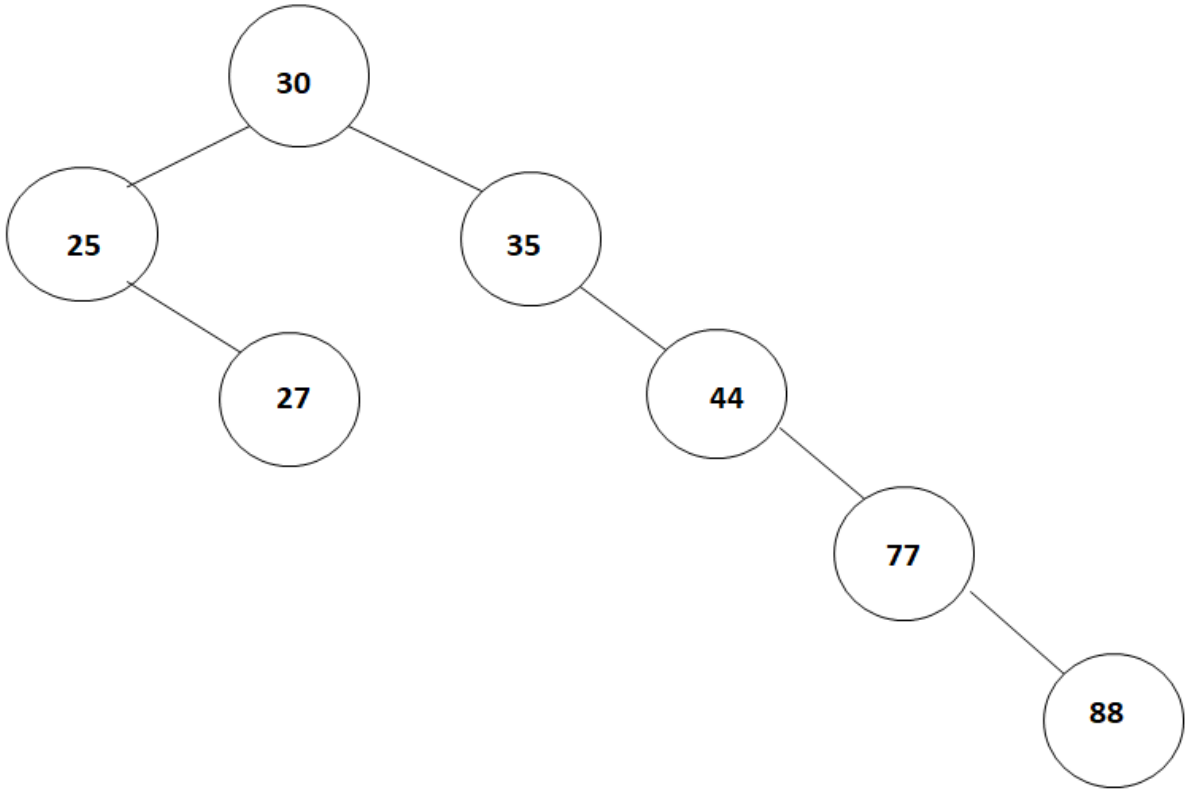
Balanceamento de Árvores Binárias: Árvores AVL

Códigos de Alta Performance

PROFa. PATRÍCIA MAGNA - profpatricia.magna@fiap.com.br

Motivação

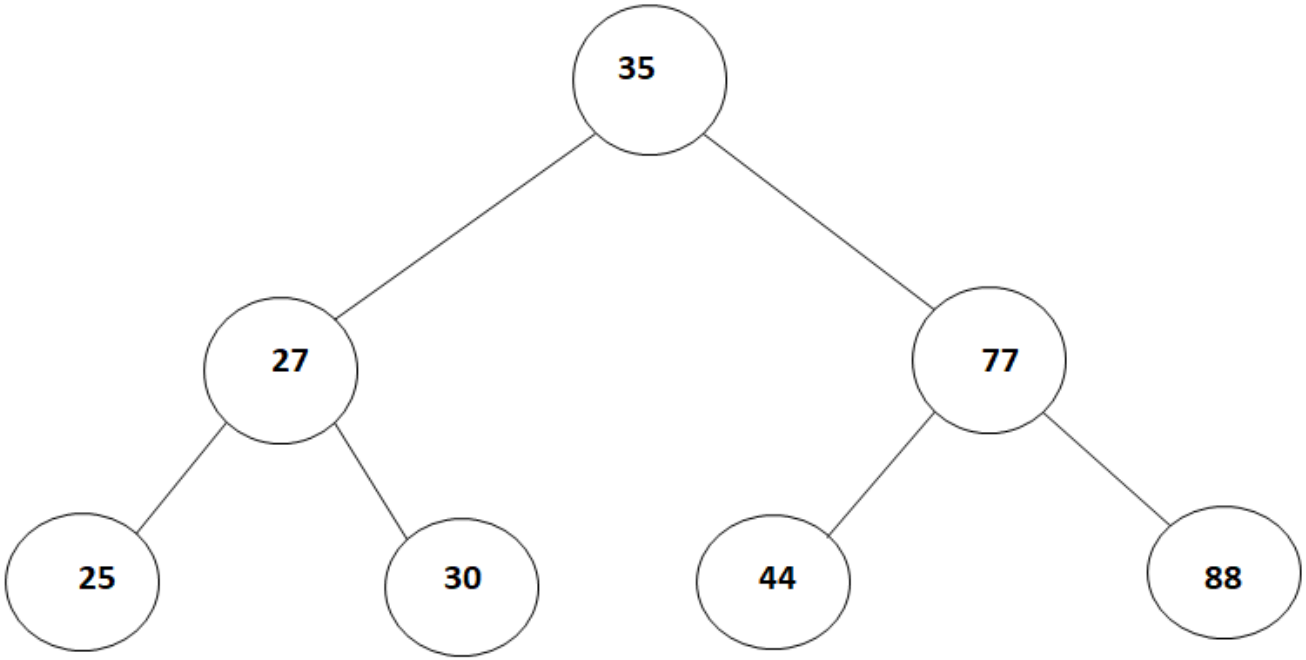
- A configuração de uma Árvore de Busca Binária é dependente da sequência de operações de inserção e remoção de valores.
- Suponha que os valores foram inseridos na ABB na seguinte sequência: 30, 25, 35, 44, 27, 77 e 88



Com apenas 7 nós podem ser necessárias até 5 comparações para acessar o nó com dado 88

Motivação

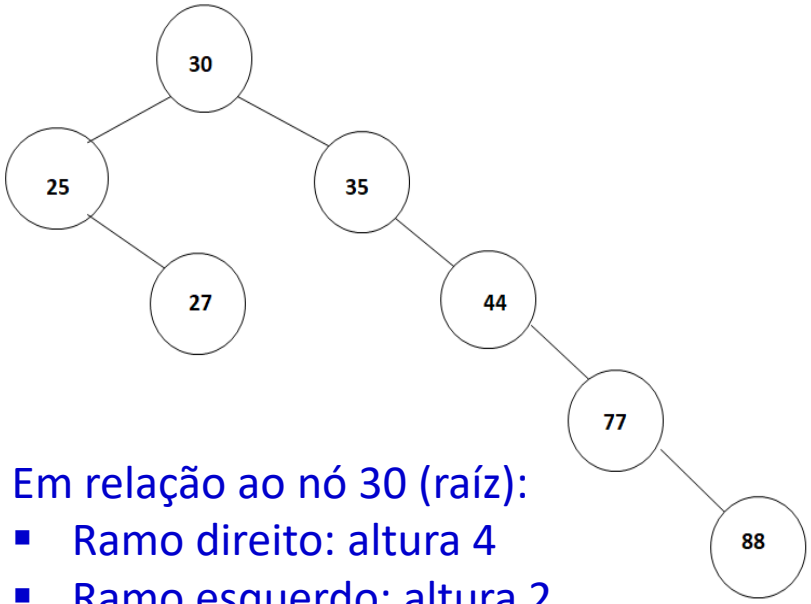
- Reorganizando os nós podemos obter a seguinte ABB:



