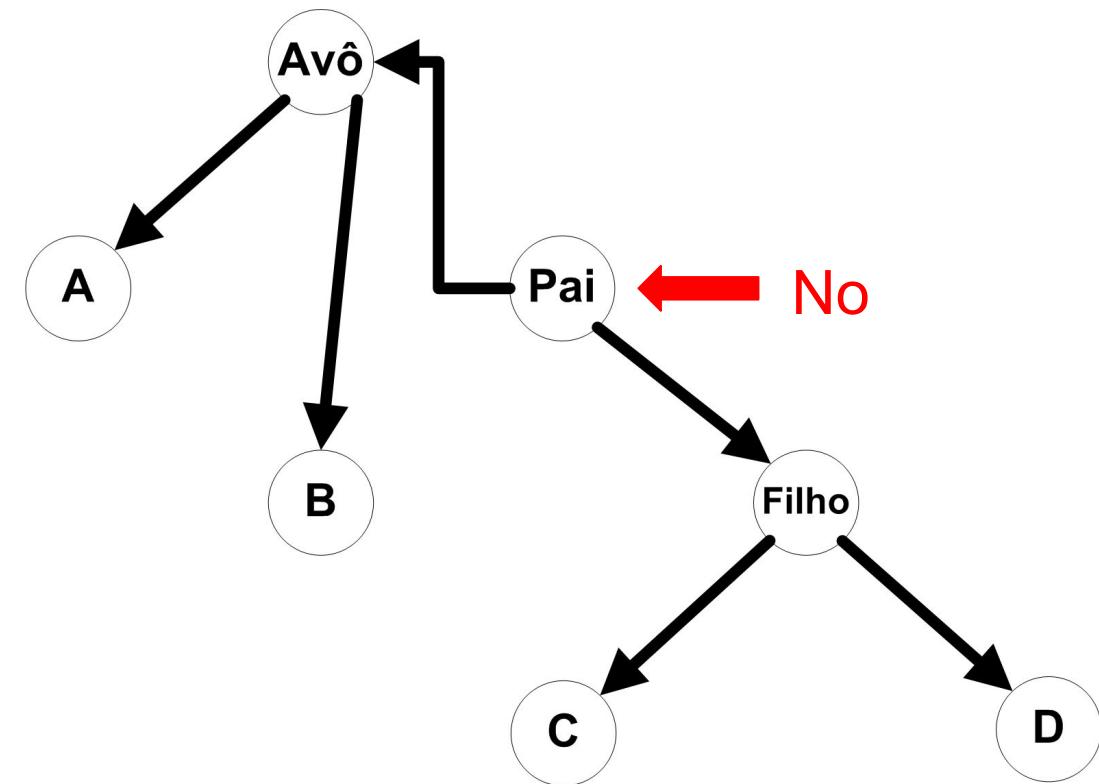
```
void metodo(){  
    ...  
    no = rotacionarEsq(no);  
    ...  
}  
No rotacionarEsq (No no) {  
    No noDir = no.dir;  
    No noDirEsq = noDir.esq;  
  
    noDir.esq = no;  
    no.dir = noDirEsq;  
  
    return noDir;  
}
```

## Implementação da Rotação à Esquerda



```
void metodo(){  
    ...  
    no = rotacionarEsq(no);  
    ...  
}  
No rotacionarEsq (No no) {  
    No noDir = no.dir;  
    No noDirEsq = noDir.esq;  
  
    noDir.esq = no;  
    no.dir = noDirEsq;  
  
    return noDir;  
}
```

## Implementação da Rotação à Esquerda



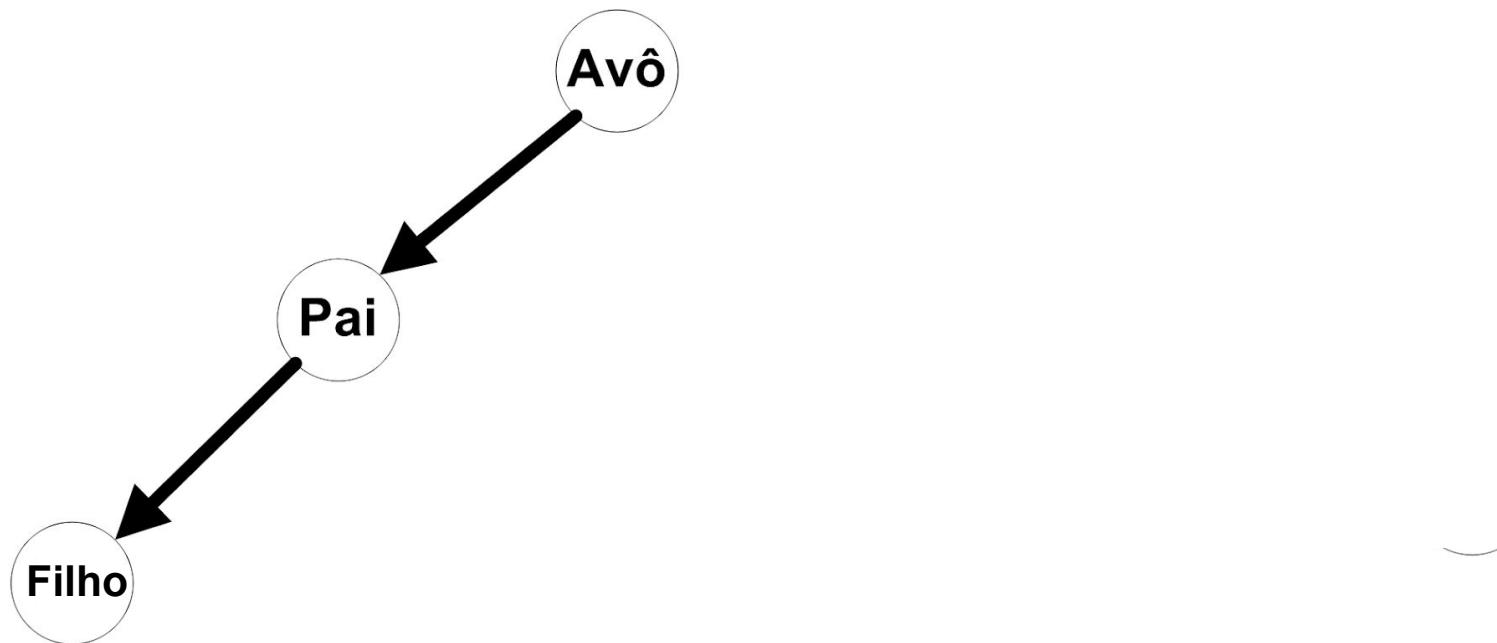
```
void metodo(){  
    ...  
    no = rotacionarEsq(no);  
    ...  
}  
No rotacionarEsq (No no) {  
    No noDir = no.dir;  
    No noDirEsq = noDir.esq;  
  
    noDir.esq = no;  
    no.dir = noDirEsq;  
  
    return noDir;  
}
```

# Tipos de Rotação

- Rotação simples à esquerda
- **Rotação simples à direita** 
- Rotação dupla direita - esquerda
- Rotação dupla esquerda - direita

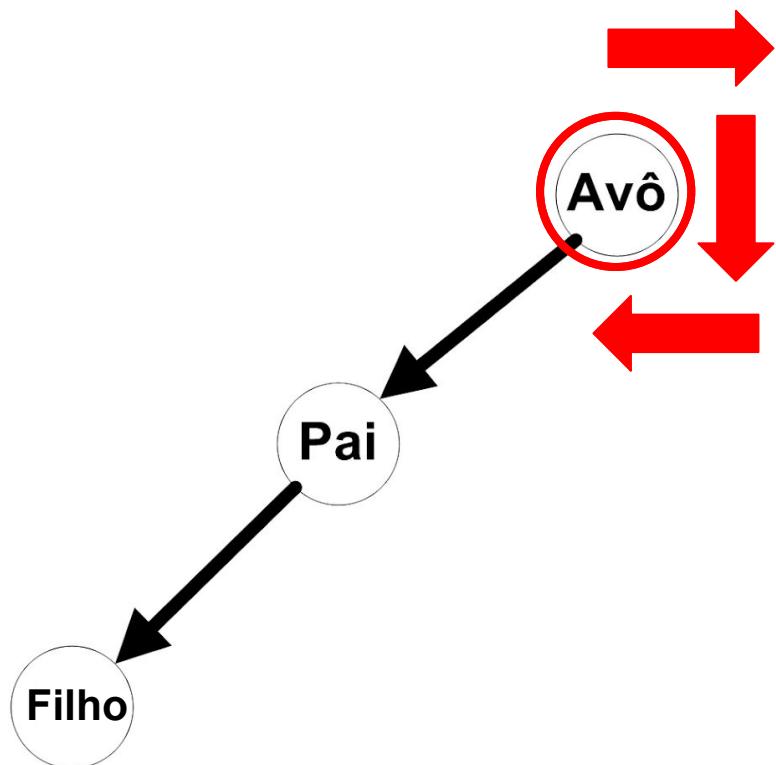
# Rotação Simples à Direita

- Usada em subárvores em que o pai e o filho estão desbalanceados para a esquerda



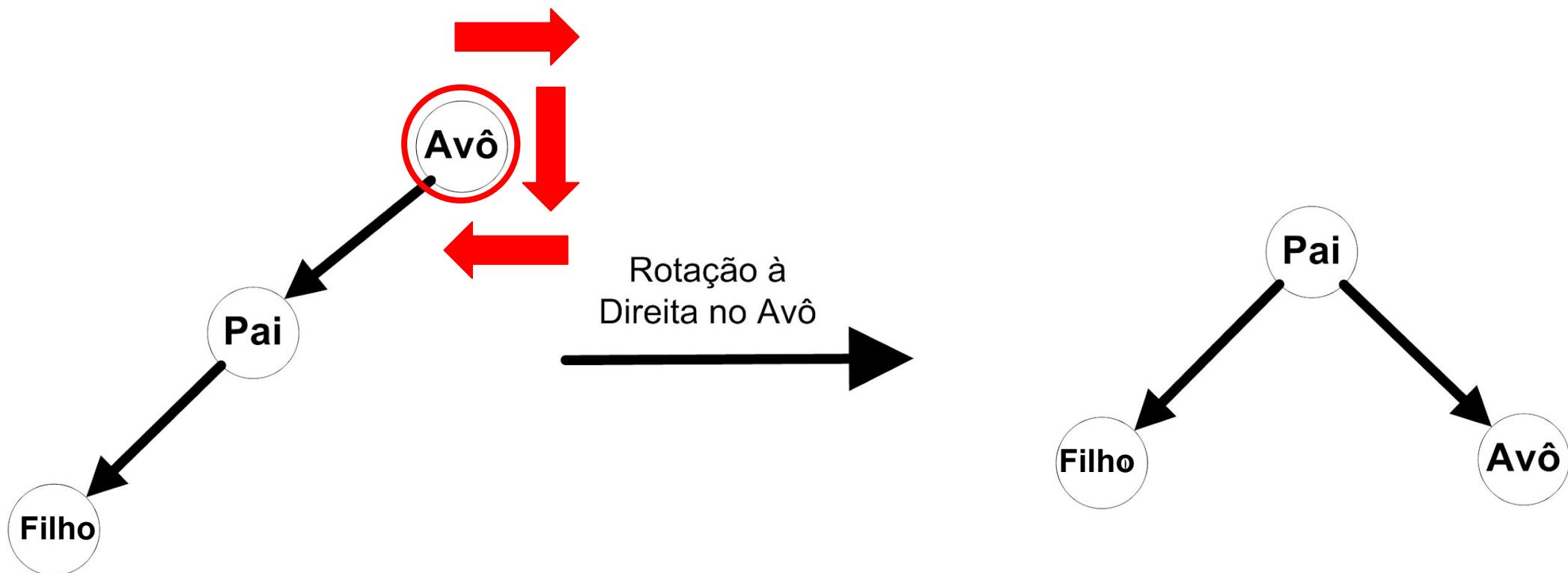
# Rotação Simples à Direita

- Usada em subárvores em que o pai e o filho estão desbalanceados para a esquerda

