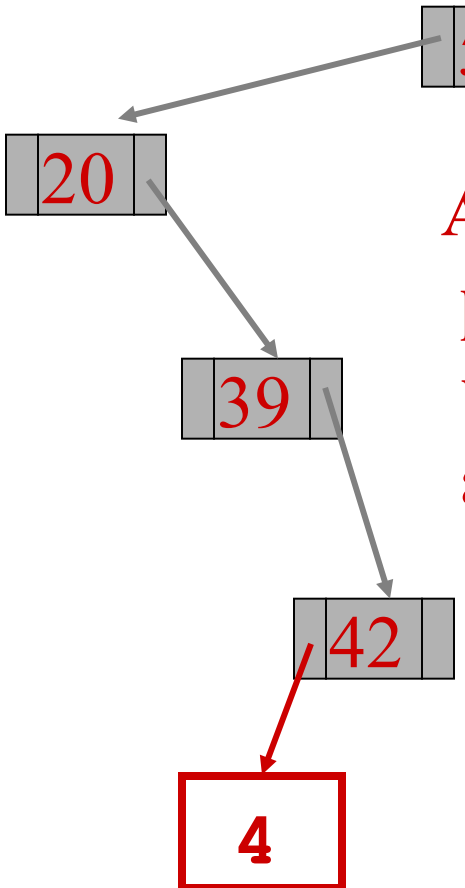
Árvores AVL

Rotação Simples e Dupla

Katia Guimarães

Possível Problema

Exemplo: 50, 20, 39, 42, 40 ...



A árvore binária pode degenerar para uma estrutura próxima a uma lista ligada, e o tempo de acesso deixa de ser logarítmico.

Solução

Procurar manter todas as folhas mais ou menos na mesma altura.

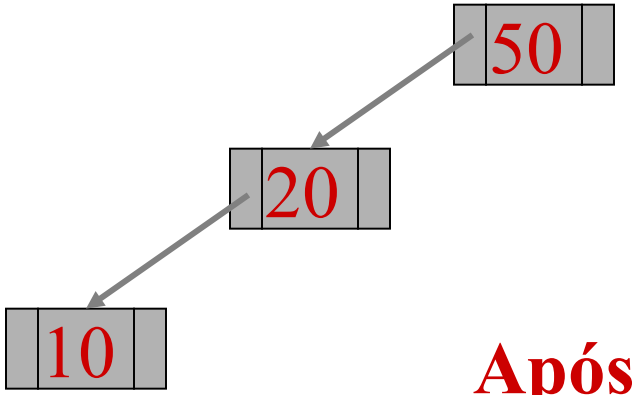
PROPRIEDADE AVL:

Para todo nó

$$| \text{altura(dir)} - \text{altura(esq)} | < 2$$

Possível Problema

Exemplo: 50, 20, 10, ...

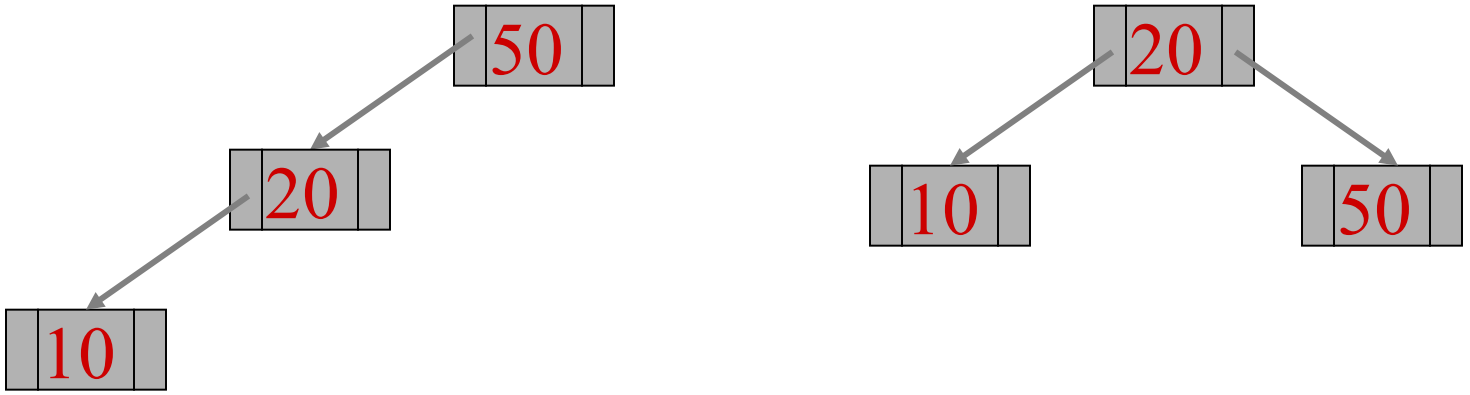


Após a inserção do elemento 10, a árvore binária perde a propriedade AVL.

SOLUÇÃO: Rotação.

Rotação Simples

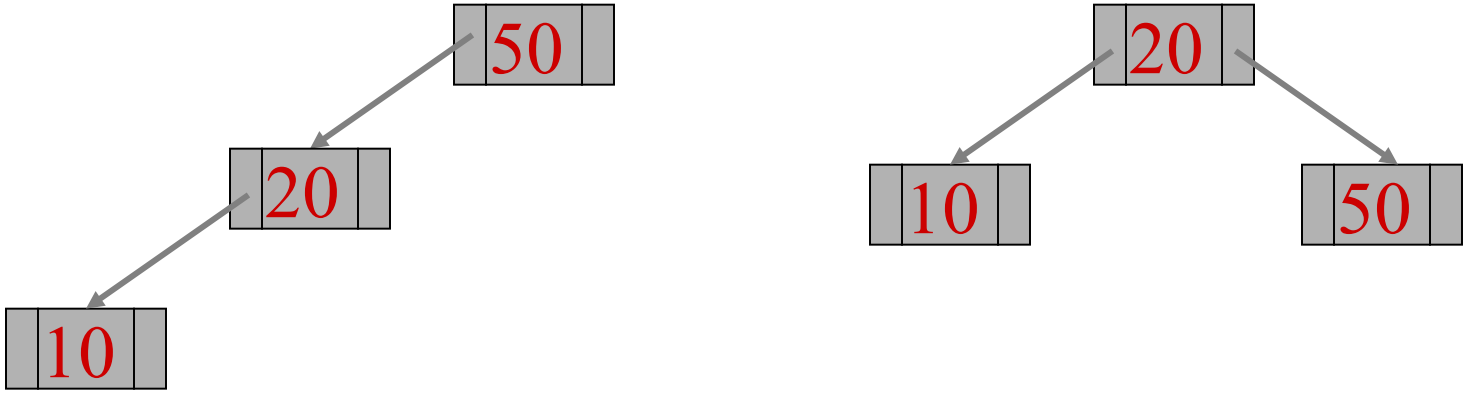
1.5 Ocorre quando um nó está desbalanceado e seu filho estiver no mesmo sentido da inclinação, formando uma linha reta.



Promover o elemento do meio através de um giro no sentido horário.

Rotação Simples à Direita

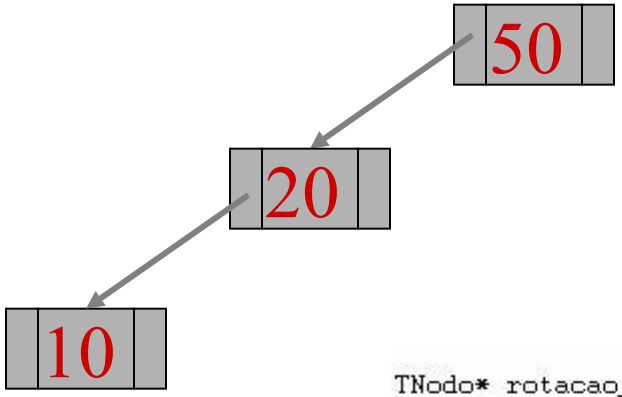
Inserção à esquerda de árvore desbalanceada à esquerda (bal = -1)

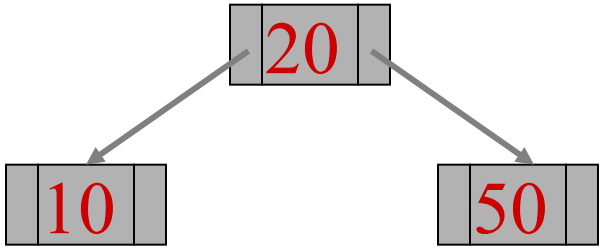


Promover o elemento do meio através de um giro no sentido horário.