Com apenas 7 nós podem ser necessárias até 3 comparações para acessar qualquer nó da ABB

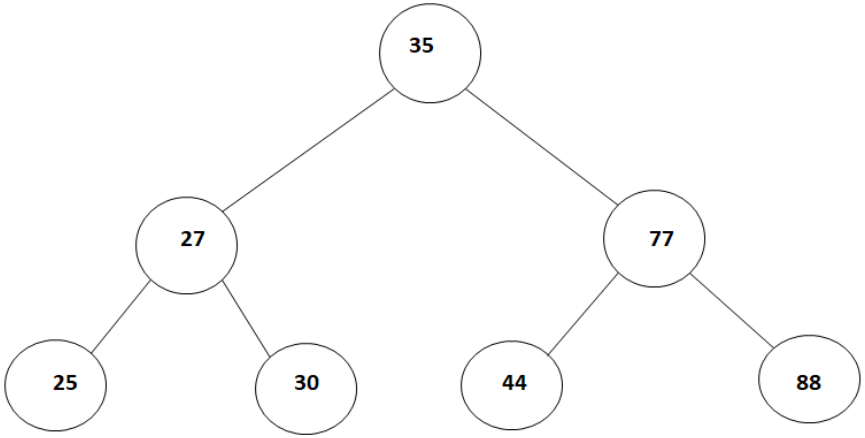
Definição de Altura em uma Árvore

A **altura** de um nó x em uma **árvore** binária é a distância entre x e o seu descendente mais afastado.



Em relação ao nó 30 (raíz):

- Ramo direito: altura 4
- Ramo esquerdo: altura 2



Em relação ao nó 35 (raíz):

- Ramo direito: altura 2
- Ramo esquerdo: altura 2

Árvore Balanceada

Recordando: Árvore Binária Completa

FIAP

$$N° \text{ nós} = 2^{\text{níveis}} - 1$$

Níveis	Número de Nós
3	7
4	15
5	31
10	1.023
11	2.047
20	1.048.575
21	2.097.151
27	134.217.727
28	268.435.455

Relação Altura da Árvore Binária e Números de Nós

Existe uma forma direta de saber quantos níveis uma árvore completa ou quase completa terá em função do número de nós, que é:

$$\boldsymbol{altura = log_2\,(Número\ de\ nós) + 1}$$

Para saber qual a altura de uma árvore binária balanceada com 6 nós, pelo cálculo teríamos $\log_2 6 = 2$ (aproximadamente 2.6), com a parte inteira 2,6 que é 2 e somado a 1, obtemos que o número de níveis para ter 6 nós em uma árvore balanceada é 3.

Recordando:

$\log_2 x = y$ uma vez que $2^y = x$

Balanceamento de Árvores

- Fica claro as vantagens de realizarmos a operação de balanceamento da árvore.
- E obviamente, também essa operação deve ser realizada de forma eficiente, pois senão não há vantagem em utilizá-la.

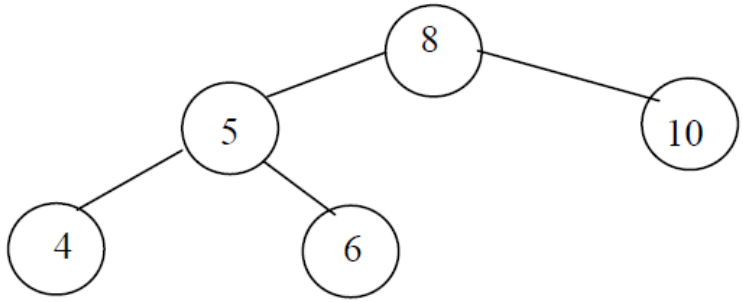
Balanceamento de Árvores de Busca Binária

- Algoritmos de balanceamento de árvore são chamados algoritmos AVL e as árvores geradas por esses algoritmos são denominadas de **árvores AVL**.
- Esse tipo de árvore surgiu em 1962 com matemáticos russos G.M. Adelson-Velskki e E.M.Landis que sugeriram uma definição para "*near balance*" e descreveram procedimentos para inserção e eliminação de nós nessas árvores.

Uma árvore AVL é uma árvore de busca binária (ABB) construída de tal modo que a **altura** de sua sub-árvore direita **difere** da altura da sub-árvore esquerda de no **máximo 1**.

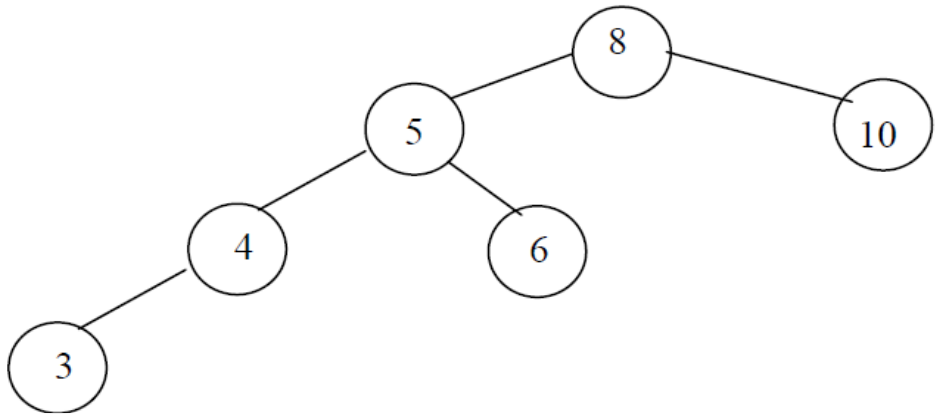
Construindo Programa para Implementar uma Árvore AVL

- Suponha a ABB ao lado
- Árvore está balanceada
  - diferença de altura (níveis) de qualquer nó é no máximo 1



A **altura** de um nó x em uma **árvore** binária é a distância entre x e o seu descendente mais afastado.

- Inserindo o nó com valor 3
- Árvore NÃO está balanceada
  - Em relação ao nó 8 a diferença de altura entre as suas sub-árvores é 2.



Considerações necessárias para balancear ABB

- Após a inserção de um novo nó, por exemplo, no ramo esquerdo, pode-se passar a ter 3 situações com relação à altura da sub-árvore esquerda (**hEsq**) e a altura da sub-árvore direita (**hDir**):

  - Se hEsq < hDir, então subárvores esquerda e direita ficam com alturas diferentes, mas continuam balanceadas.
  - Se hEsq = hDir, então subárvores esquerda e direita ficam com alturas iguais e balanceamento foi melhorado.
  - Se hEsq > hDir, então subárvore esquerda fica ainda maior e balanceamento foi violado.

- Para saber em qual caso ficará a árvore com a nova inserção é preciso utilizar uma informação sobre as alturas das sub-árvores de cada nó.