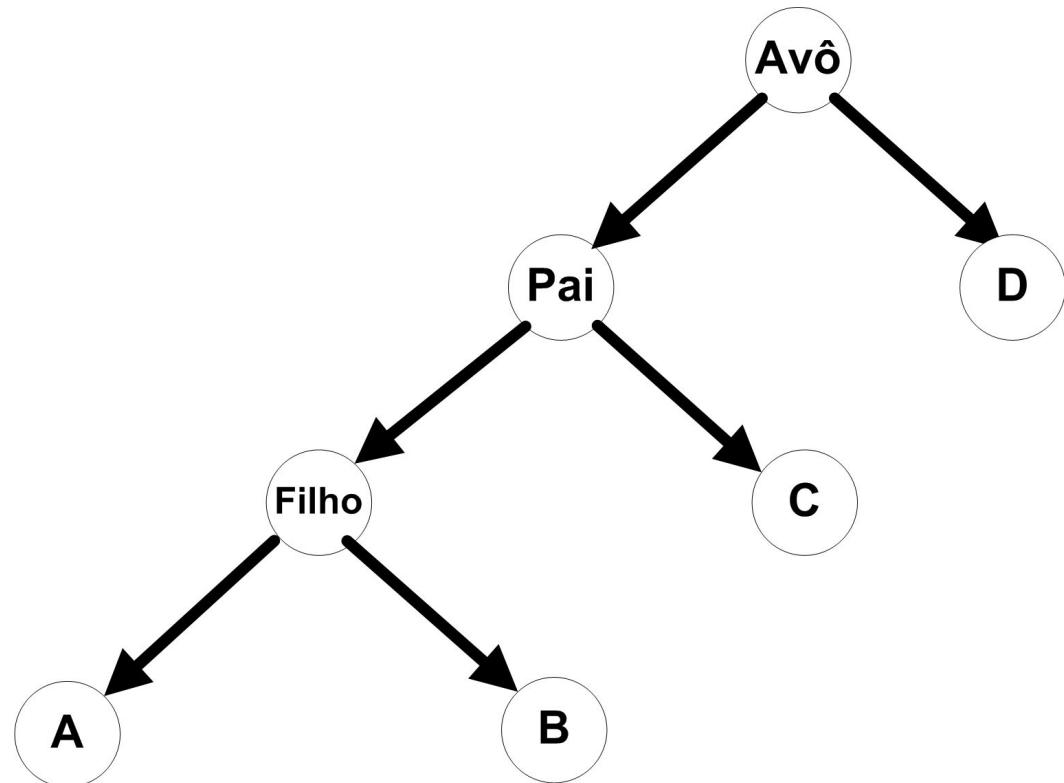


# Rotação Simples à Direita

- Usada em subárvores em que o pai e o filho estão desbalanceados para a esquerda

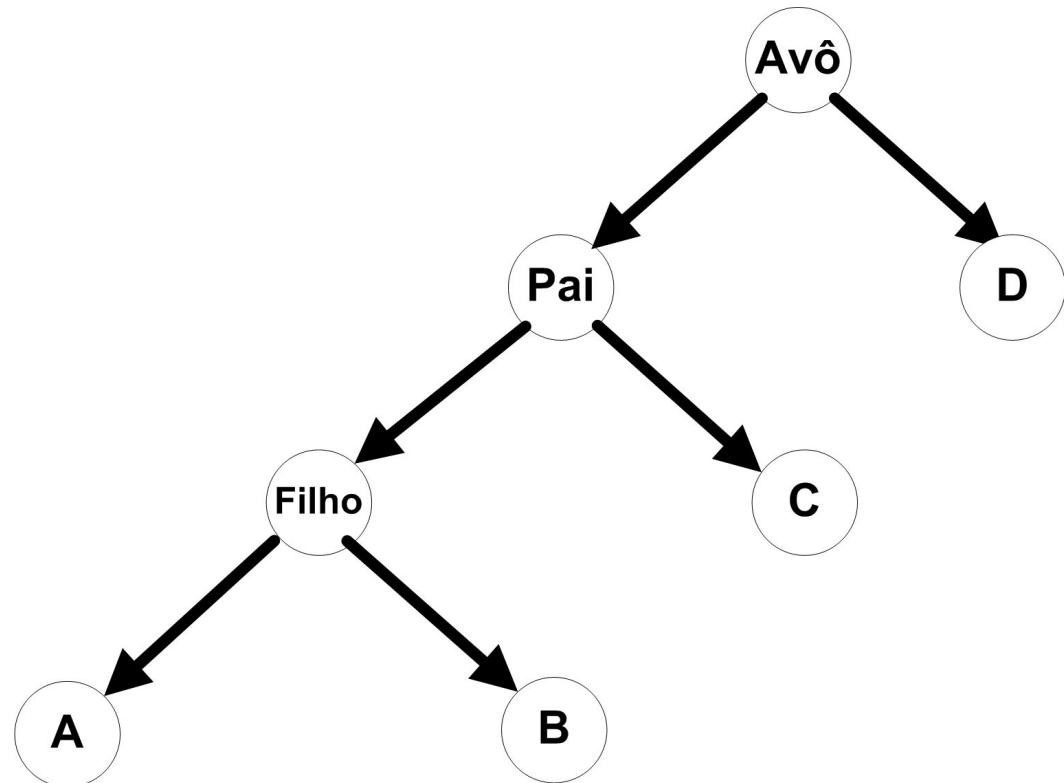


# Implementação da Rotação à Direita



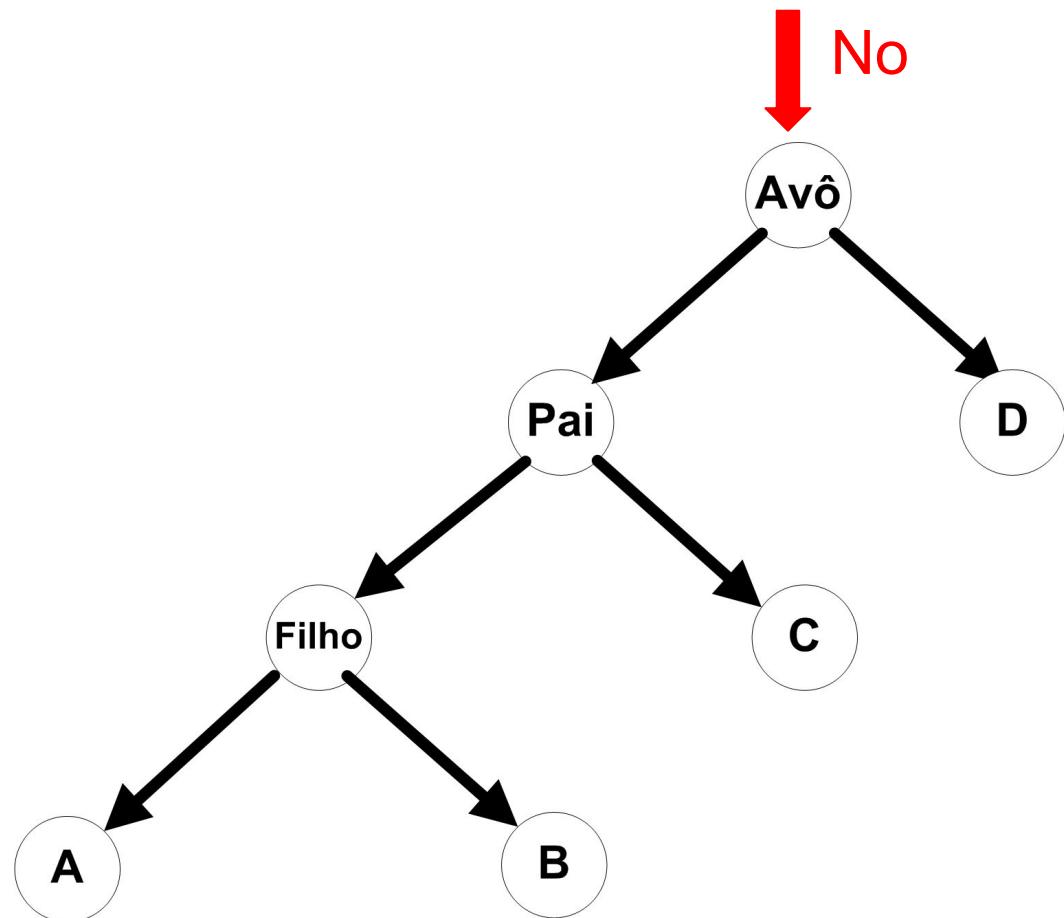
```
void metodo(){  
    ...  
    no = rotacionarDir(no);  
    ...  
}  
No rotacionarDir (No no) {  
  
    //Exercício: Implemente  
    //este método!!!  
}
```

## Implementação da Rotação à Direita



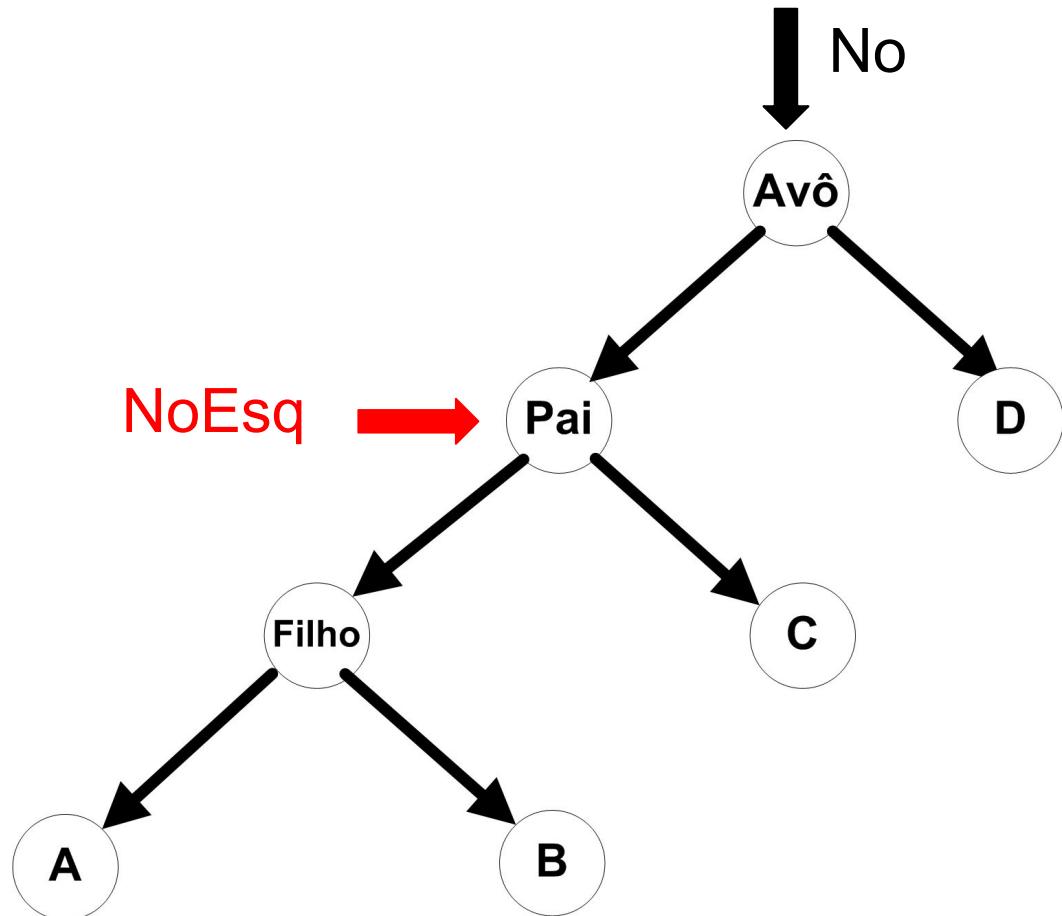
```
void metodo(){  
    ...  
    no = rotacionarDir(no);  
    ...  
}  
No rotacionarDir (No no) {  
    No noEsq = no.esq;  
    No noEsqDir = noEsq.dir;  
  
    noEsq.dir = no;  
    no.esq = noEsqDir;  
  
    return noEsq;  
}
```