Rotação Simples à Direita

Inserção à esquerda de árvore desbalanceada à esquerda (bal = -1)

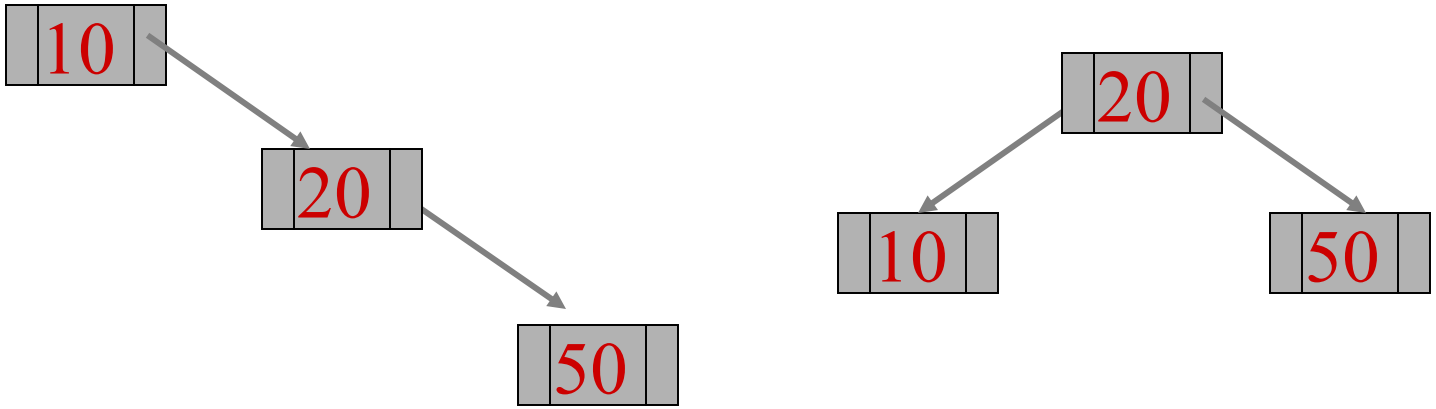





```
TNodo* rotacao_direita(TNodo* N3)
{
        TNodo* N2= N3->esq;
        if(N2->dir) N3->esq = N2->dir;
        else N3->esq=NULL;
        N2->dir=N3;
        return N2;
}
```

Rotação Simples à Esquerda

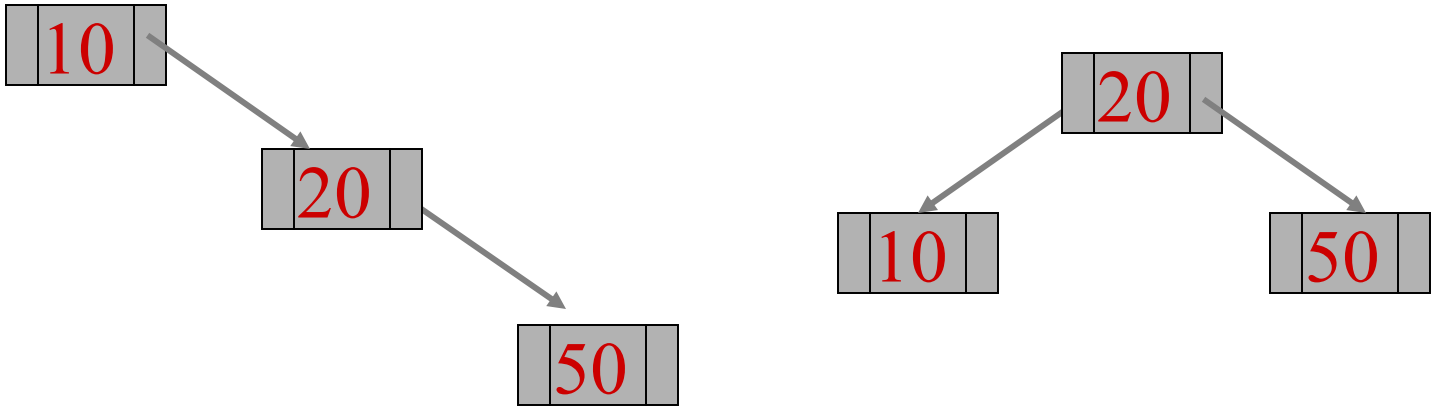
Inserção à direita de árvore desbalanceada à direita (bal = +1)



Promover o elemento do meio através de um giro no sentido anti-horário.

Rotação Simples à Esquerda

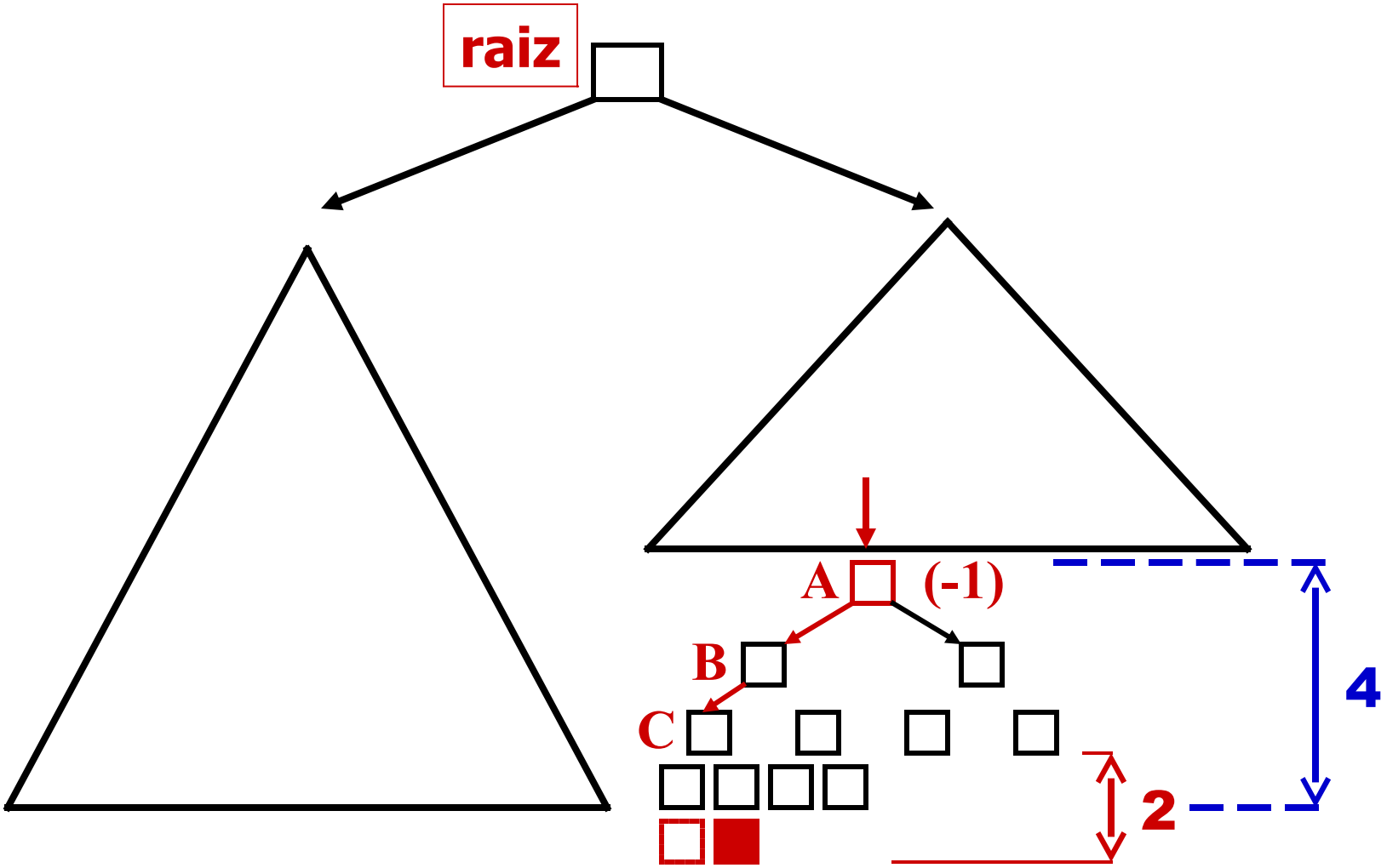
Inserção à direita de árvore desbalanceada à direita (bal = +1)




```
TNodo* rotacao_esquerda(TNodo* N1)
{
        TNodo* N2= N1->dir;
        if(N2->esq) N1->dir= N2->esq;
        else N1->dir=NULL;
        N2->esq=N1;
        return N2;
}
```