Fator de Balanceamento

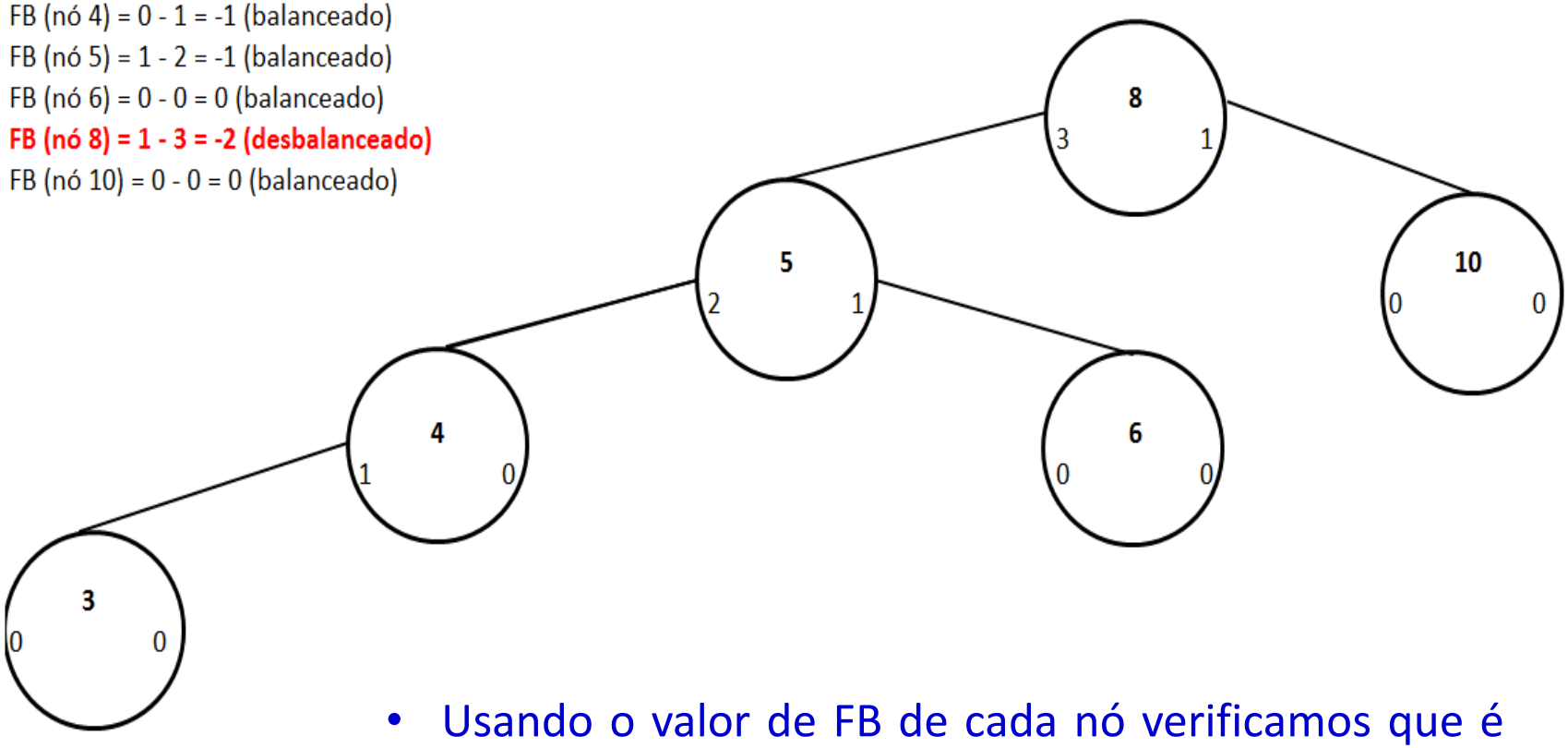
- Para verificar se a árvore continua balanceada é preciso calcular o **Fator de Balanceamento (FB)** de um nó.
- FB é a altura da sub-árvore direita do nó subtraído da altura da sub-árvore esquerda do nó

(FB = hDir – hEsq)

- Se o FB de um nó for maior do que 2 ou menor do que –2 a árvore está desbalanceada.

Exemplo 1: ABB com FB de cada nó

FB (nó 3) = 0 - 0 = 0 (balanceado)
FB (nó 4) = 0 - 1 = -1 (balanceado)
FB (nó 5) = 1 - 2 = -1 (balanceado)
FB (nó 6) = 0 - 0 = 0 (balanceado)
FB (nó 8) = 1 - 3 = -2 (desbalanceado)
FB (nó 10) = 0 - 0 = 0 (balanceado)



- Usando o valor de FB de cada nó verificamos que é necessário realizar um “rebalanceamento”.
  - É preciso fazer a rotação dos nós a fim de deixar a árvore balanceada novamente.

Alteração da Estrutura de Nó de uma ABB

Cada nó de uma AVL deve conter mais 2 atributos, a altura de cada ramificação.

```
private class ARVORE{
     int dado;
     ARVORE dir;
     ARVORE esq;
     int hEsq ;
     int hDir;
    }
```

Inserção de Nó Armazenando Alturas das Sub-Árvores

```
public ARVORE inserirH(ARVORE p, int info) {
if (p == null) { //nó inserido sempre será nó folha
          p=new ARVORE();
          p.dado = info;
          p.esq = null;
          p.dir = null;
          p.hDir=0;
          p.hEsq=0;
     }
else if (p.dado > info){
          p.esq= inserirH (p.esq, info);
          if (p.esq.hDir > p.esq.hEsq) //Altura do nó será a maior
              p.hEsq = p.esq.hDir + 1; //altura dos seus filhos
          else
              p.hEsq = p.esq.hEsq + 1;
          }
      else {
          p.dir=inserirH(p.dir, info);
          if (p.dir.hDir > p.dir.hEsq)
              p.hDir = p.dir.hDir + 1;
          else
              p.hDir = p.dir.hEsq + 1;
         }
}
return p;
}
```

Exercícios

1. Elabore o método **mostraFB()** que apresenta o Fator de Balanceamento de cada nó que compõe uma ABB.
2. Construa um programa que crie uma ABB para armazenar valores inteiros e que possua um menu com as seguintes opções:

0 - Sair do programa

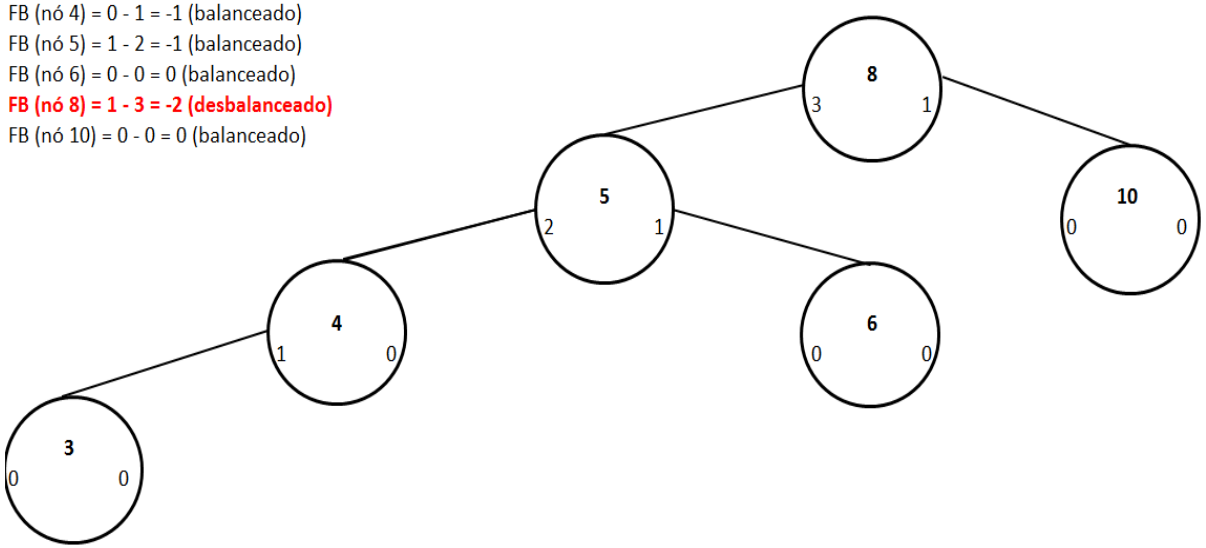
1 - Insere 1 valor na ABB;

2 - Apresenta pós ordem os nós da ABB apresentando também o FB do nó;