## Implementação da Rotação à Direita



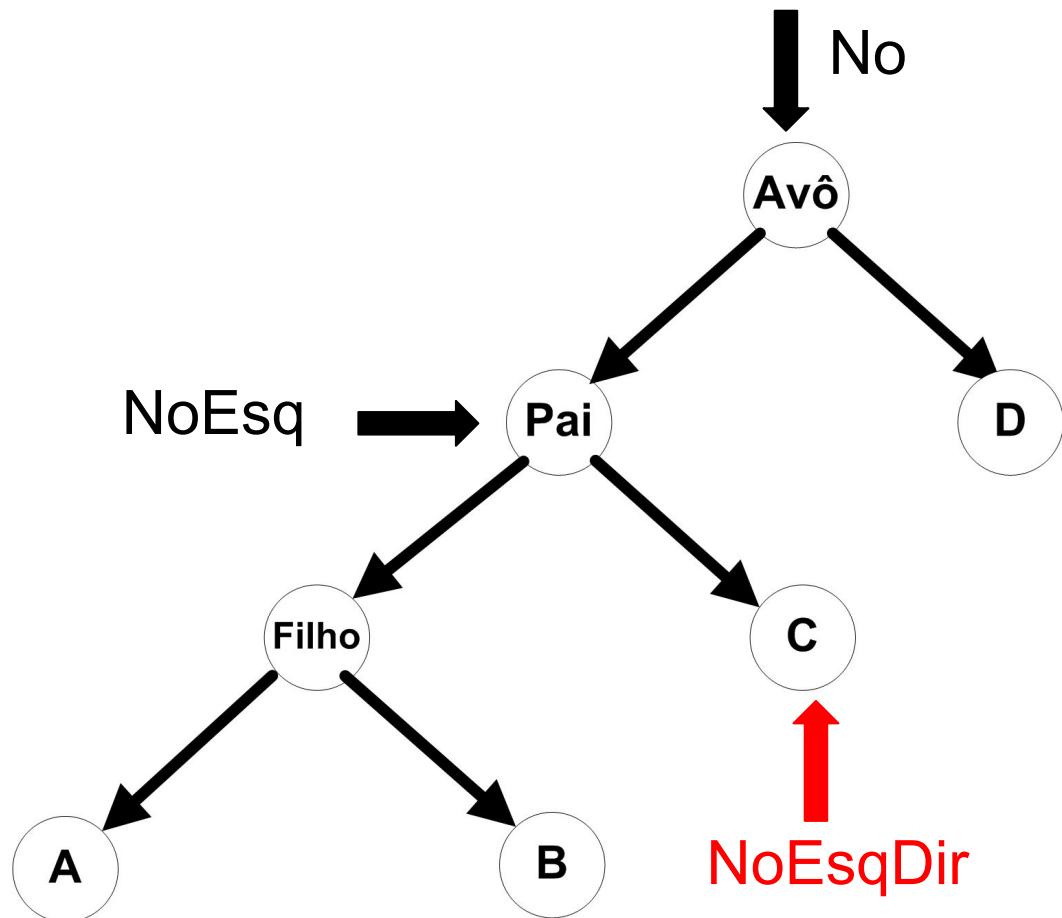
```
void metodo(){  
    ...  
    no = rotacionarDir(no);  
    ...  
}  
No rotacionarDir (No no) {  
    No noEsq = no.esq;  
    No noEsqDir = noEsq.dir;  
  
    noEsq.dir = no;  
    no.esq = noEsqDir;  
  
    return noEsq;  
}
```

## Implementação da Rotação à Direita



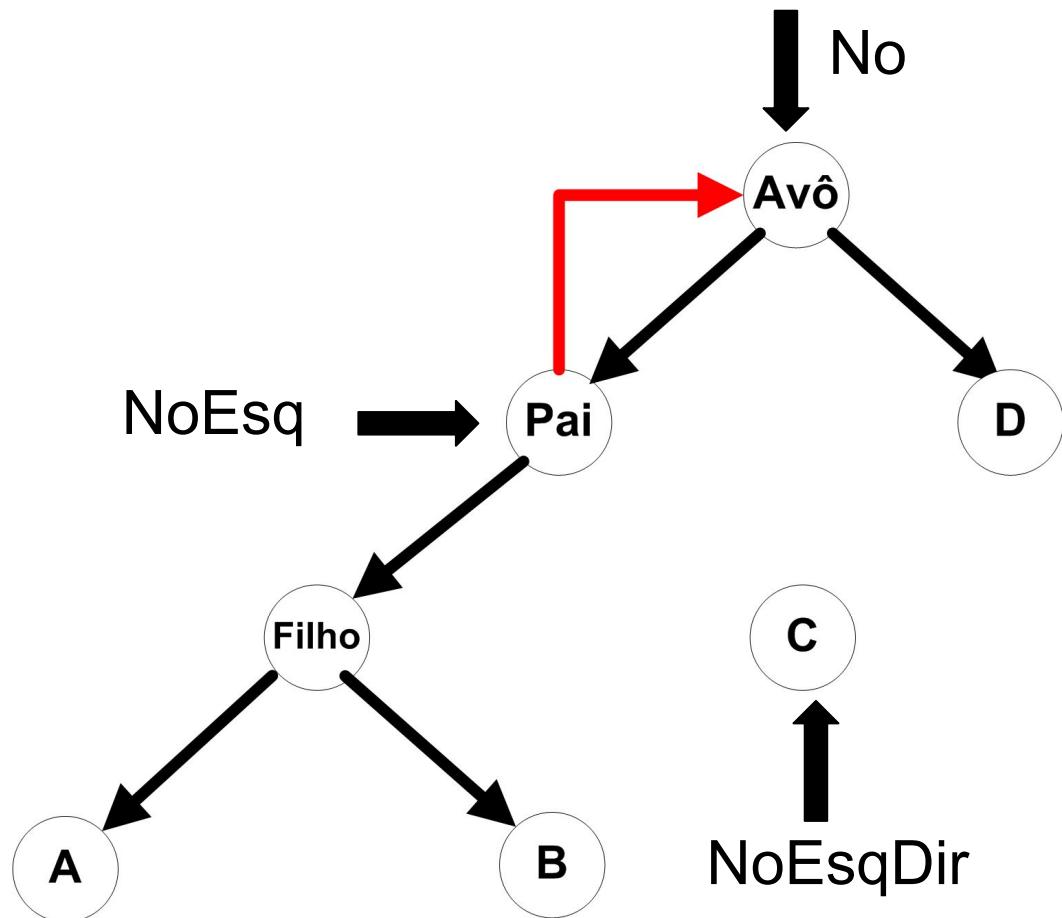
```
void metodo(){  
    ...  
    no = rotacionarDir(no);  
    ...  
}  
No rotacionarDir (No no) {  
    No noEsq = no.esq;  
    No noEsqDir = noEsq.dir;  
  
    noEsq.dir = no;  
    no.esq = noEsqDir;  
  
    return noEsq;  
}
```

## Implementação da Rotação à Direita



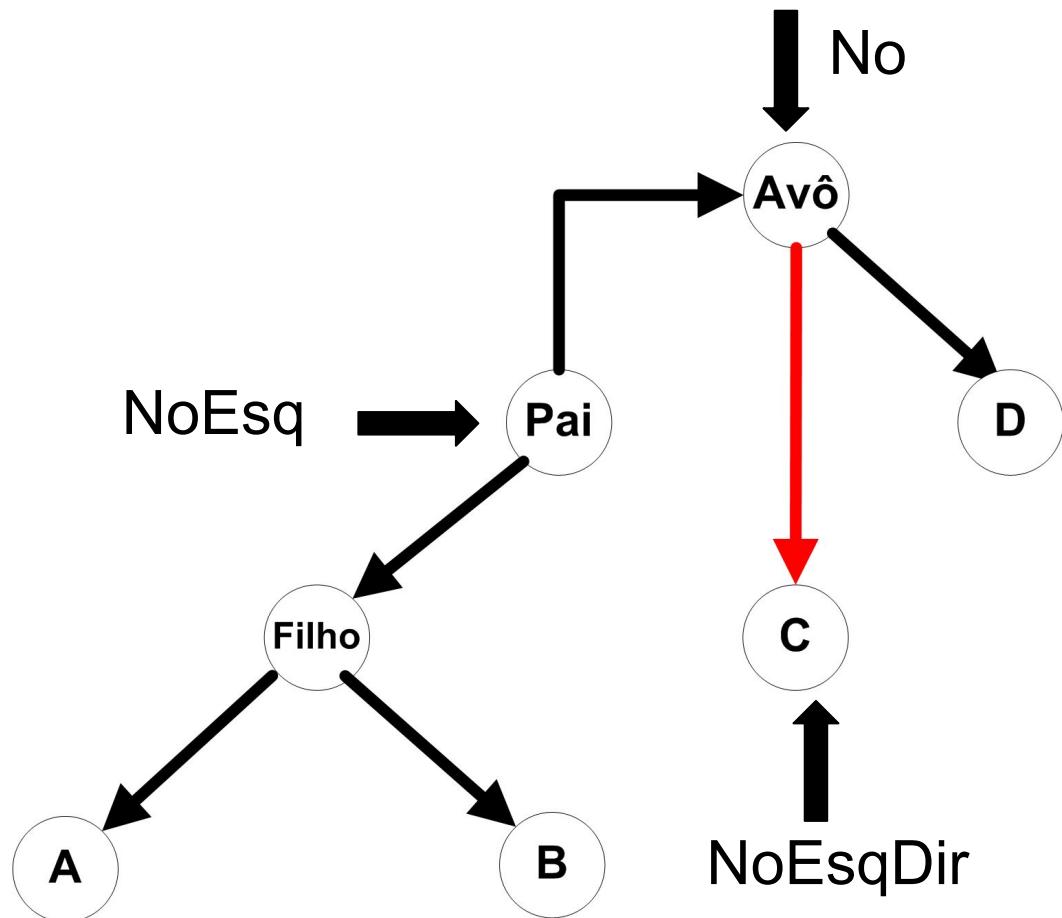
```
void metodo(){  
    ...  
    no = rotacionarDir(no);  
    ...  
}  
No rotacionarDir (No no) {  
    No noEsq = no.esq;  
    No noEsqDir = noEsq.dir;  
  
    noEsq.dir = no;  
    no.esq = noEsqDir;  
  
    return noEsq;  
}
```

## Implementação da Rotação à Direita



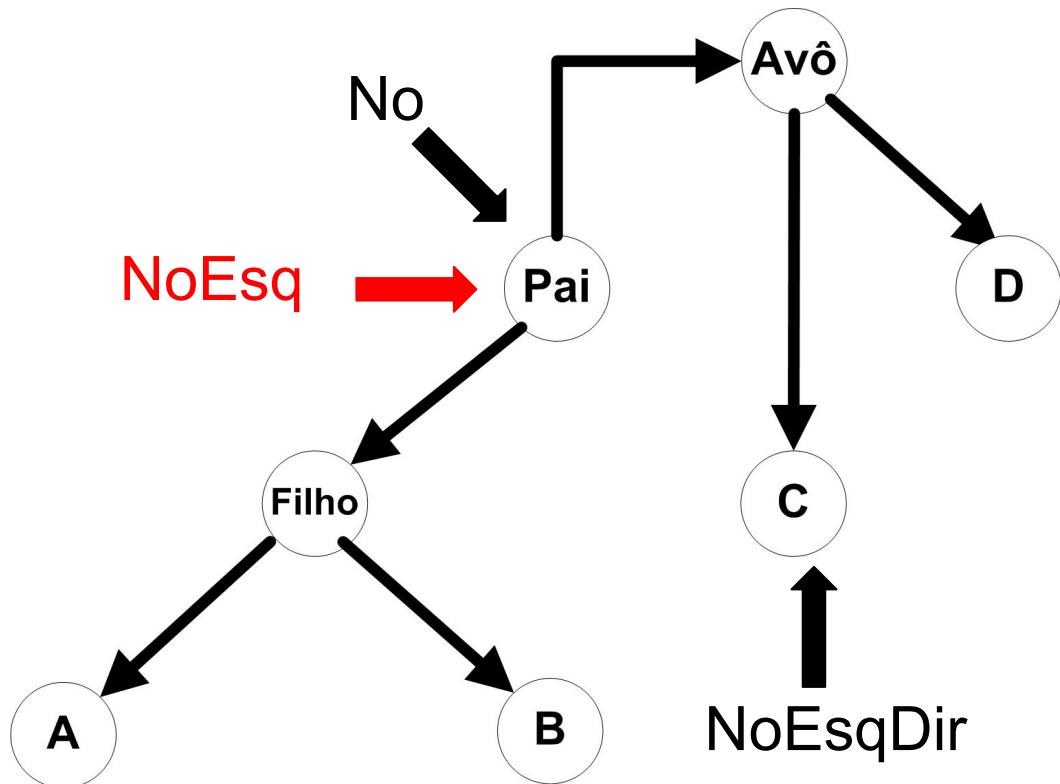
```
void metodo(){  
    ...  
    no = rotacionarDir(no);  
    ...  
}  
No rotacionarDir (No no) {  
    No noEsq = no.esq;  
    No noEsqDir = noEsq.dir;  
  
    noEsq.dir = no;  
    no.esq = noEsqDir;  
  
    return noEsq;  
}
```