Rotação Simples

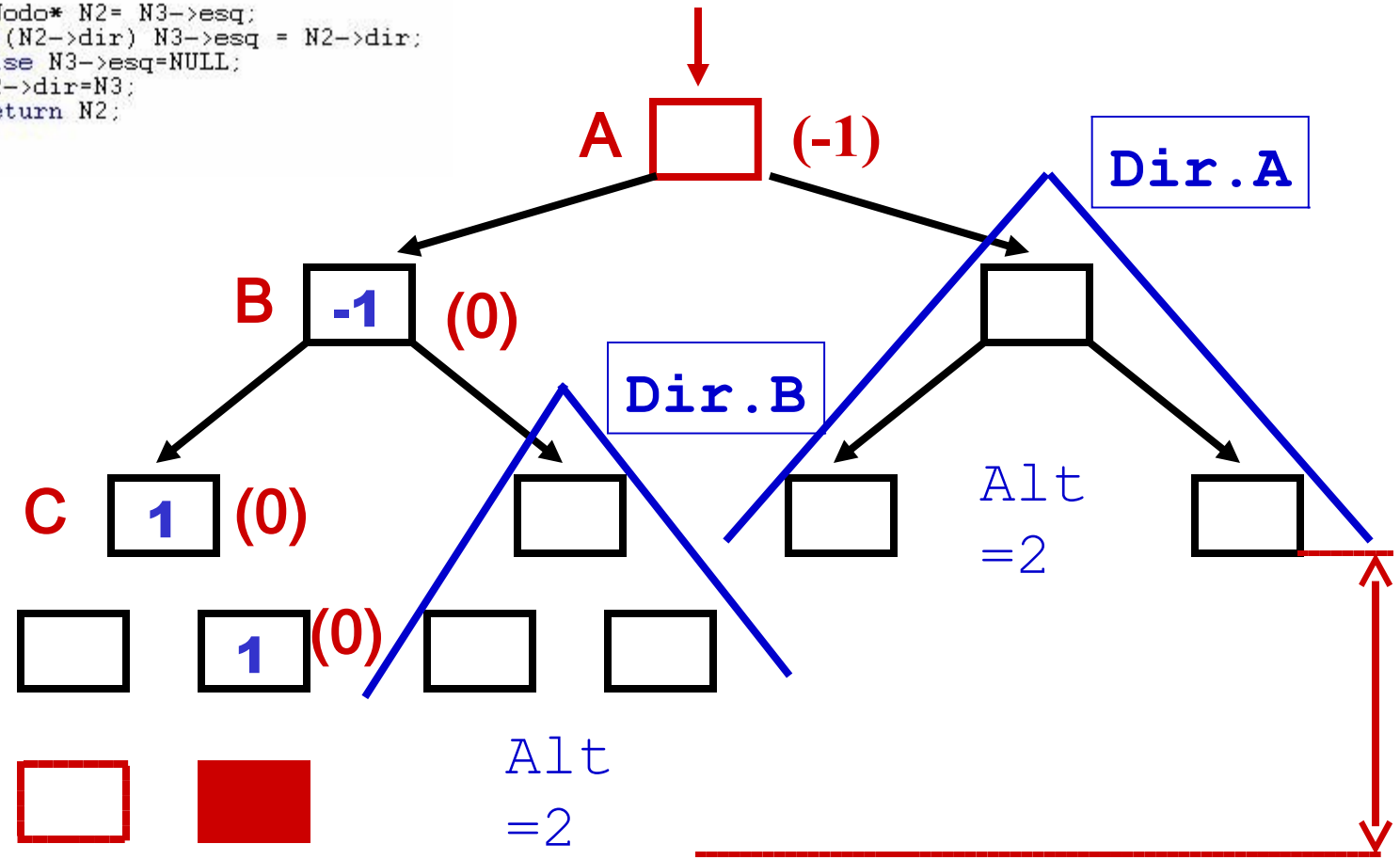
Inserção à esquerda de nó crítico A com (bal = -1), à esquerda do nó B.



Rotação Simples

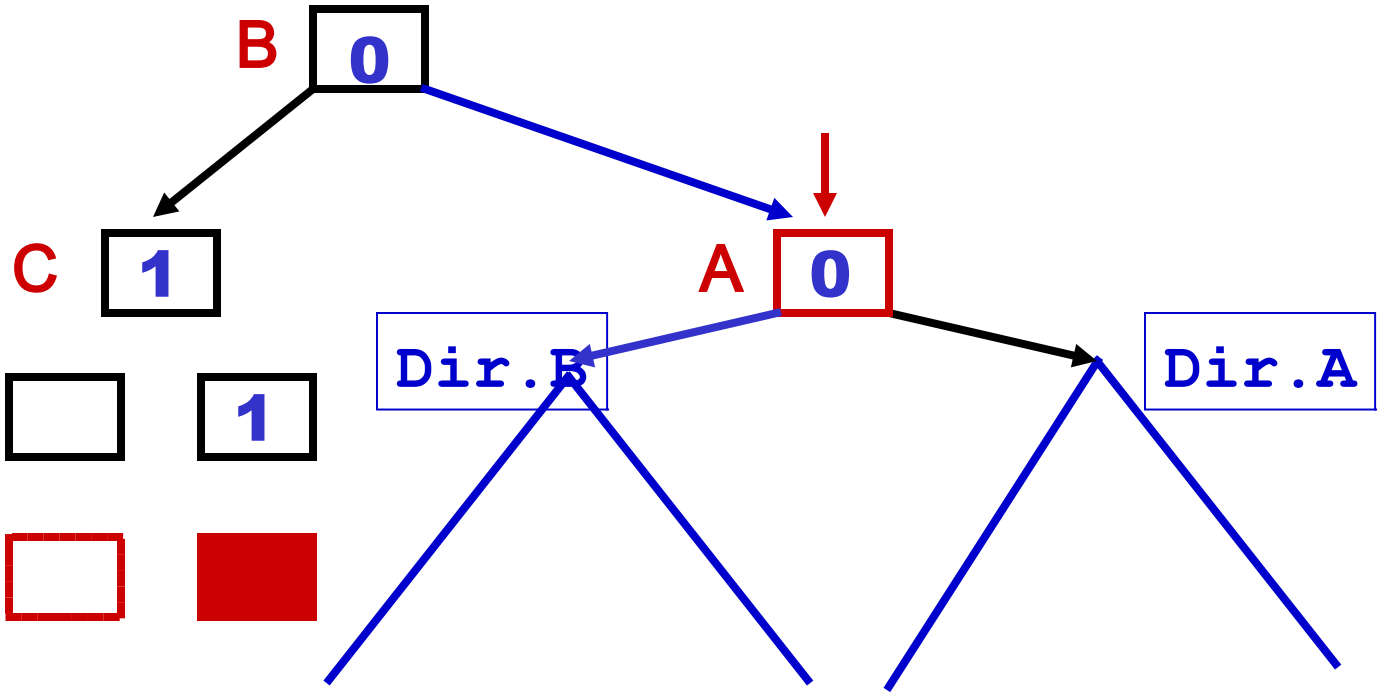
Inserção à esquerda de nó crítico A com (bal = -1), à esquerda do nó B.

```
TNodo* rotacao_direita(TNodo* N3)
{
        TNodo* N2= N3->esq;
        if(N2->dir) N3->esq = N2->dir;
        else N3->esq=NULL;
        N2->dir=N3;
        return N2;
}
```