Exemplo Balanceamento de uma ABB – Algoritmos AVL

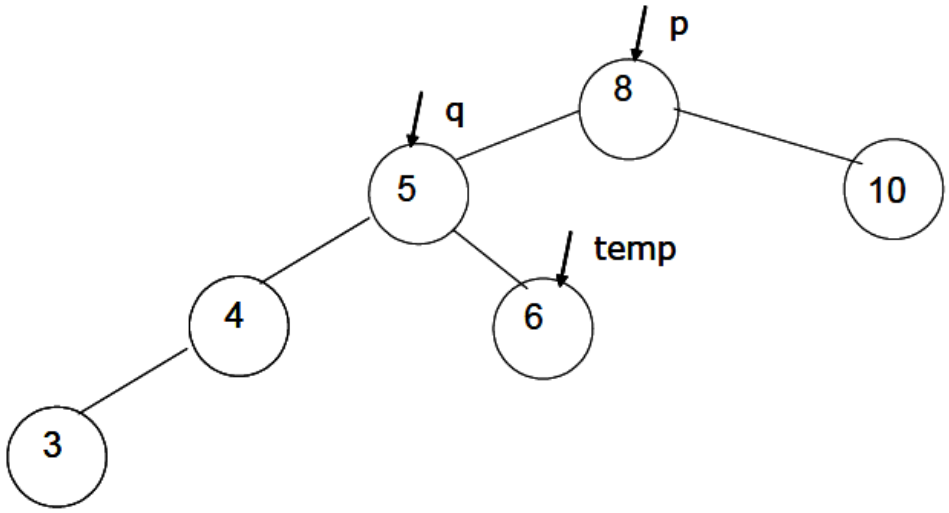
- O nó 8 tem FB=1 – 3 = **-2**
- Para decidir o que deve ser feito para a árvore voltar ao estado balanceada ainda é preciso analisar como está o balanceamento do seu filho que está na sub-árvore com maior altura.
  - nó 5 tem FB=1 – 2 = **-1**.
- **Ambos FBs** são de **negativos** significa que a inserção do nó 3 deixou a árvore com maior altura apenas no ramo esquerdo tanto em relação ao nó 8 como também em relação ao seu nó filho (nó 5).
- Assim, deve-se fazer a **rotação para a direita**, já que FB é negativo, dos nós a **partir do nó 8**

FB (nó 3) = 0 - 0 = 0 (balanceado)
FB (nó 4) = 0 - 1 = -1 (balanceado)
FB (nó 5) = 1 - 2 = -1 (balanceado)
FB (nó 6) = 0 - 0 = 0 (balanceado)
FB (nó 8) = 1 - 3 = -2 (desbalanceado)
FB (nó 10) = 0 - 0 = 0 (balanceado)



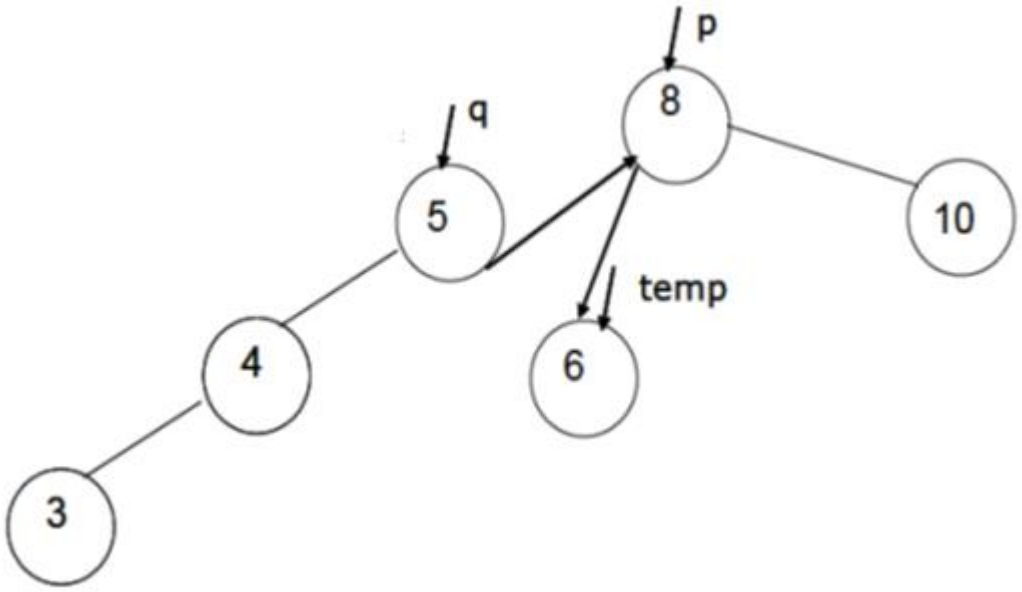
Método para Efetuar Rotação para Direita de Nós

```
public ARVORE rotacaoDireita (ARVORE p){
// faz rotação para direita em relação ao nó apontado por p
    ARVORE q,temp;
    q = p.esq;
    temp = q.dir;
    q.dir = p;
    p.esq = temp;
    return q;
}
```



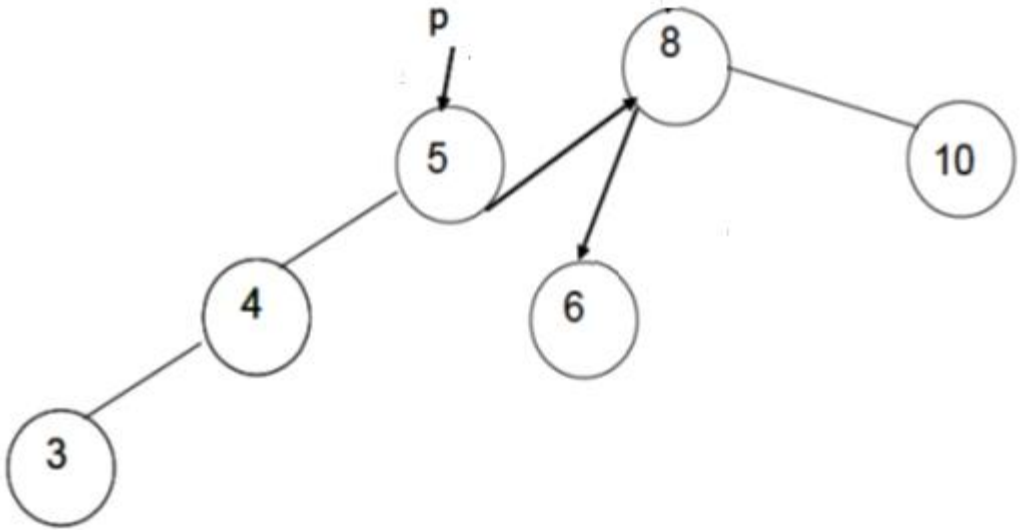
Método para Efetuar Rotação para Direita de Nós

```
public ARVORE rotacaoDireita (ARVORE p){
// faz rotação para direita em relação ao nó apontado por p
    ARVORE q,temp;
    q = p.esq;
    temp = q.dir;
    q.dir = p;
    p.esq = temp;
    return q;
}
```