## Implementação da Rotação à Direita



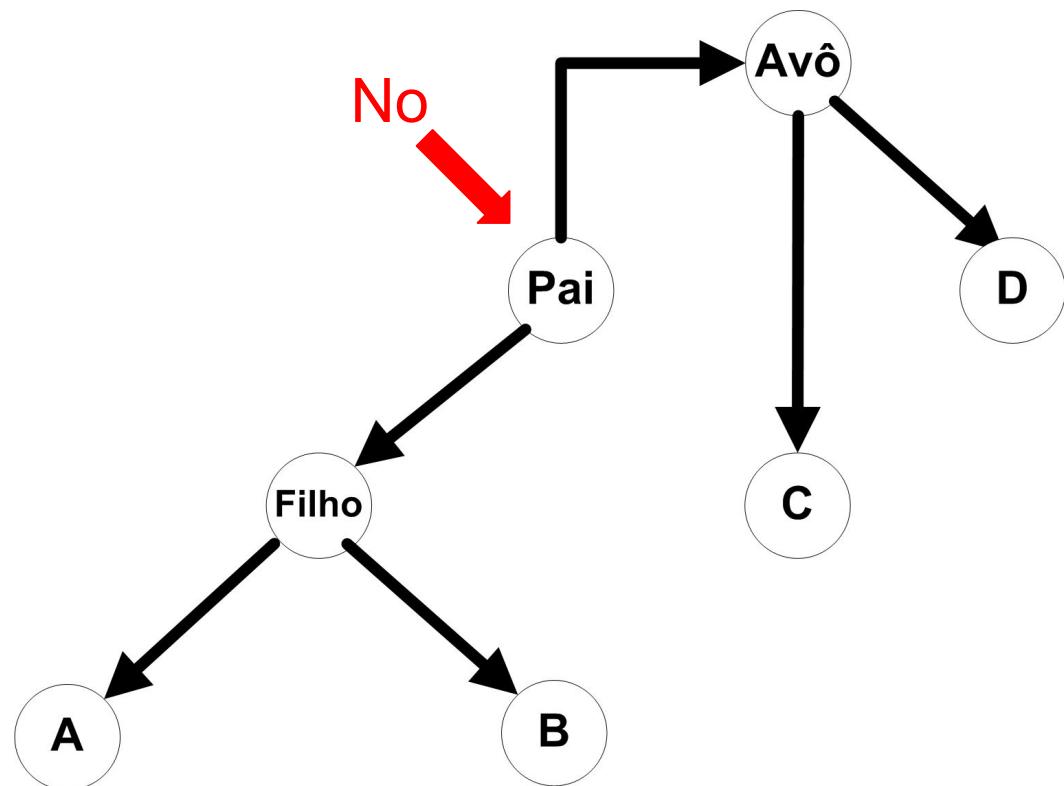
```
void metodo(){  
    ...  
    no = rotacionarDir(no);  
    ...  
}  
No rotacionarDir (No no) {  
    No noEsq = no.esq;  
    No noEsqDir = noEsq.dir;  
  
    noEsq.dir = no;  
    no.esq = noEsqDir;  
  
    return noEsq;  
}
```

## Implementação da Rotação à Direita



```
void metodo(){  
    ...  
    no = rotacionarDir(no);  
    ...  
}  
No rotacionarDir (No no) {  
    No noEsq = no.esq;  
    No noEsqDir = noEsq.dir;  
  
    noEsq.dir = no;  
    no.esq = noEsqDir;  
  
    return noEsq;  
}
```

## Implementação da Rotação à Direita



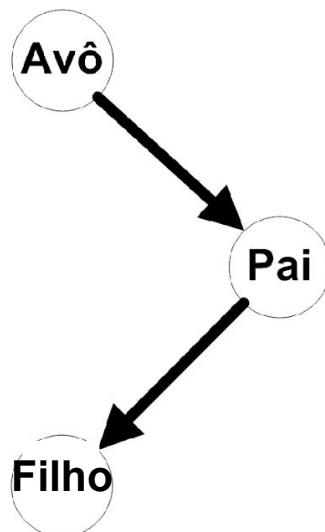
```
void metodo(){  
    ...  
    no = rotacionarDir(no);  
    ...  
}  
No rotacionarDir (No no) {  
    No noEsq = no.esq;  
    No noEsqDir = noEsq.dir;  
  
    noEsq.dir = no;  
    no.esq = noEsqDir;  
  
    return noEsq;  
}
```

# Tipos de Rotação

- Rotação simples à esquerda
- Rotação simples à direita
- **Rotação dupla direita - esquerda** 
- Rotação dupla esquerda - direita

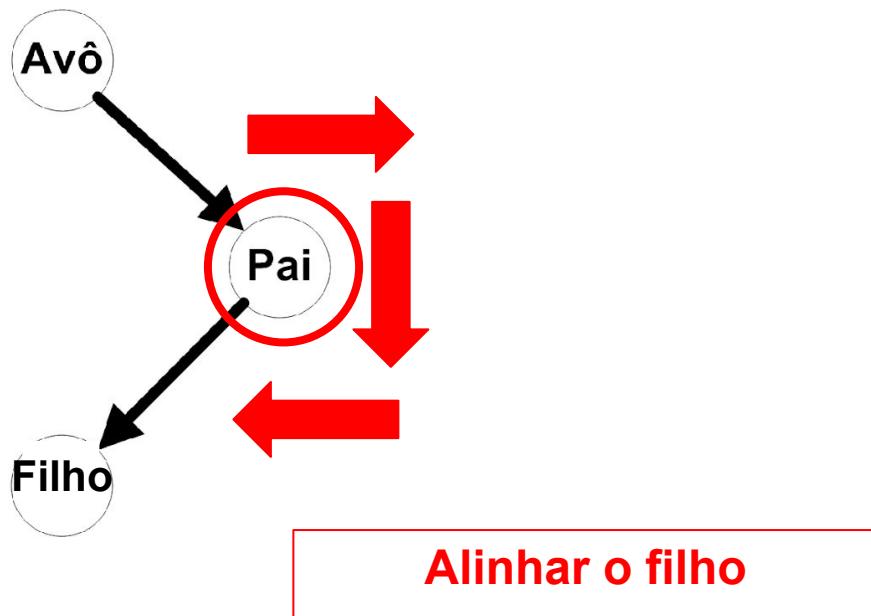
# Rotação Dupla Direita - Esquerda

- Usada em subárvores em que o pai está desbalanceado para a direita e o filho para a esquerda



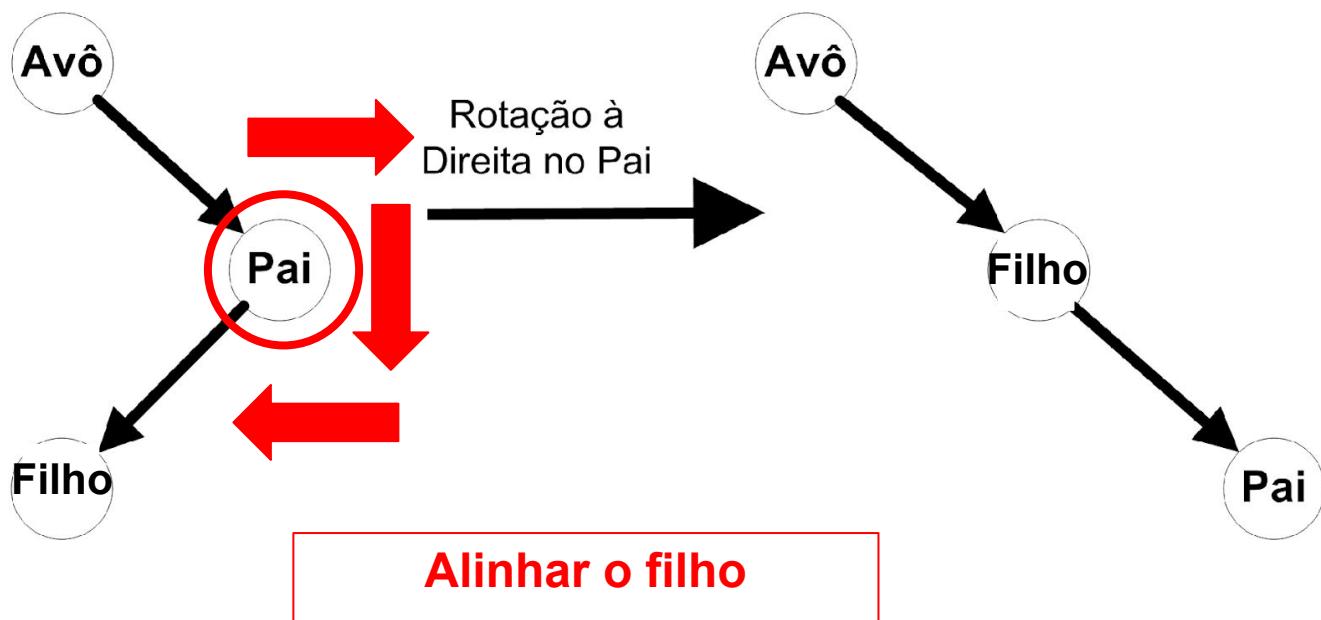
# Rotação Dupla Direita - Esquerda

- Usada em subárvores em que o pai está desbalanceado para a direita e o filho para a esquerda



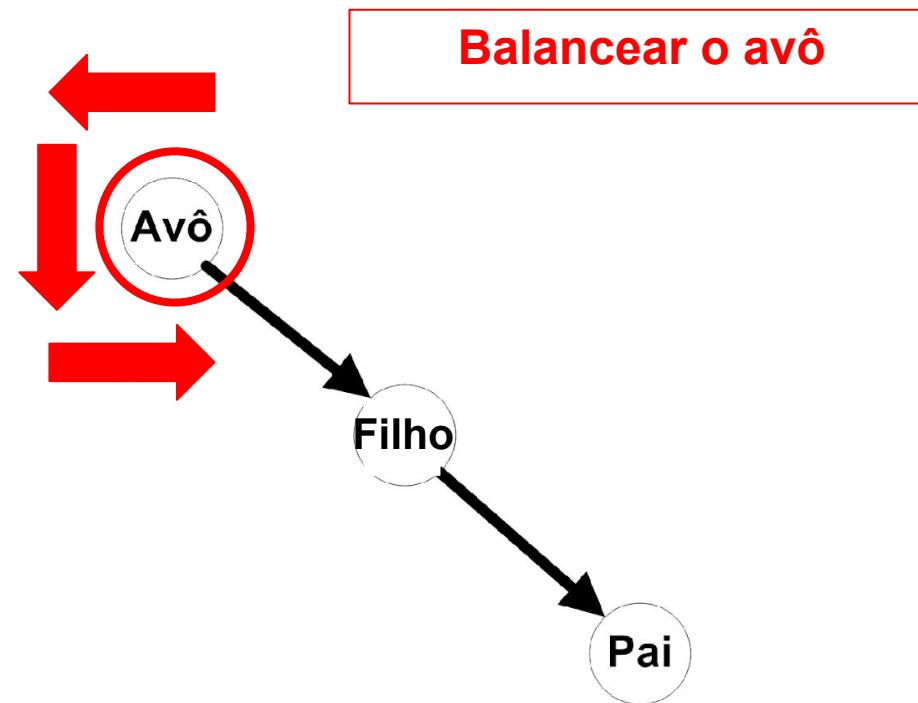
# Rotação Dupla Direita - Esquerda

- Usada em subárvores em que o pai está desbalanceado para a direita e o filho para a esquerda