Rotação Simples

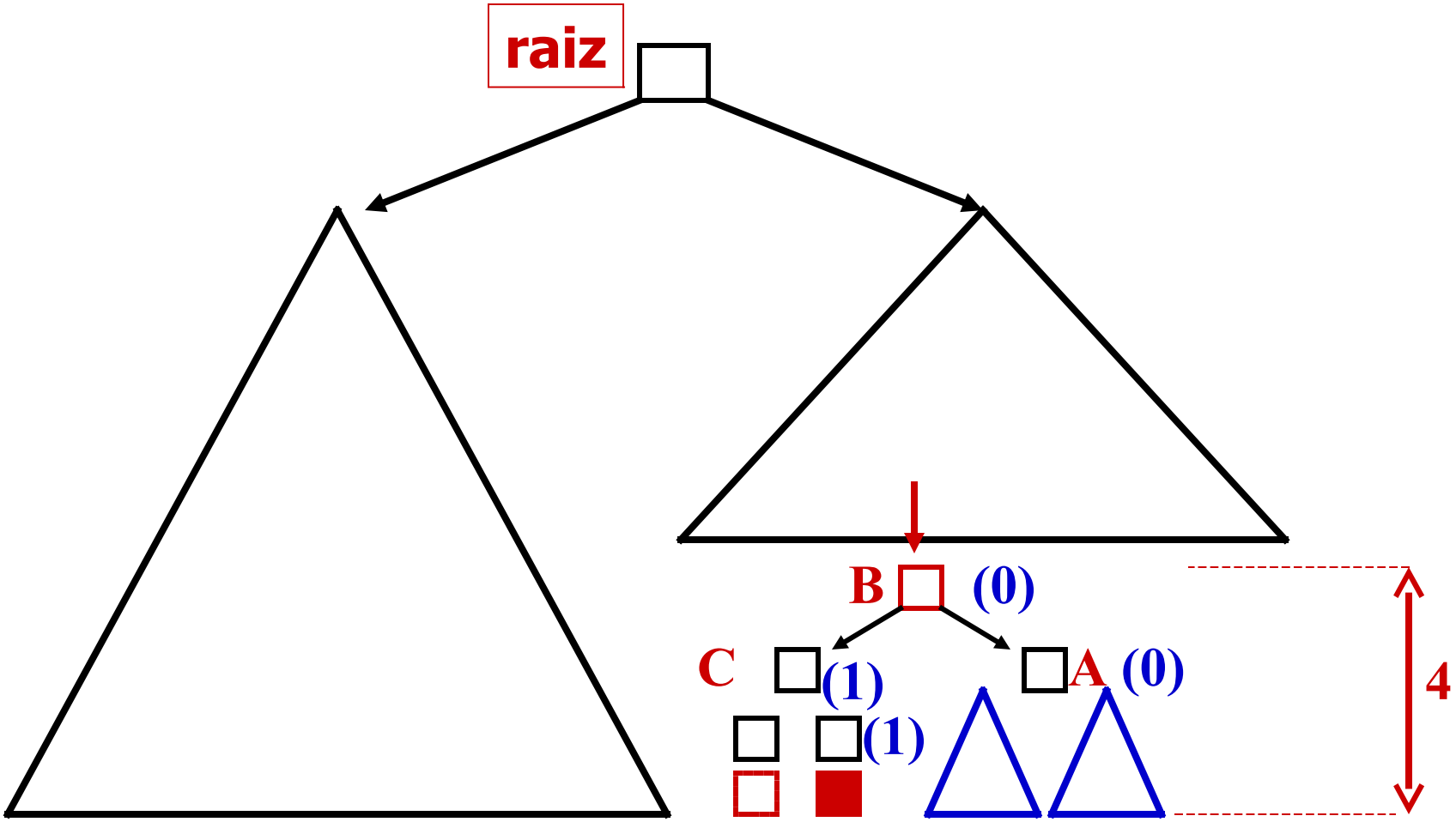
Inserção à esquerda de nó crítico A com (bal = -1), à esquerda do nó B.




```
TNodo* rotacao_direita(TNodo* N3)
{
        TNodo* N2= N3->esq;
        if(N2->dir) N3->esq = N2->dir;
        else N3->esq=NULL;
        N2->dir=N3;
        return N2;
}
```

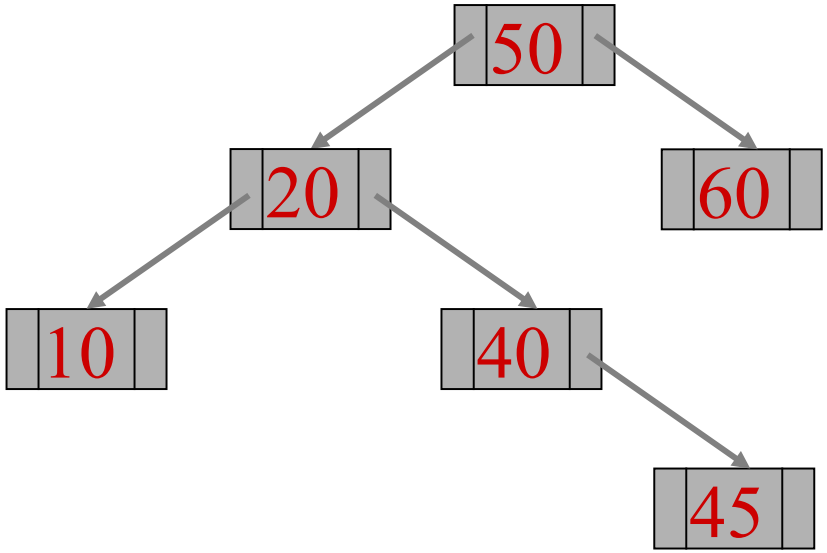
Rotação Simples

Inserção à esquerda de nó crítico A com (bal = -1), à esquerda do nó B.