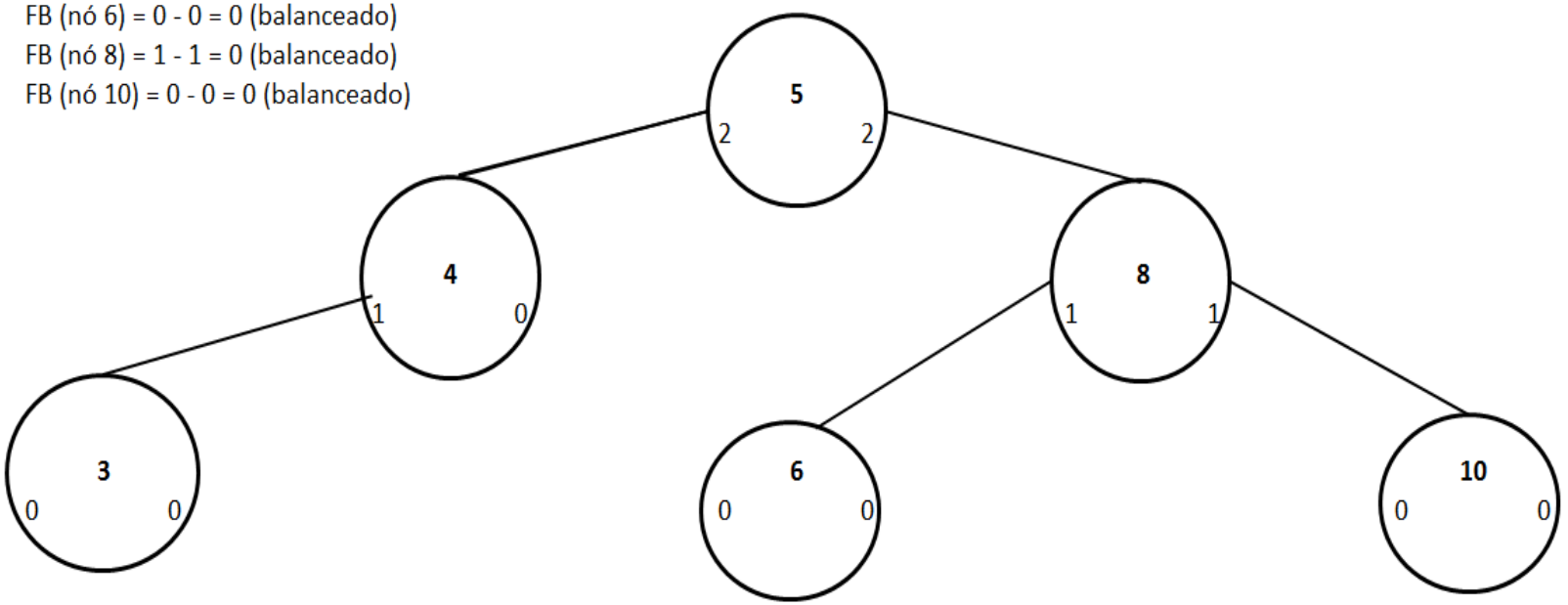
Método para Efetuar Rotação para Direita de Nós

```
public ARVORE rotacaoDireita (ARVORE p){
// faz rotação para direita em relação ao nó apontado por p
    ARVORE q,temp;
    q = p.esq;
    temp = q.dir;
    q.dir = p;
    p.esq = temp;
    return q;
}
```



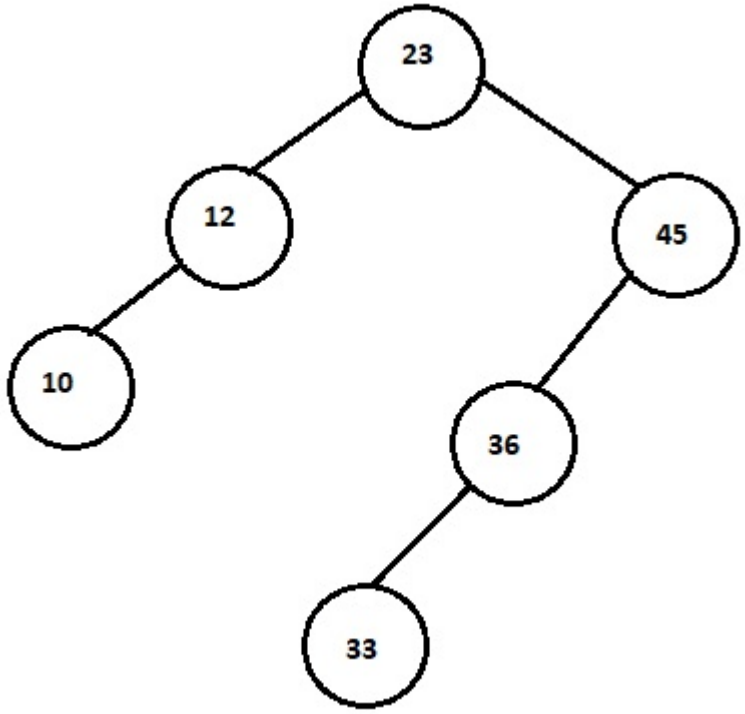
Após a Rotação para Direita – Árvore Balanceada

FB (nó 3) = 0 - 0 = 0 (balanceado)
FB (nó 4) = 0 - 1 = -1 (balanceado)
FB (nó 5) = 2 - 2 = 0 (balanceado)
FB (nó 6) = 0 - 0 = 0 (balanceado)
FB (nó 8) = 1 - 1 = 0 (balanceado)
FB (nó 10) = 0 - 0 = 0 (balanceado)



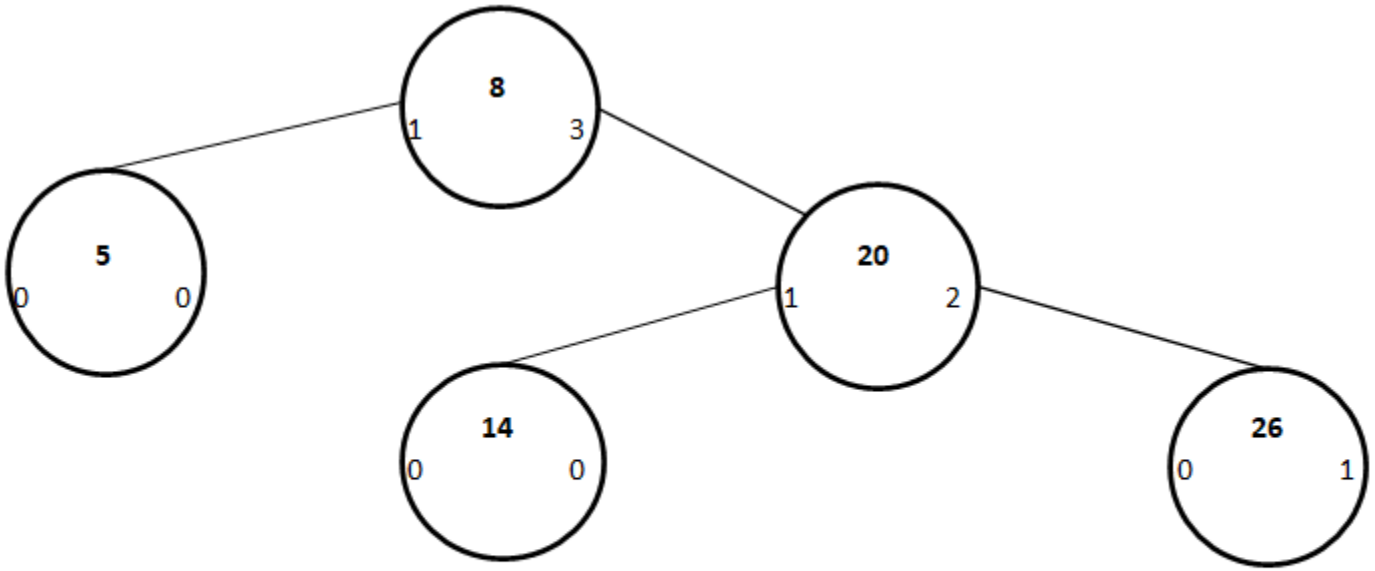
Exercício Conceitual

1. A árvore a seguir teve como última inserção o nó com dado 33, deixando-a conforma a figura a seguir.
   a) Qual o FB de cada nó presente na árvore?
   b) Sabendo que deve ser feita uma rotação para a direita em relação a algum nó (definido pelo FB desse nó) como fica a AVL?