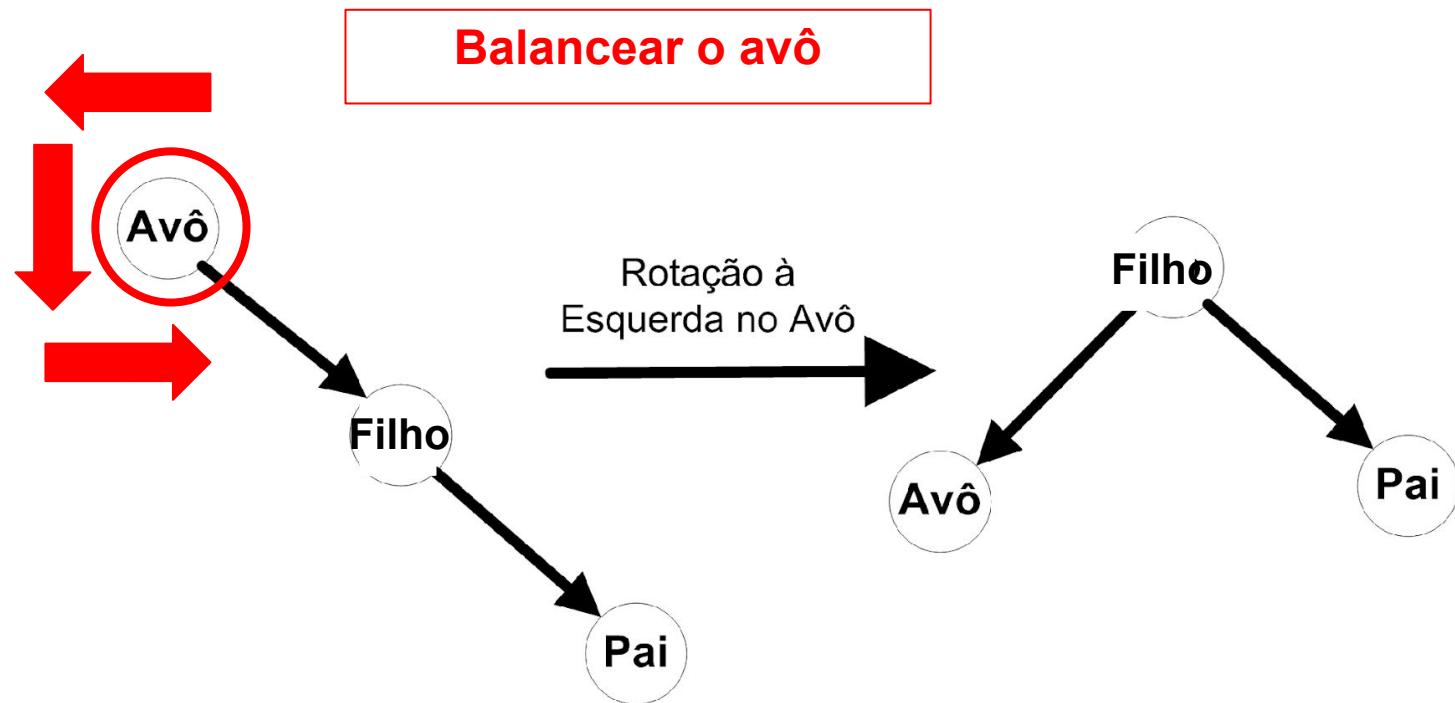
# Rotação Dupla Direita - Esquerda

- Usada em subárvores em que o pai está desbalanceado para a direita e o filho para a esquerda



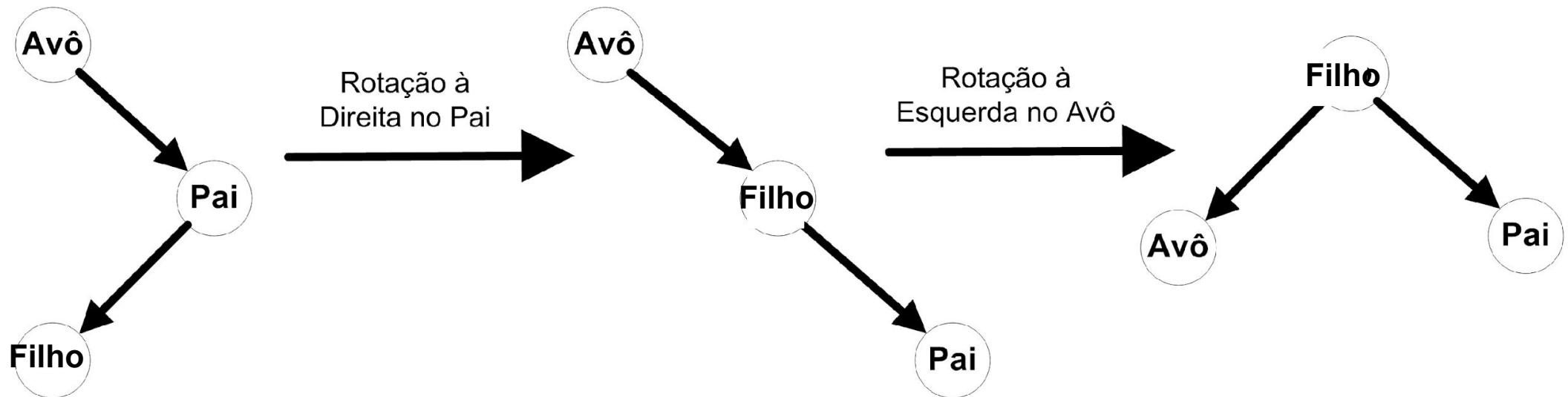
# Rotação Dupla Direita - Esquerda

- Usada em subárvores em que o pai está desbalanceado para a direita e o filho para a esquerda

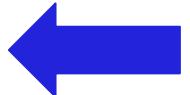


# Implementação da Rotação à Direita - Esquerda

```
No rotacionarDirEsq(No no) {  
    no.dir = rotacionarDir(no.dir);  
    return rotacionarEsq(no);  
}
```

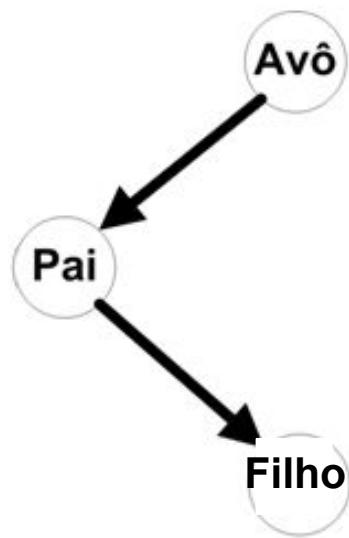


# Tipos de Rotação

- Rotação simples à esquerda
- Rotação simples à direita
- Rotação dupla direita - esquerda
- **Rotação dupla esquerda - direita** 

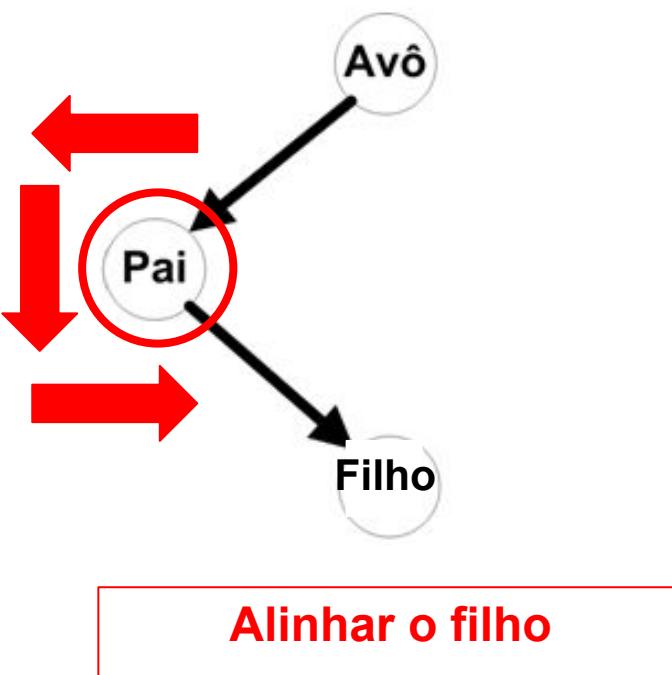
# Rotação Dupla Esquerda - Direita

- Usada em subárvores em que o pai está desbalanceado para a esquerda e o filho para a direita



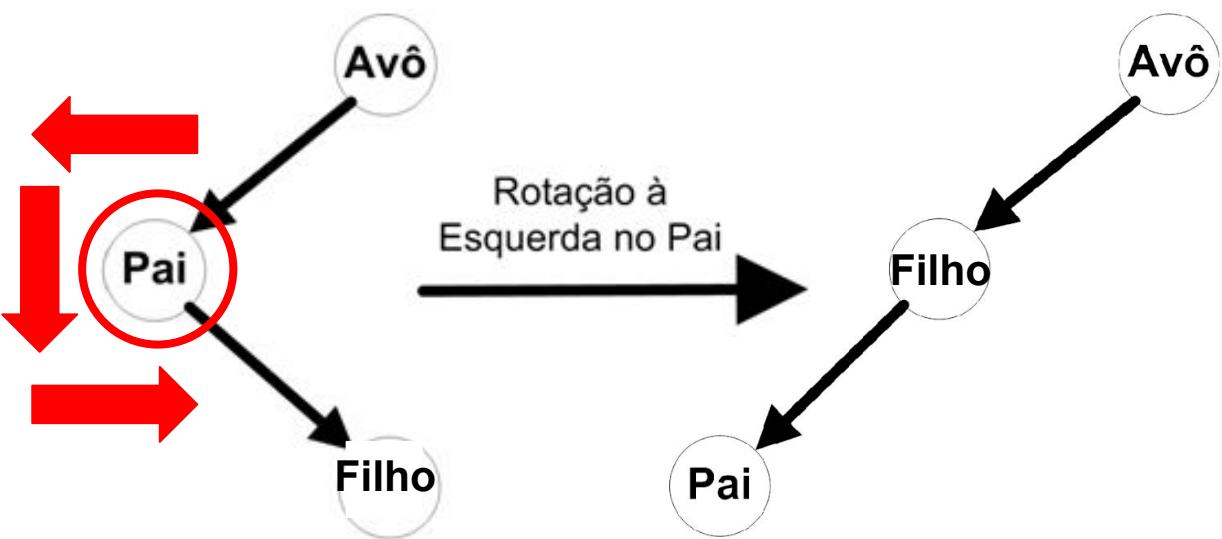
# Rotação Dupla Esquerda - Direita

- Usada em subárvores em que o pai está desbalanceado para a esquerda e o filho para a direita