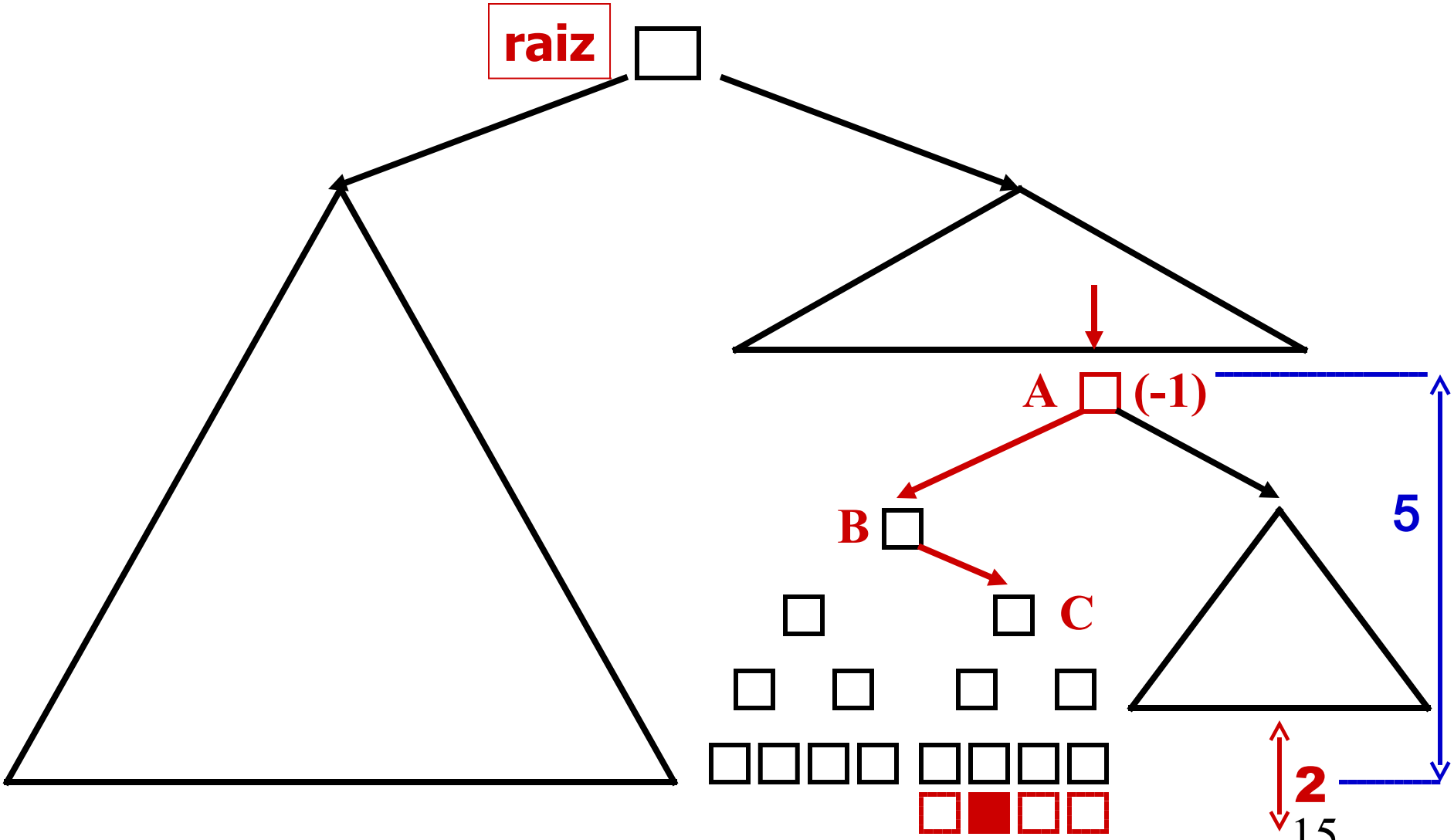
Rotação Dupla

Ocorre quando um nó estiver desbalanceado e seu filho estiver inclinado no sentido inverso ao pai, formando um "joelho"



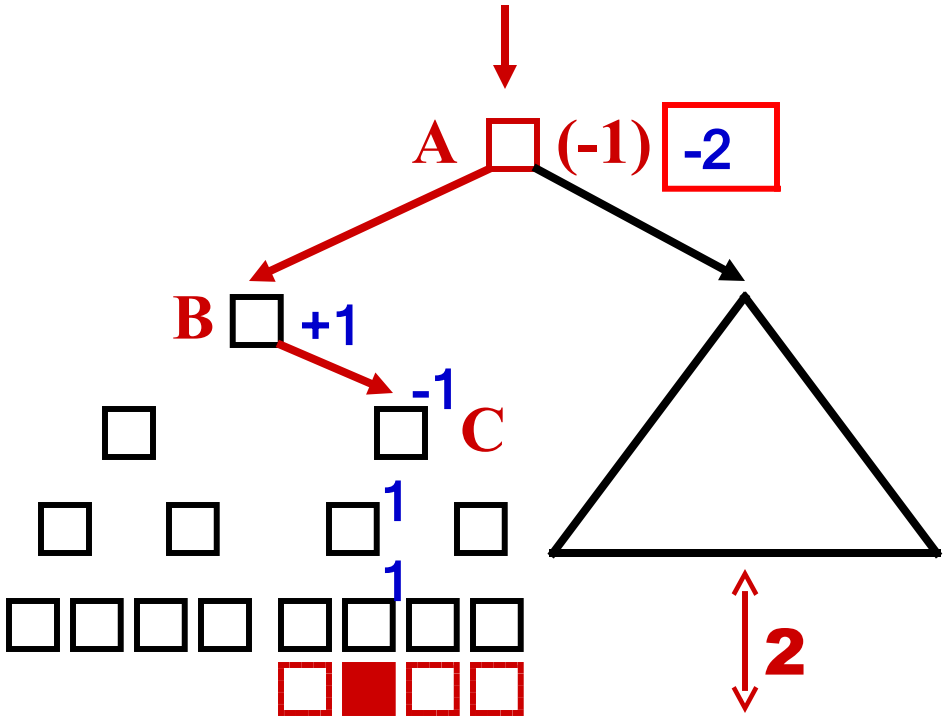
Rotação Dupla

Inserção à esquerda de nó crítico A com (bal = -1), à direita do nó B.



Rotação Dupla à Direita

Inserção à esquerda de nó crítico A com (bal = -1), à direita do nó B.



Rotação Dupla à Direita

Inserção à esquerda de nó crítico A com (bal = -1), à direita do nó B.

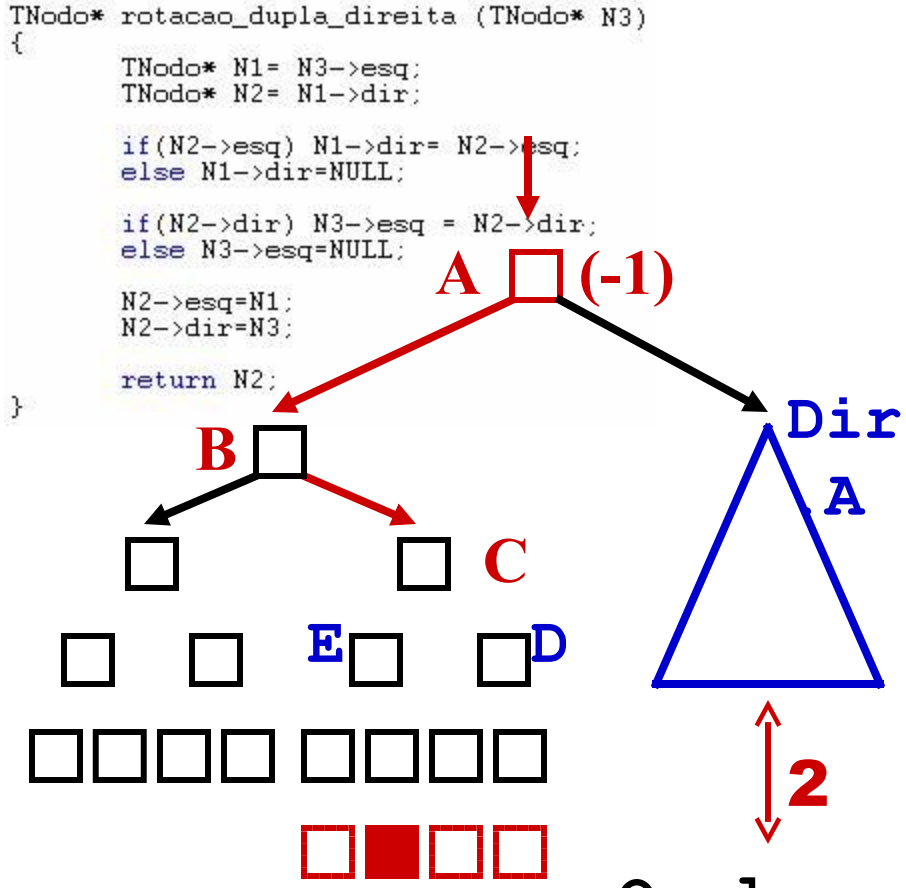
```
TNodo* rotacao_dupla_direita (TNodo* N3)
{
        TNodo* N1= N3->esq;
        TNodo* N2= N1->dir;

        if(N2->esq) N1->dir= N2->esq;
        else N1->dir=NULL;

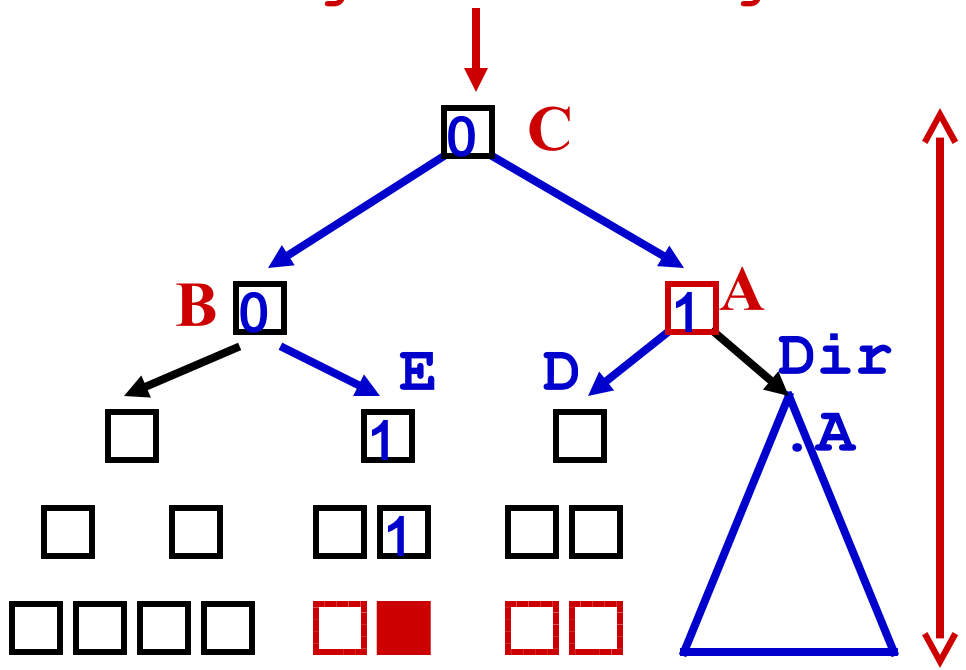
        if(N2->dir) N3->esq = N2->dir;
        else N3->esq=NULL;

        N2->esq=N1;
        N2->dir=N3;

        return N2;
}
```