Exemplo 2: AVL com FB de cada nó

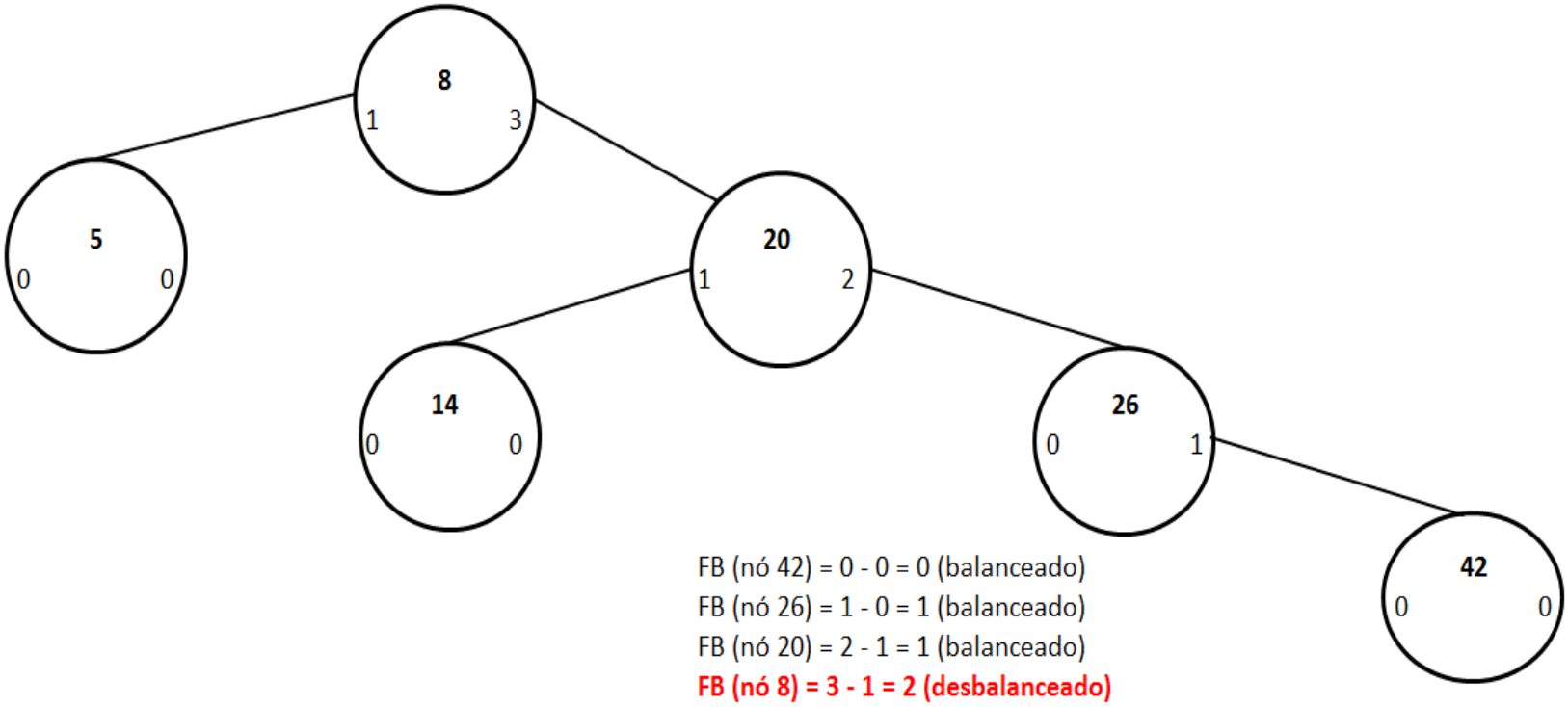
- Considere a AVL (balanceada)
- Depois, deve ter o nó com dado 42 a ser inserido



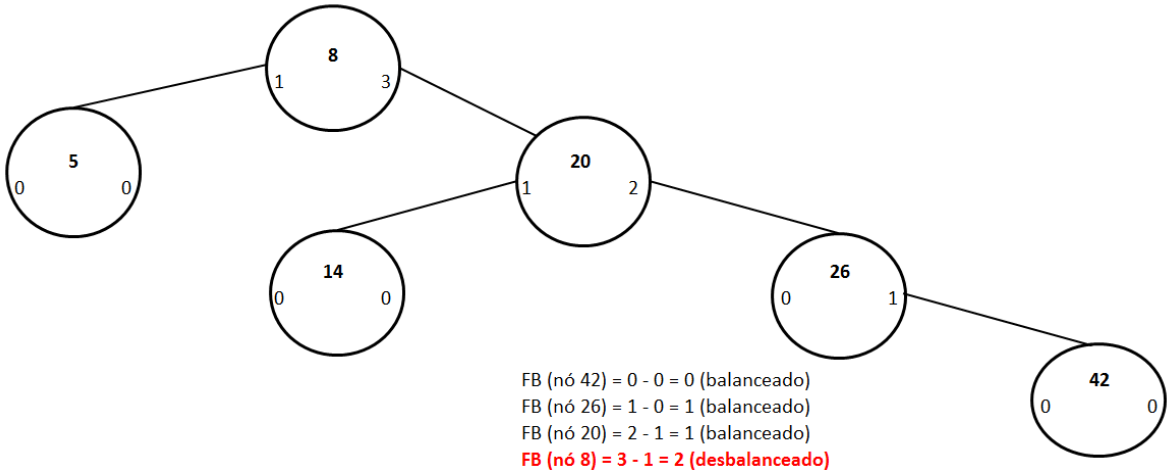
FB (nó 26) = 1 - 0 = 1 (balanceado)
FB (nó 20) = 2 - 1 = 1 (balanceado)
FB (nó 14) = 0 - 0 = 0 (balanceado)
FB (nó 8) = 2 - 1 = 1 (balanceado)
FB (nó 5) = 0 - 0 = 0 (balanceado)