# Rotação Dupla Esquerda - Direita

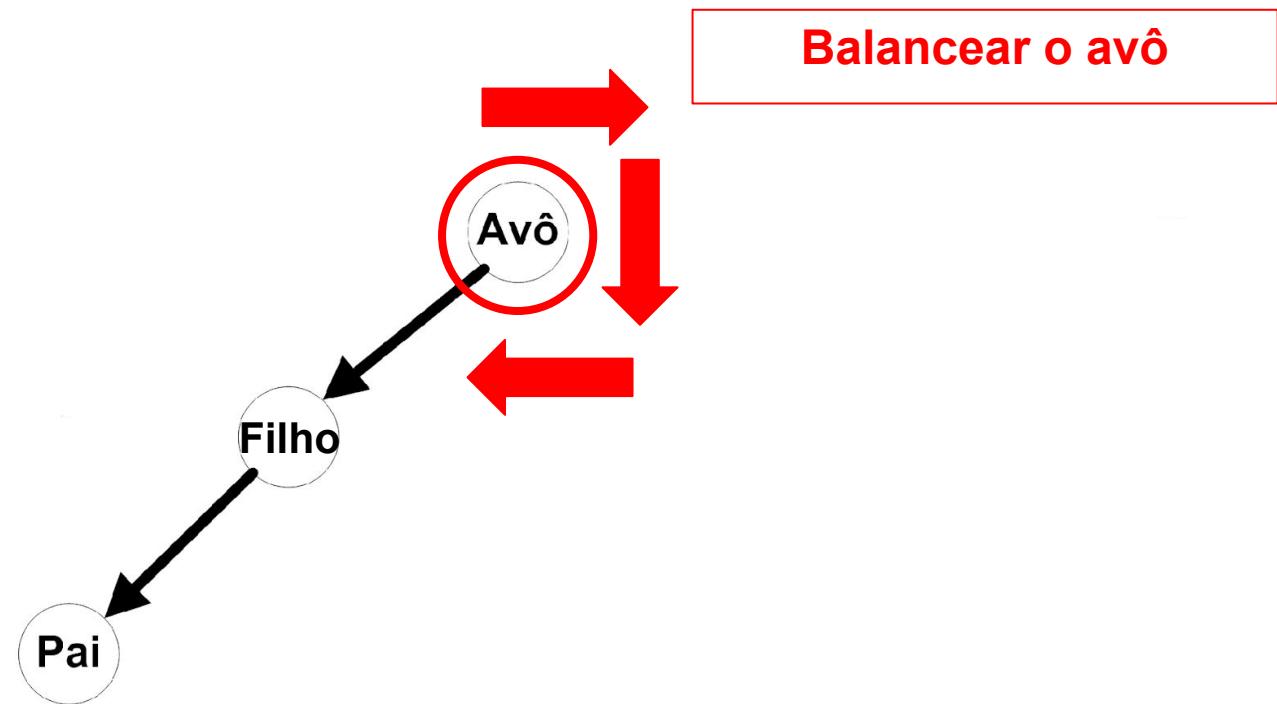
- Usada em subárvores em que o pai está desbalanceado para a esquerda e o filho para a direita



**Alinhar o filho**

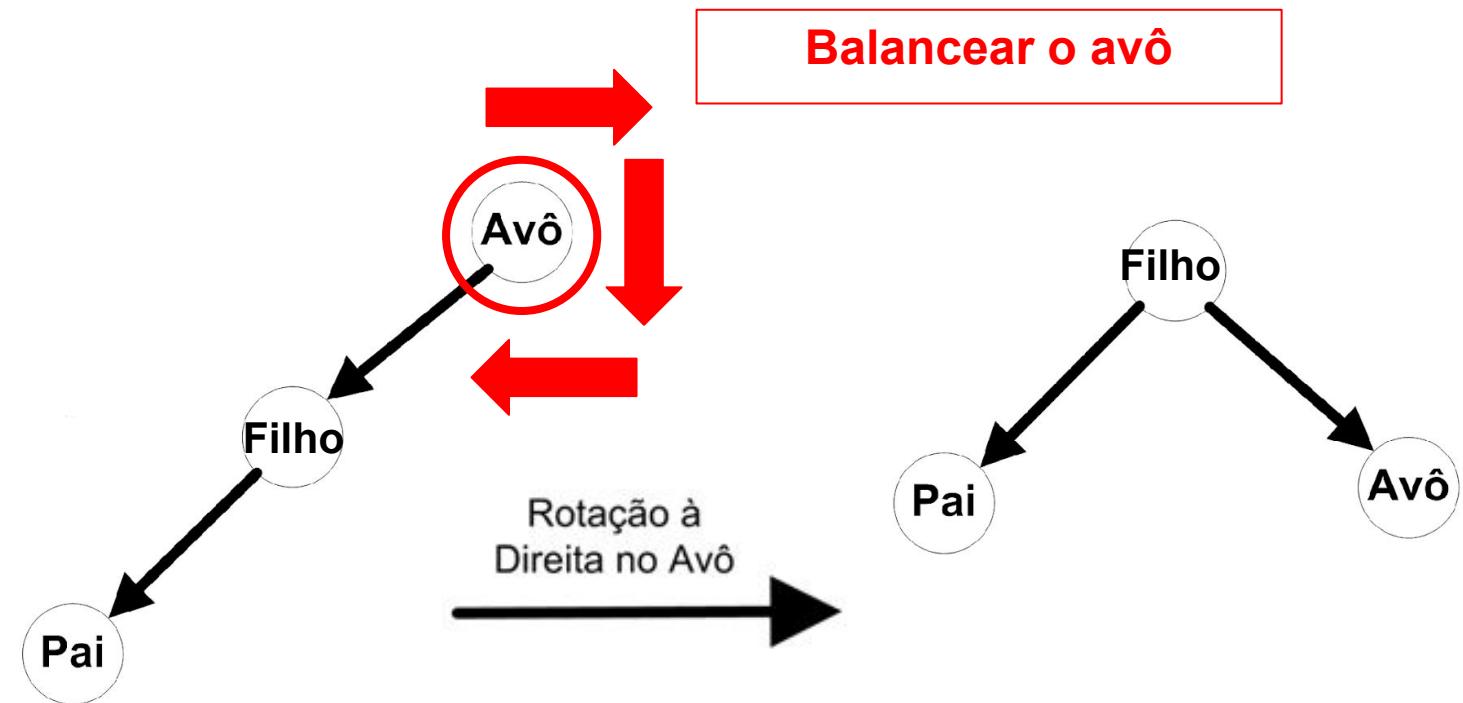
# Rotação Dupla Esquerda - Direita

- Usada em subárvores em que o pai está desbalanceado para a esquerda e o filho para a direita



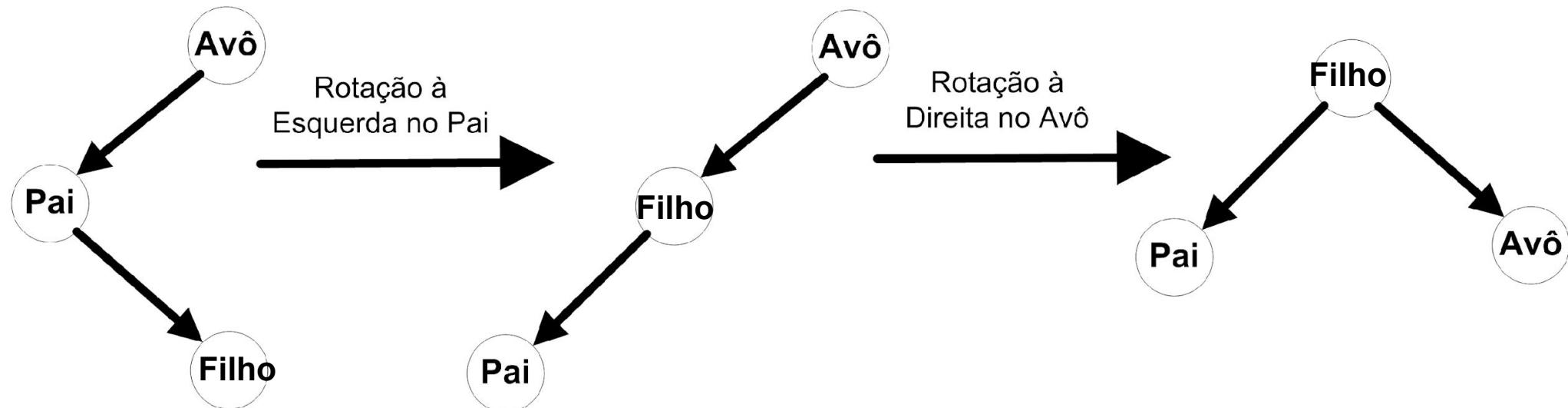
# Rotação Dupla Esquerda - Direita

- Usada em subárvores em que o pai está desbalanceado para a esquerda e o filho para a direita



# Implementação da Rotação à Esquerda - Direita

```
No rotacionarEsqDir(No no) {  
    no.esq = rotacionarEsq(no.esq);  
    return rotacionarDir(no);  
}
```



# Exercício Resolvido (1)

- No código da árvore binária, faça um método que leia três números inteiros e os insira na árvore. Se essa árvore estiver com 3 níveis, efetue uma das 4 rotações apresentadas nesta unidade

# Exercício Resolvido (1)

- No código da árvore binária, faça um método que leia três números inteiros e os insira na árvore. Se essa árvore estiver com 3 níveis, efetue uma das 4 rotações apresentadas nesta unidade

Para responder esta pergunta, precisamos saber quantas árvores distintas podemos fazer com três elementos

Da matemática, em combinações, temos

$\frac{3}{3} \quad \frac{2}{2} \quad \frac{1}{1}$

1, 2, 3 -- 1, 3, 2 --

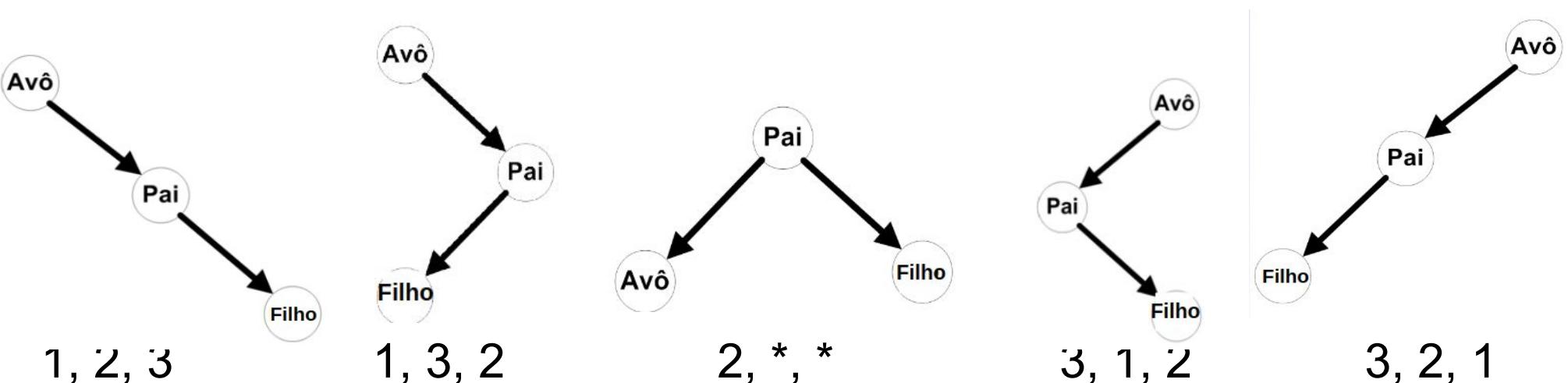
2, 1, 3 -- 2, 3, 1 --

3, 1, 2 -- 3, 2, 1

## Exercício Resolvido (1)

- No código da árvore binária, faça um método que leia três números inteiros e os insira na árvore. Se essa árvore estiver com 3 níveis, efetue uma das 4 rotações apresentadas nesta unidade