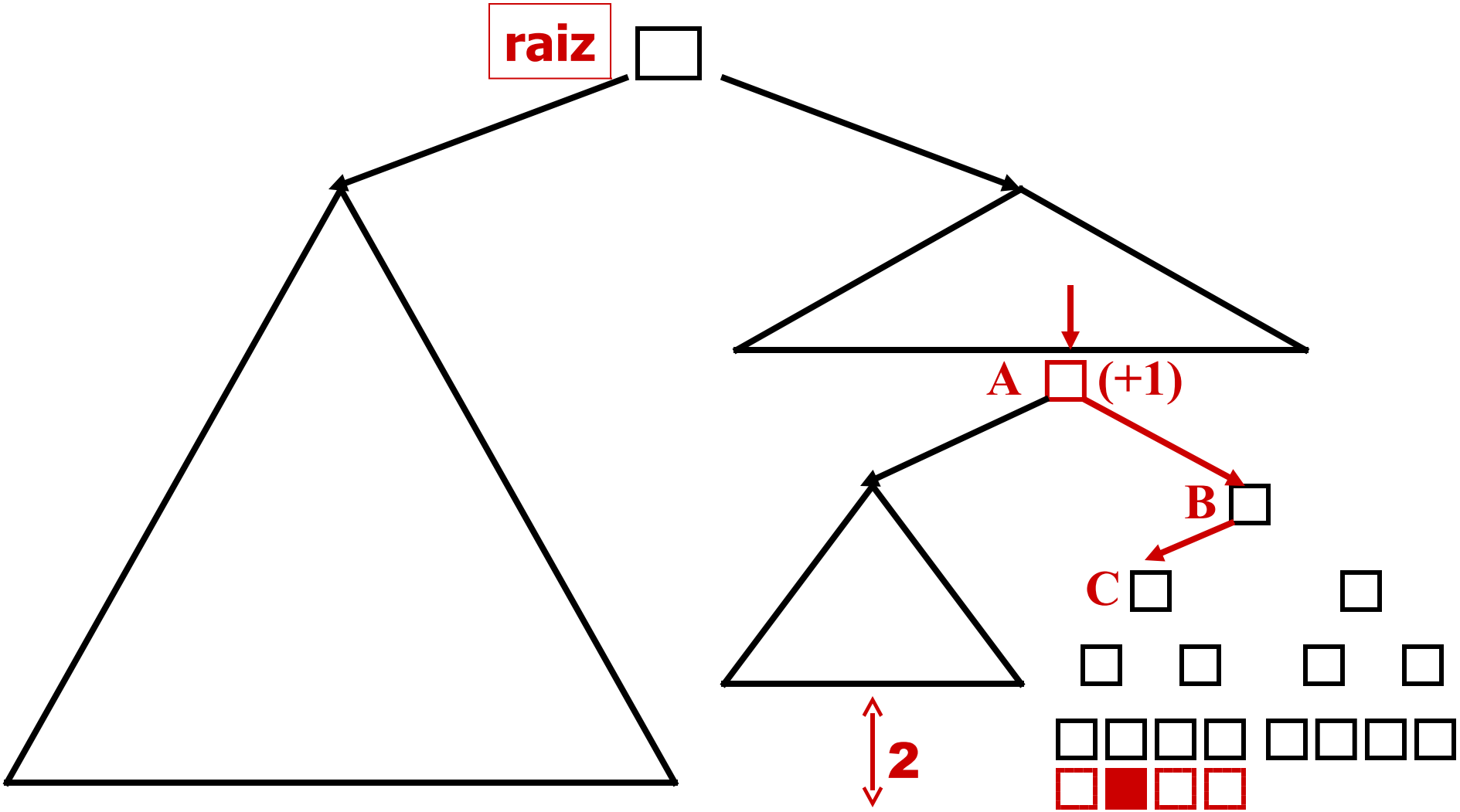
A altura da sub-árvore é igual à da original.



Qualquer que seja a posição de inclusão na sub-árvore de C, a árvore terá a propriedade AVL.

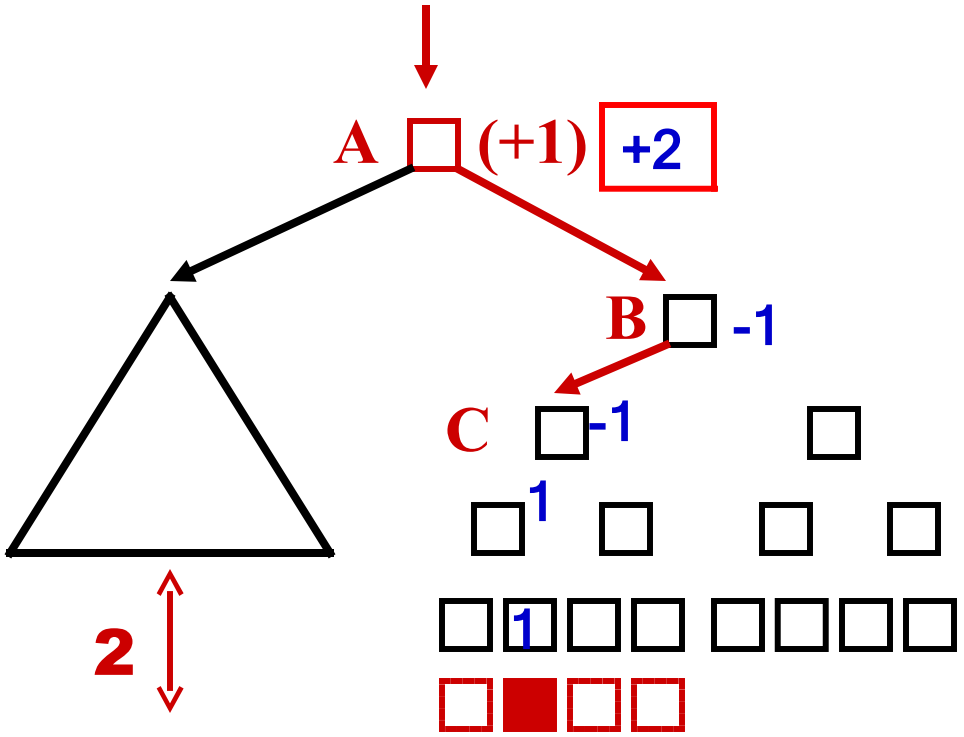
Rotação Dupla à Esquerda

Inserção à direita do nó crítico A com (bal = +1), à esquerda do nó B.



Rotação Dupla à Esquerda

Inserção à direita do nó crítico A com (bal = +1), à esquerda do nó B.



Rotação Dupla à Esquerda

Inserção à direita de nó crítico A com (bal = +1), à esquerda do nó B.