Exemplo 2: Após uma inserção em AVL

- A configuração da AVL depois da inserção a deixa **não** balanceada



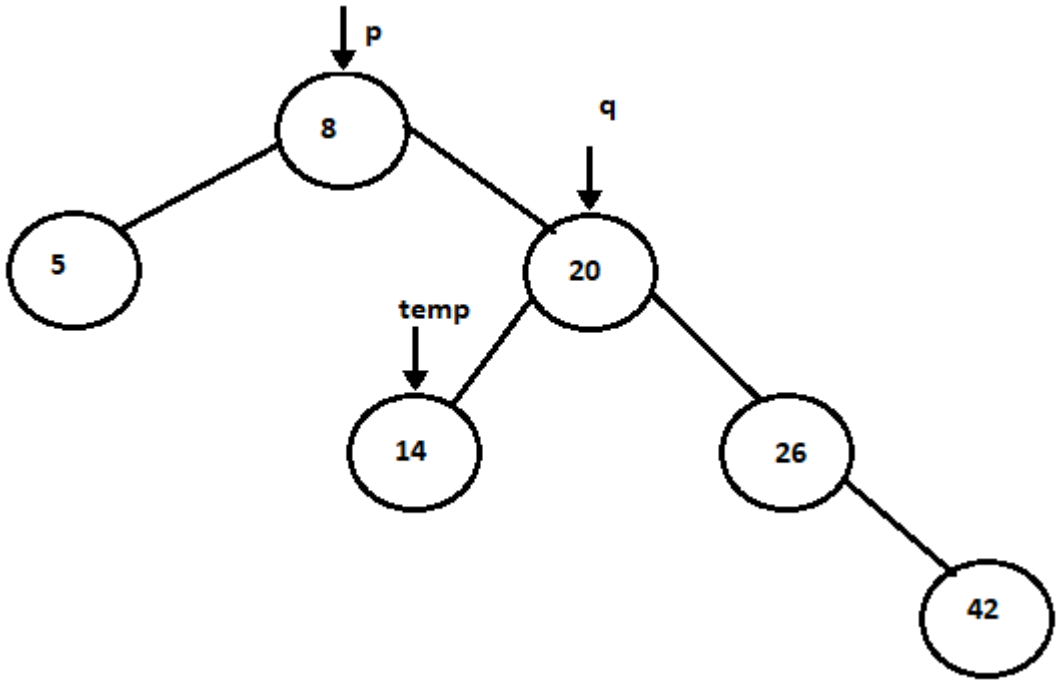
Algoritmo AVL para Balancear a Árvore



- O **FB (nó 8)** tem FB=1 – 3 = **2**
- É preciso também analisar FB do nó que está à direita do nó 8 (que é o ramo com maior altura).
  - Determinamos que **FB (nó 20)** = 2 - 1 = **1**
- Como os **2 nós têm FB positivos** significa que a inserção do nó 42 deixou a árvore com maior altura apenas no ramo direito tanto em relação ao nó 8 como também em relação ao seu nó filho (nó 20).
- A operação a ser realizada para efetuar o balanceamento é **uma rotação para a esquerda em relação ao nó 8**.

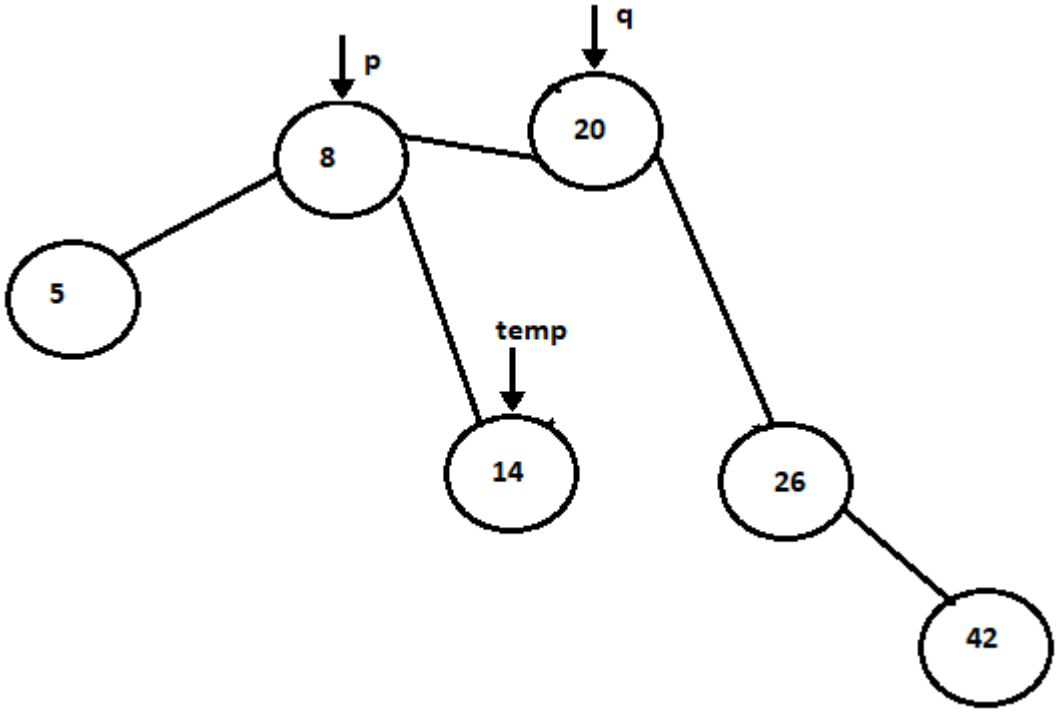
Método para Efetuar Rotação para Esquerda de Nós

```
public ARVORE rotacaoEsquerda(ARVORE p) {
// faz rotação para esquerda em relação ao nó apontado por p
   ARVORE q,temp;
   q = p.dir;
   temp = q.esq;
   q.esq = p;
   p.dir = temp;
   return q;
   }
```



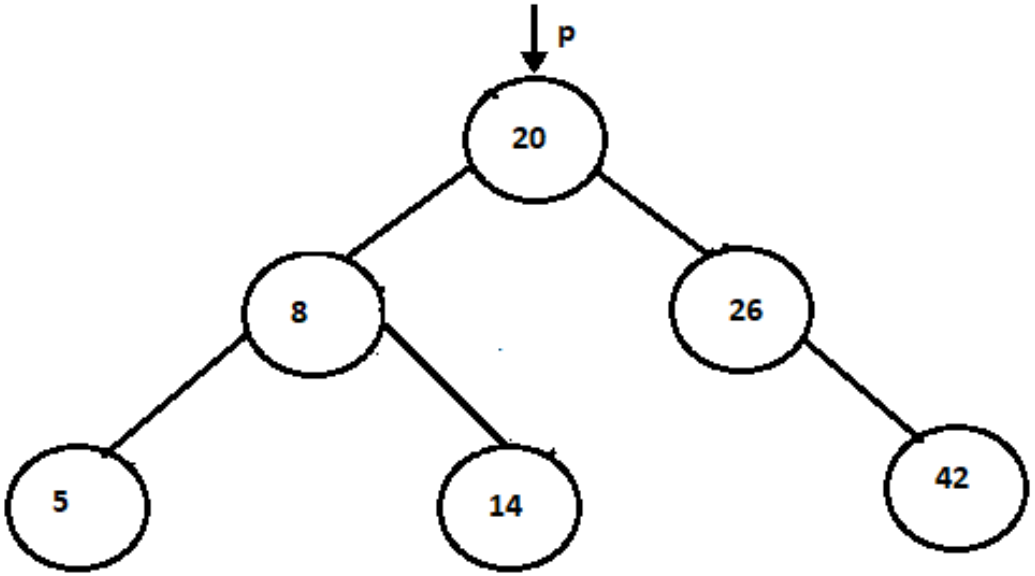
Método para Efetuar Rotação para Esquerda de Nós

```
public ARVORE rotacaoEsquerda(ARVORE p) {
// faz rotação para esquerda em relação ao nó apontado por p
    ARVORE q,temp;
    q = p.dir;
    temp = q.esq;
    q.esq = p;
    p.dir = temp;
    return q;
    }
```



Método para Efetuar Rotação para Esquerda de Nós

```
public ARVORE rotacaoEsquerda(ARVORE p) {
// faz rotação para esquerda em relação ao nó apontado por p
    ARVORE q,temp;
    q = p.dir;
    temp = q.esq;
    q.esq = p;
    p.dir = temp;
    return q;
    }
```