Árvores possíveis



## Exercício Resolvido (1)

- No código
- e os ins
- rotações

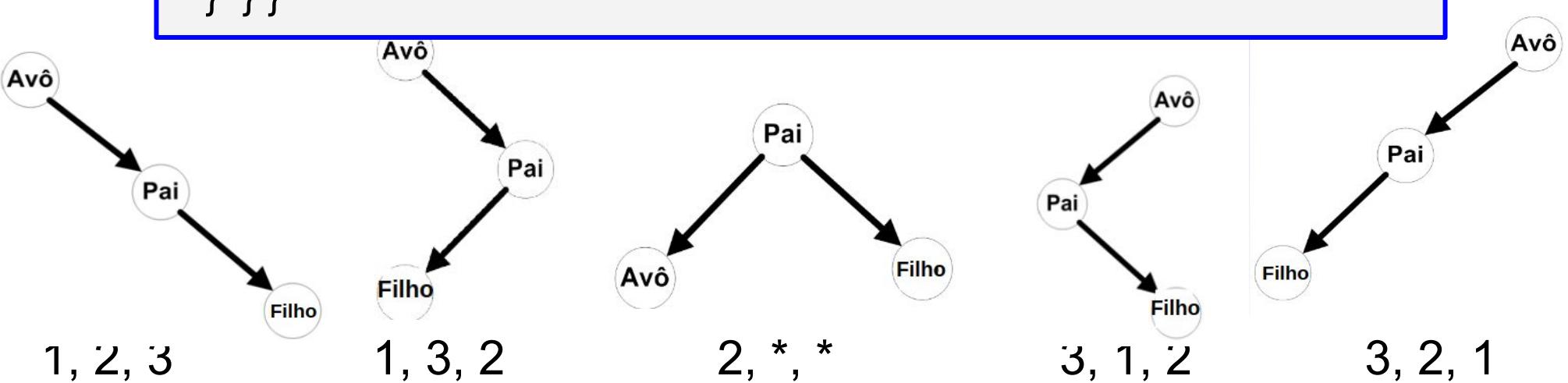
Árvores

```

void balancear (){
    if(raiz.esq != null && raiz.dir != null){ //casos [2,1,3] e [2,3,1]
    } else if (raiz.dir != null){
        if (raiz.dir.dir != null){ //caso [1,2,3]
        } else { // caso [1,3,2]
        }
    } else { // if(raiz.esq != null)
        if (raiz.esq.dir != null){ //caso [3, 1, 2]
        } else { //caso [3,2,1]
        }
    }
}

```

inteiros  
a das 4



1, 2, 3

1, 3, 2

2, \*, \*

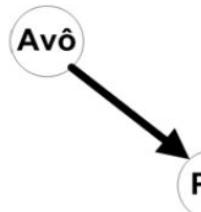
3, 1, 2

3, 2, 1

## Exercício Resolvido (1)

- No código e os ins
- rotações

Árvores



```

void balancear (){
    if(raiz.esq != null && raiz.dir != null){ //casos [2,1,3] e [2,3,1]
        ****
    } else if (raiz.dir != null){
        if (raiz.dir.dir != null){ //caso [1,2,3]
            ****
        } else { // caso [1,3,2]
            ****
        }
    } else { // if(raiz.esq != null)
        if (raiz.esq.dir != null){ //caso [3, 1, 2]
            ****
        } else { //caso [3,2,1]
            ****
        }
    }
}

```

1, 2, 3

1, 3, 2

2, \*, \*

3, 1, 2

3, 2, 1

inteiros

a das 4

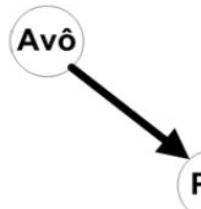
Avô



## Exercício Resolvido (1)

- No código e os ins rotações

Árvores



```

void balancear (){
    if(raiz.esq != null && raiz.dir != null){ //casos [2,1,3] e [2,3,1]
        System.out.println("Árvore balanceada");
    } else if (raiz.dir != null){
        if (raiz.dir.dir != null){ //caso [1,2,3]
            raiz = rotacionarEsq(raiz);
        } else { // caso [1,3,2]
            raiz = rotacionarDirEsq(raiz);
        }
    } else { // if(raiz.esq != null)
        if (raiz.esq.dir != null){ //caso [3, 1, 2]
            raiz = rotacionarEsqDir(raiz);
        } else { //caso [3,2,1]
            raiz = rotacionarDir(raiz);
        }
    }
}

```

1, 2, 3

1, 3, 2

2, \*, \*

3, 1, 2

3, 2, 1

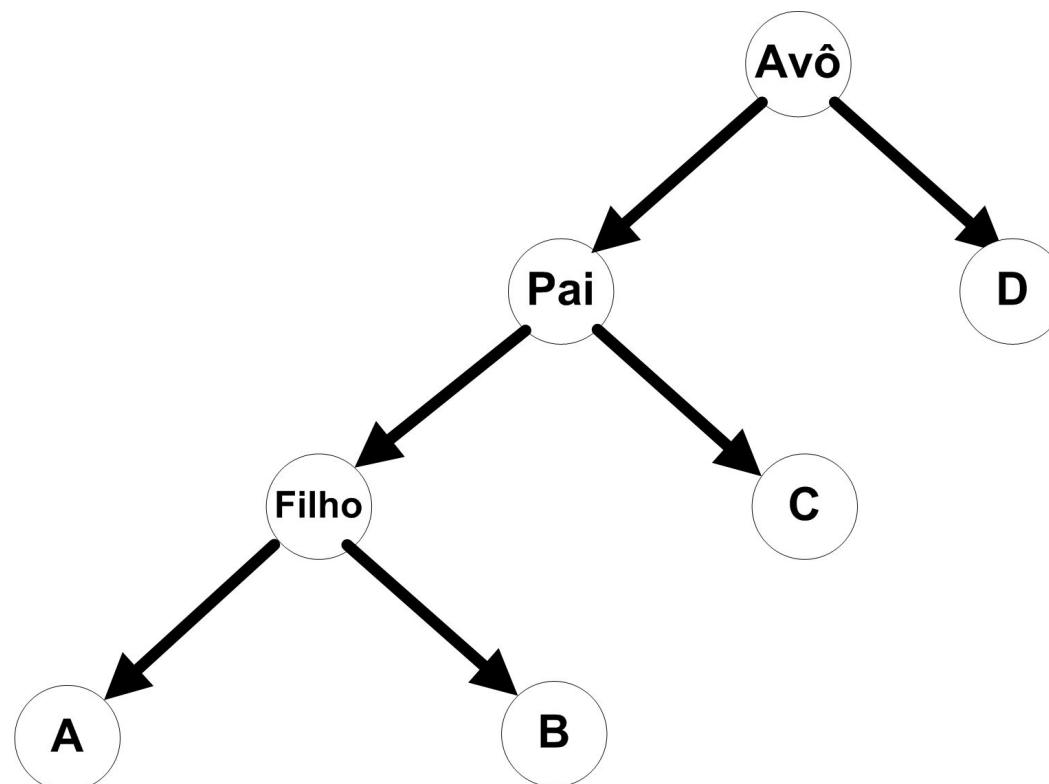
Avô

inteiros

a das 4

## Exercício Resolvido (2)

- Leia e insira os números em uma árvore binária não balanceada (unidade anterior) para que ela fique como a figura abaixo. Em seguida, execute o caminhar pré. Finalmente, efetue uma rotação à **direita** no nó avô para balancear nossa árvore e execute o caminhar pré novamente.

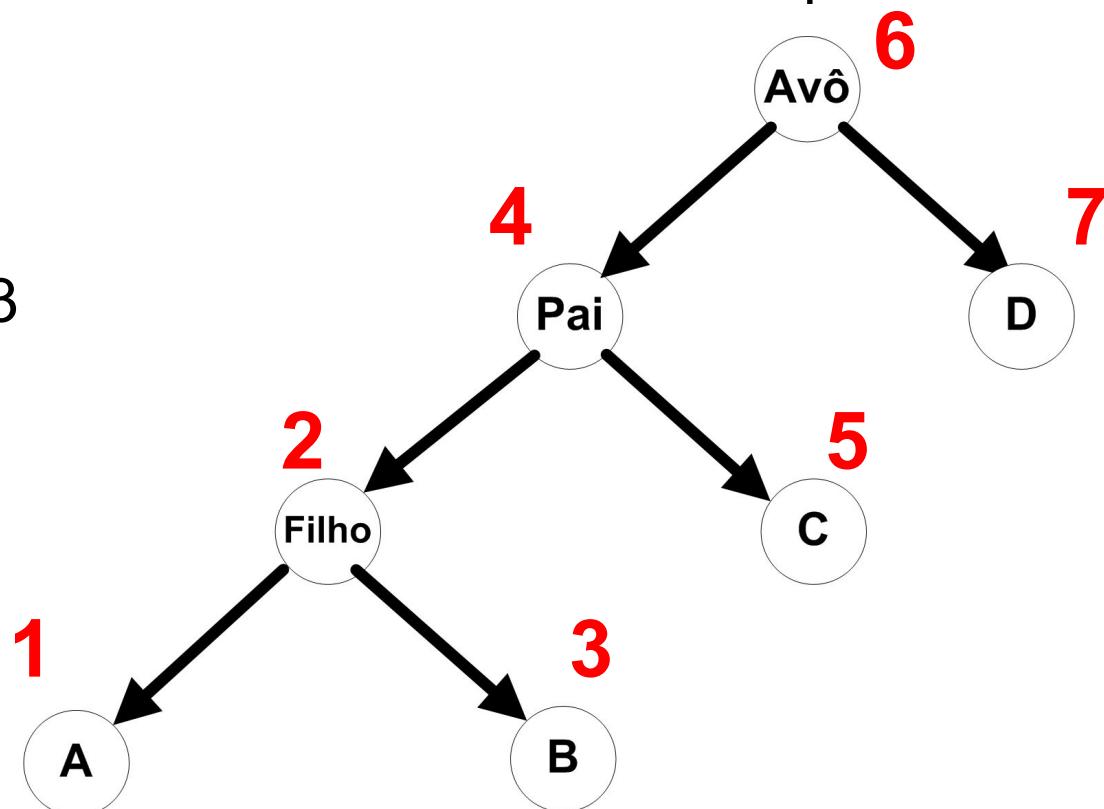


## Exercício Resolvido (2)

- Leia e insira os números em uma árvore binária não balanceada (unidade anterior) para que ela fique como a figura abaixo. Em seguida, execute o caminhar pré. Finalmente, efetue uma rotação à **direita** no nó avô para balancear nossa árvore e execute o caminhar pré novamente.

Sequência:

6, 4, 7, 2, 5, 1 e 3

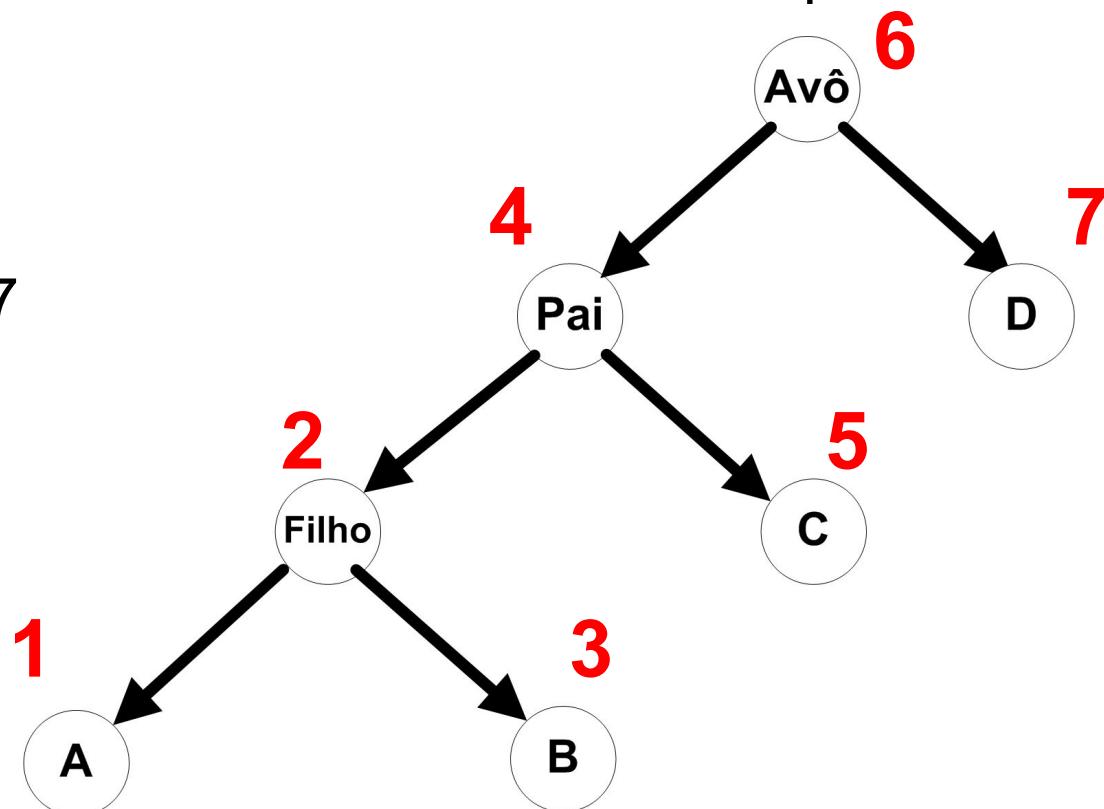


## Exercício Resolvido (2)

- Leia e insira os números em uma árvore binária não balanceada (unidade anterior) para que ela fique como a figura abaixo. Em seguida, execute o caminhar pré. Finalmente, efetue uma rotação à **direita** no nó avô para balancear nossa árvore e execute o caminhar pré novamente.