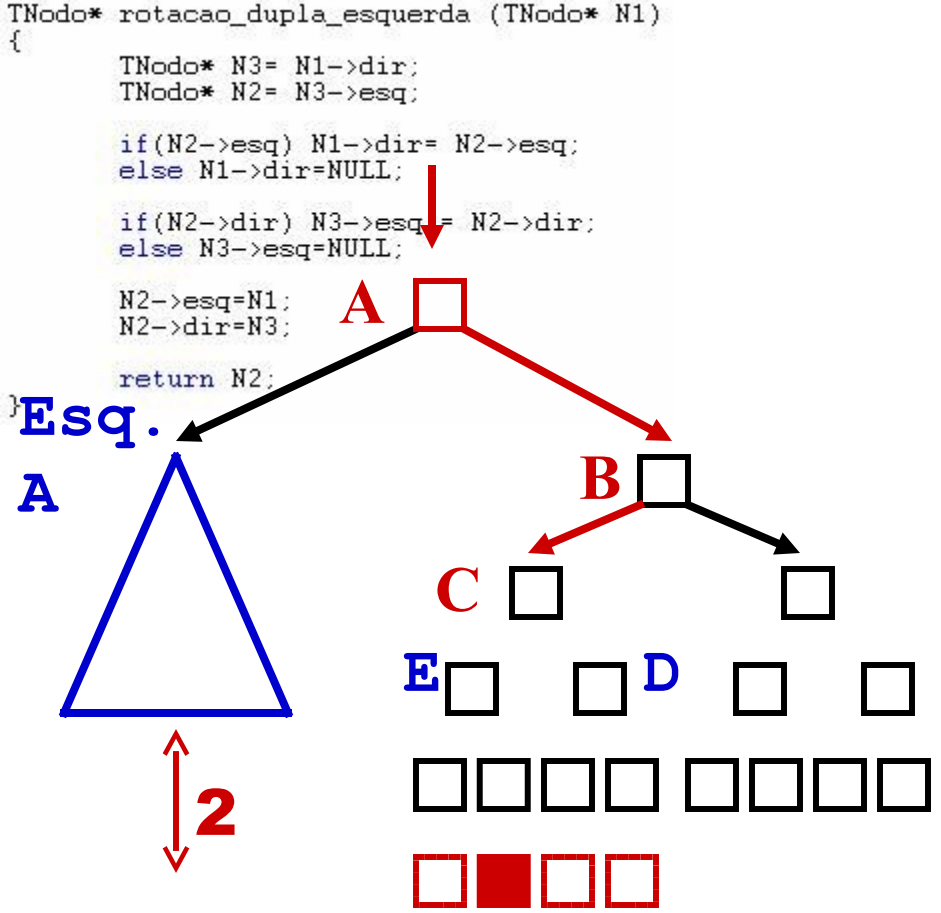
```
TNodo* rotacao_dupla_esquerda (TNodo* N1)
{
        TNodo* N3= N1->dir;
        TNodo* N2= N3->esq;

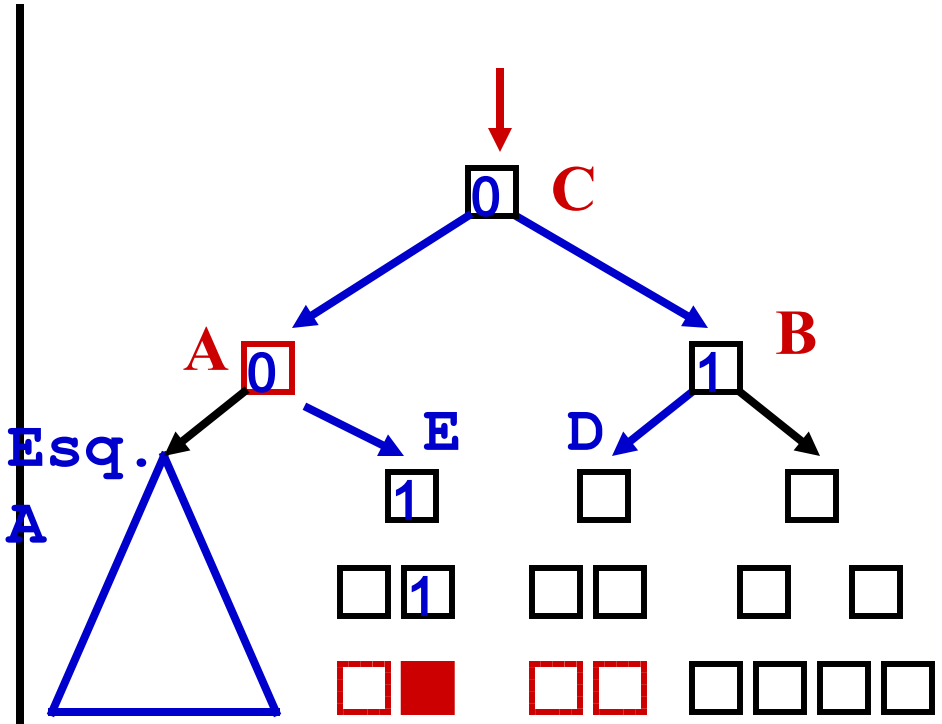
        if(N2->esq) N1->dir= N2->esq;
        else N1->dir=NULL;

        if(N2->dir) N3->esq= N2->dir;
        else N3->esq=NULL;

        N2->esq=N1;
        N2->dir=N3;

        return N2;
}
```



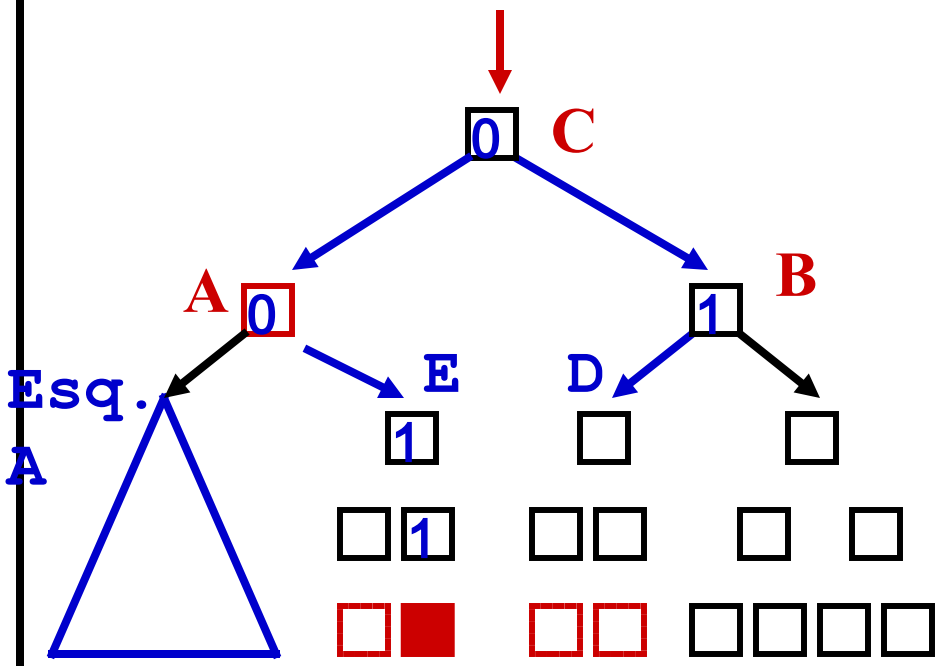


Rotação Dupla à Esquerda

Inserção à direita de nó crítico A com (bal = +1), à esquerda do nó B.

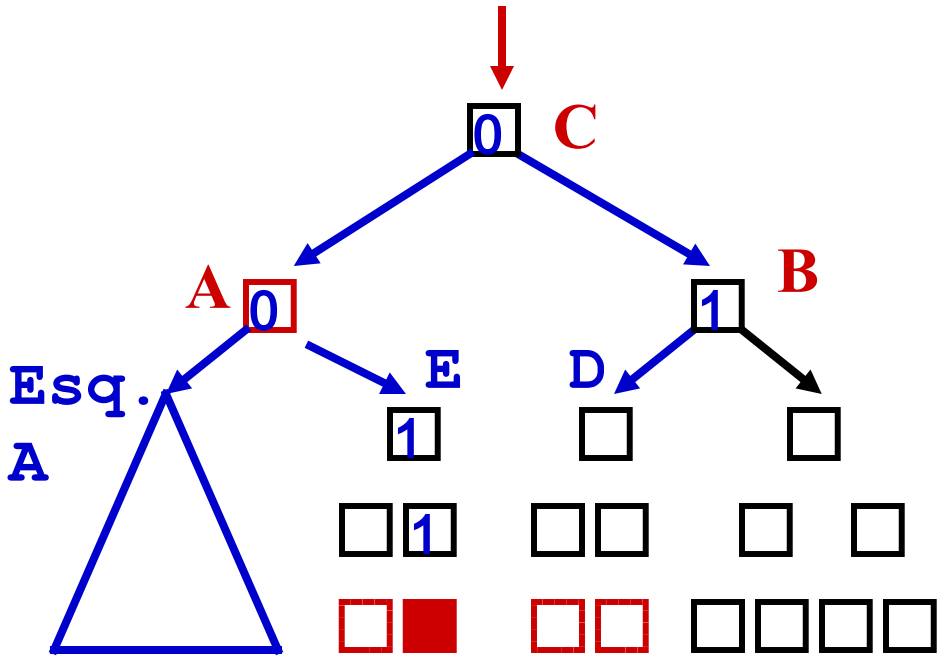
Qualquer que seja a posição de inclusão na sub-árvore de C, a árvore terá a propriedade AVL.