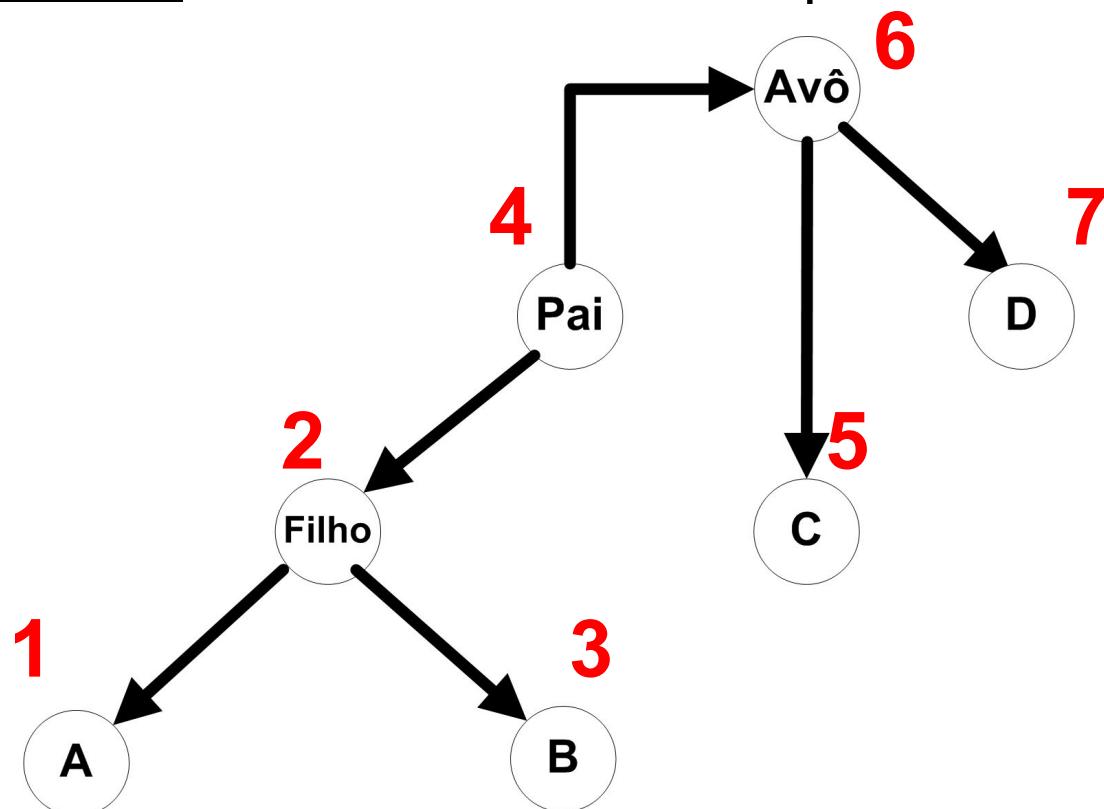
Caminhar pré:

6, 4, 2, 1, 3, 5 e 7



## Exercício Resolvido (2)

- Leia e insira os números em uma árvore binária não balanceada (unidade anterior) para que ela fique como a figura abaixo. Em seguida, execute o caminhar pré. Finalmente, efetue uma rotação à **direita** no nó avô para balancear nossa árvore e execute o caminhar pré novamente.

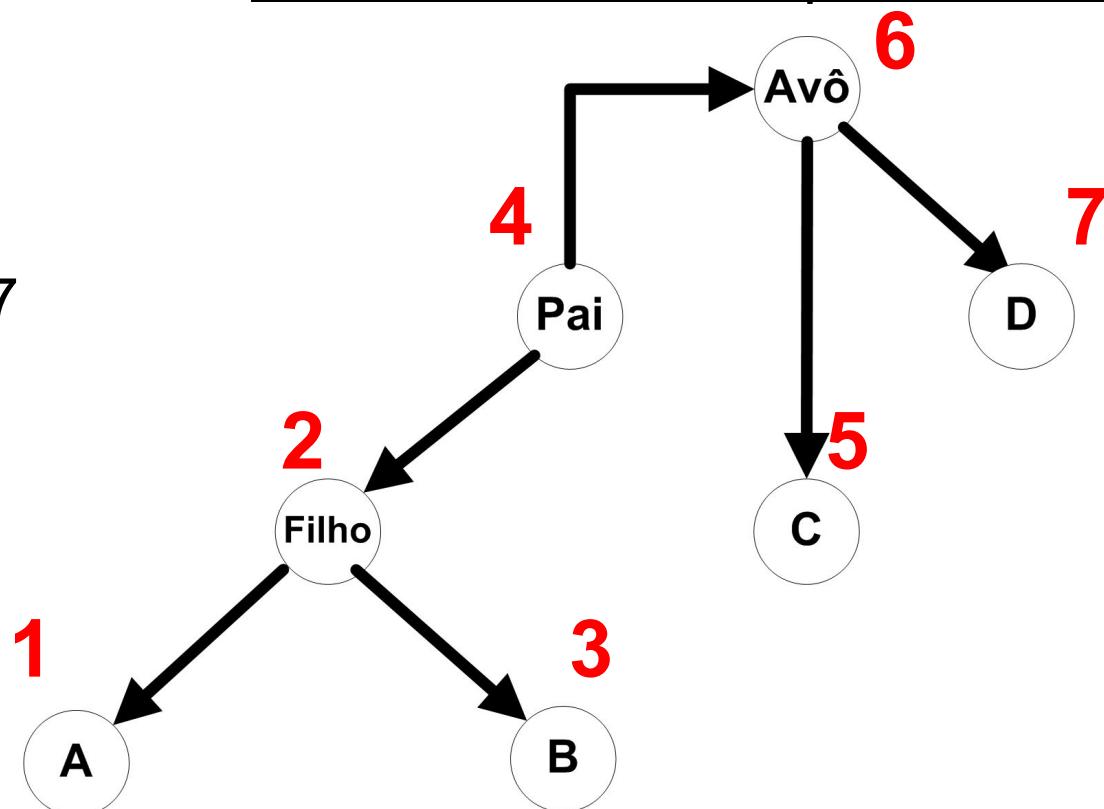


## Exercício Resolvido (2)

- Leia e insira os números em uma árvore binária não balanceada (unidade anterior) para que ela fique como a figura abaixo. Em seguida, execute o caminhar pré. Finalmente, efetue uma rotação à **direita** no nó avô para balancear nossa árvore e execute o caminhar pré novamente.

Caminhar pré:

4, 2, 1, 3, 6, 5 e 7

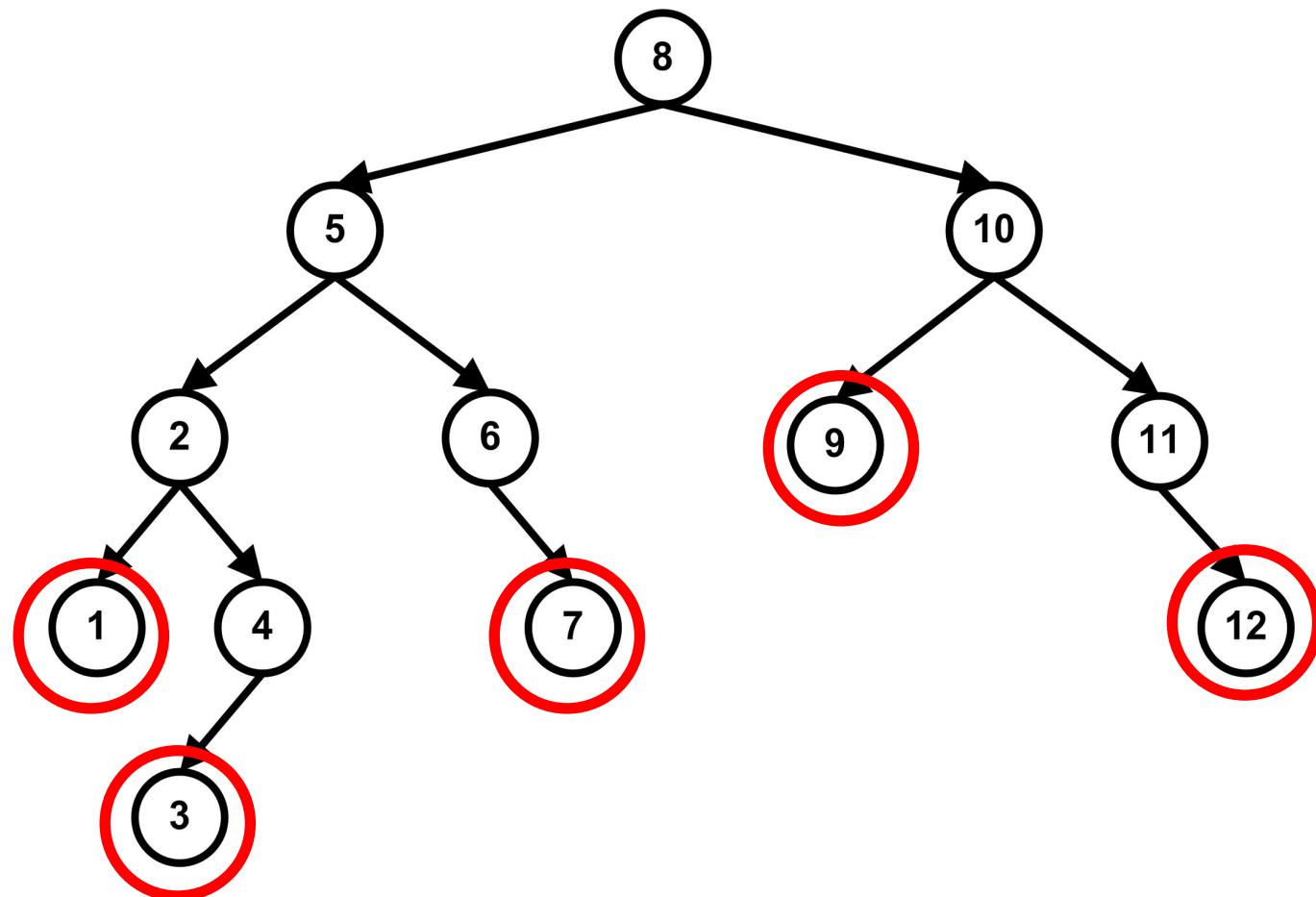


# Balanceamento de Árvores

- Qual é o custo para se manter uma árvore balanceada?
- Na prática, não existe “muita” diferença entre árvores balanceadas ou praticamente balanceadas
- Algumas árvores balanceadas como a AVL e a Alvinegra permitem árvores praticamente平衡adas

# Balanceamento de Árvores

- Exemplo de árvore AVL em que as folhas ocupam mais de dois níveis



# Balanceamento de Árvores

- Exemplo de árvore Alvinegra em que as folhas ocupam mais de dois níveis