IMPORTANTE: Os balances de A e de B podem ser -1 e 0, respectivamente.



Rotação Dupla à Esquerda

Inserção à direita de nó crítico A com (bal = +1), à esquerda do nó B.



Os valores dos balances de cada nó são os mesmos, a menos daqueles nos nós A, B e C, que tomam valores dependendo se a inclusão foi à esquerda ou à direita de C.

Considerações para Implementação Recursiva

Encaminhamento top-down:

1. O processamento se inicia pela raiz.
2. O novo vértice inserido será sempre uma folha.
3. Esta nova folha tem *balance* = 0, e retorna ao pai a informação (Altura Alterada = V, <próprio endereço>)

Considerações para Implementação Recursiva

Encaminhamento bottom-up:

Se um nó pai receber a info: (V, <ender>)

- Calcular novo balance:

  Se filho à esq, então Bal ← Bal – 1

  senão Bal ← Bal +1

- Se Bal = 0 então

  devolver (F, <próprio ender>)

  Se Bal = +1 ou –1 então

  devolver (V, <próprio ender>)

  Se Bal = +2 ou –2 então {

  rotacionar;

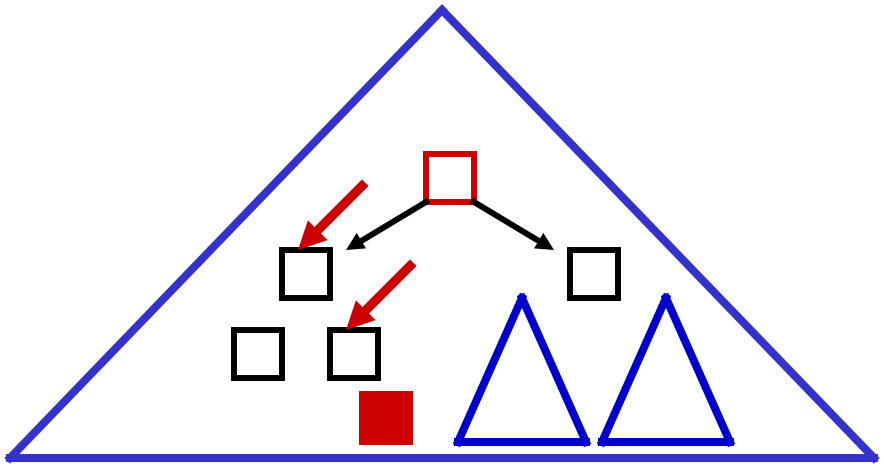
  devolver (F, <ender. nova raiz>) }

Considerações para Implementação Recursiva

Encaminhamento bottom-up:

Se Bal = +1 ou –1 então

devolver (V, <próprio ender>)

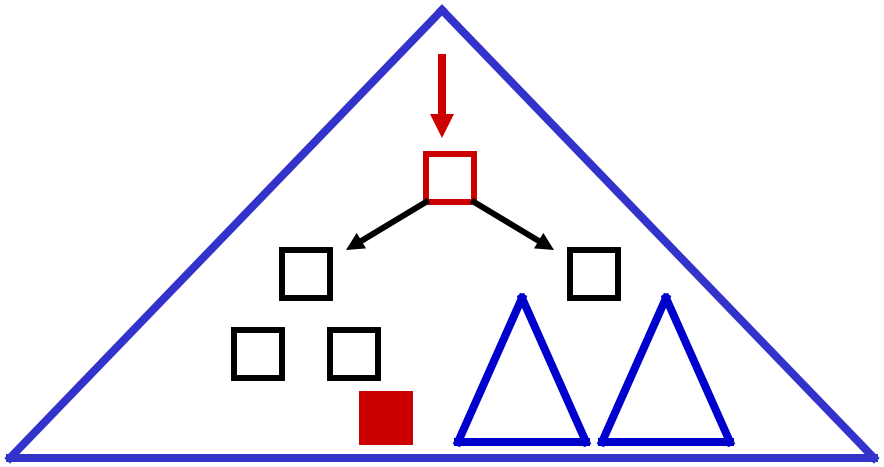


Considerações para Implementação Recursiva

Encaminhamento bottom-up:

Se Bal = 0 então

devolver (F, <próprio ender>)



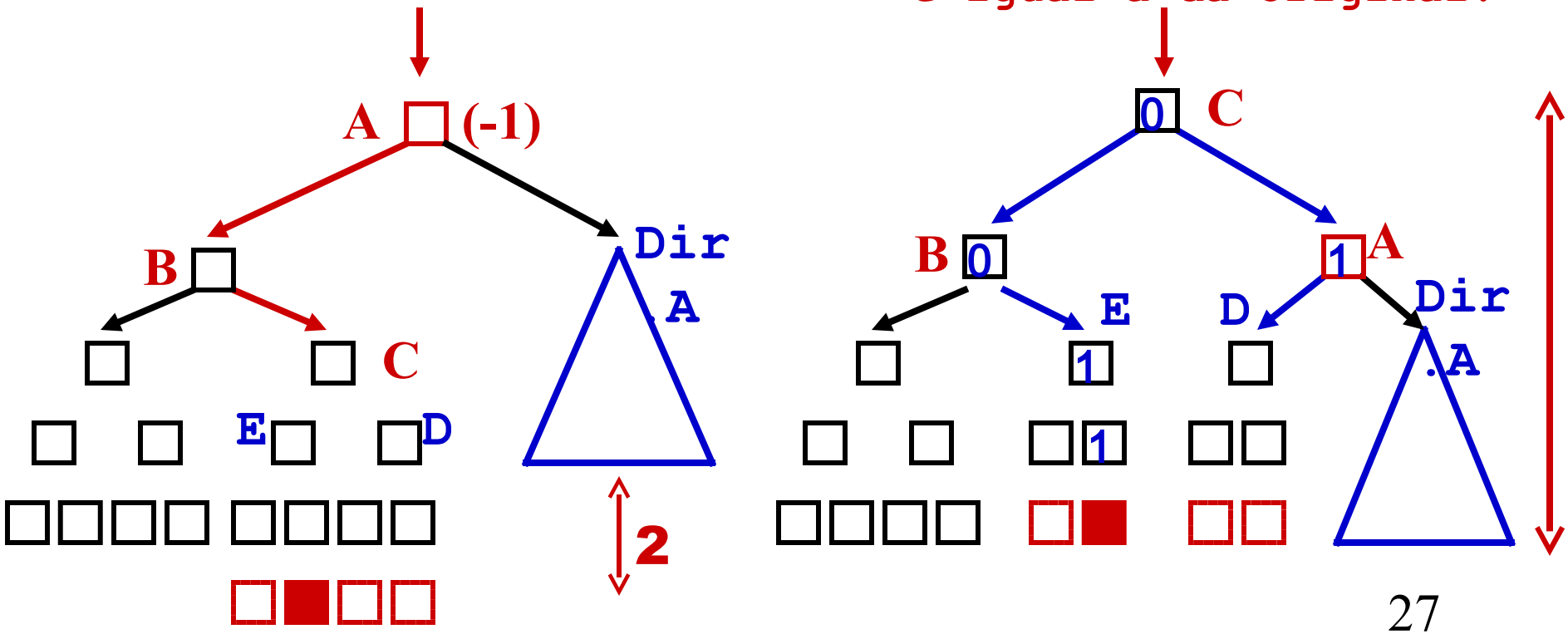
Considerações para Implementação Recursiva

Se Bal = +2 ou –2 então {

***rotacionar*;**

devolver (F, <ender. nova raiz>) }

A altura da sub-árvore é igual à da original.



Considerações para Implementação Recursiva

Encaminhamento bottom-up:

Se um nó pai receber a info: (F, <ender>)
então devolver (F, <próprio ender>)