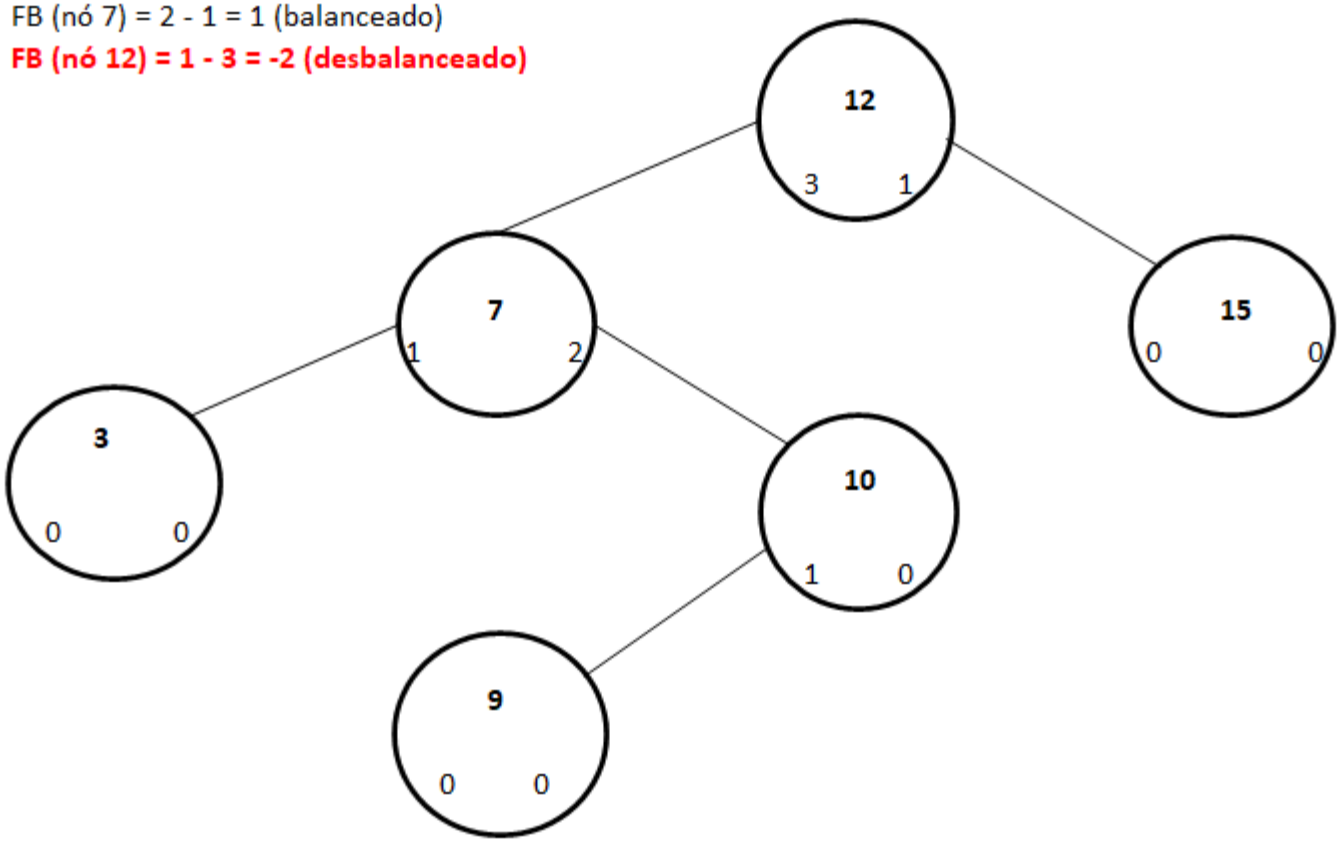


Regras para Decisão de Rotações AVL

FB do Nó	FB do Nó Filho da sub-árvore com maior altura	Operações
2	1	Rotação para esquerda em relação ao nó pai
	0	
	-1	Rotação para direita em relação ao nó filho Rotação para esquerda em relação ao nó pai
-2	1	Rotação para esquerda em relação ao nó filho Rotação para direita em relação ao nó pai
	0	Rotação para direita em relação ao nó pai
	-1	

Exemplo 3: Após uma inserção em AVL

FB (nó 9) = 0 - 0 = 0 (balanceado)
FB (nó 10) = 0 -1 = -1 (balanceado)
FB (nó 7) = 2 - 1 = 1 (balanceado)
FB (nó 12) = 1 - 3 = -2 (desbalanceado)



Regras para Decisão de Rotações AVL

FB do Nó	FB do Nó Filho da sub-árvore com maior altura	Operações
2	1	Rotação para esquerda em relação ao nó pai
	0	
	-1	Rotação para direita em relação ao nó filho Rotação para esquerda em relação ao nó pai
-2	**1**	**Rotação para esquerda em relação ao nó filho** **Rotação para direita em relação ao nó pai**
	0	Rotação para direita em relação ao nó pai
	-1	

FB (nó 9) = 0 - 0 = 0 (balanceado)

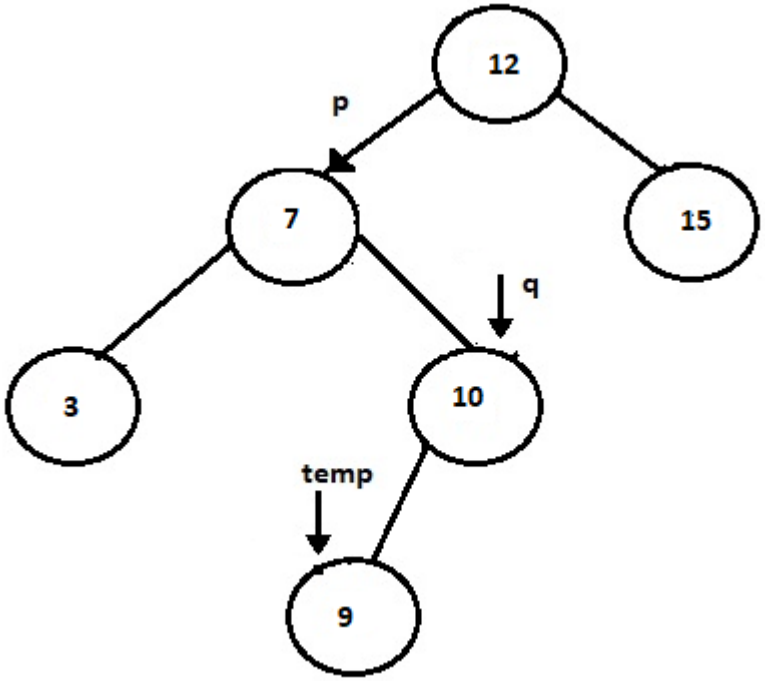
FB (nó 10) = 0 -1 = -1 (balanceado)

FB (nó 7) = 2 - 1 = 1 (balanceado)

FB (nó 12) = 1 - 3 = -2 (desbalanceado)

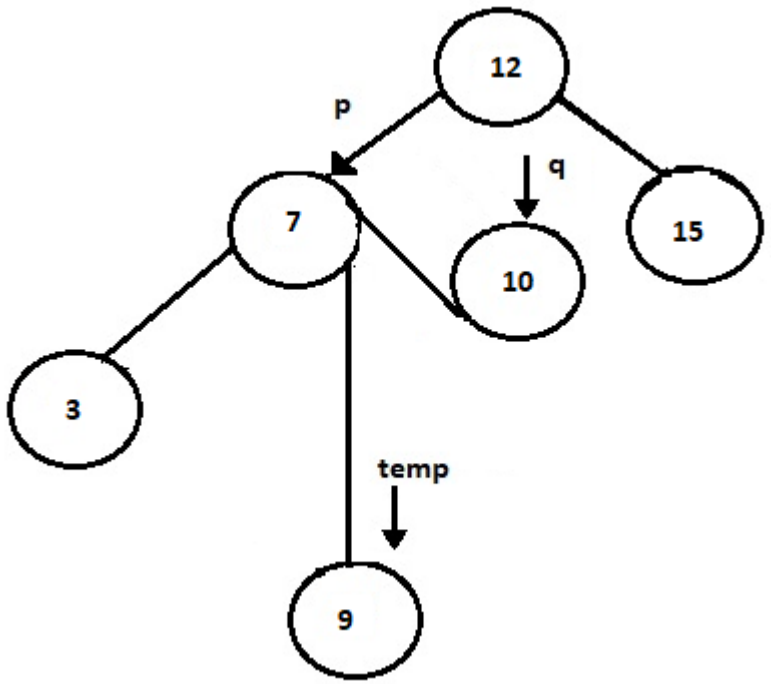
Rotação para Esquerda em Relação ao Nó 7

```
public ARVORE rotacaoEsquerda(ARVORE p) {
// faz rotação para esquerda em relação ao nó apontado por p
    ARVORE q,temp;
    q = p.dir;
    temp = q.esq;
    q.esq = p;
    p.dir = temp;
    return q;
    }
```



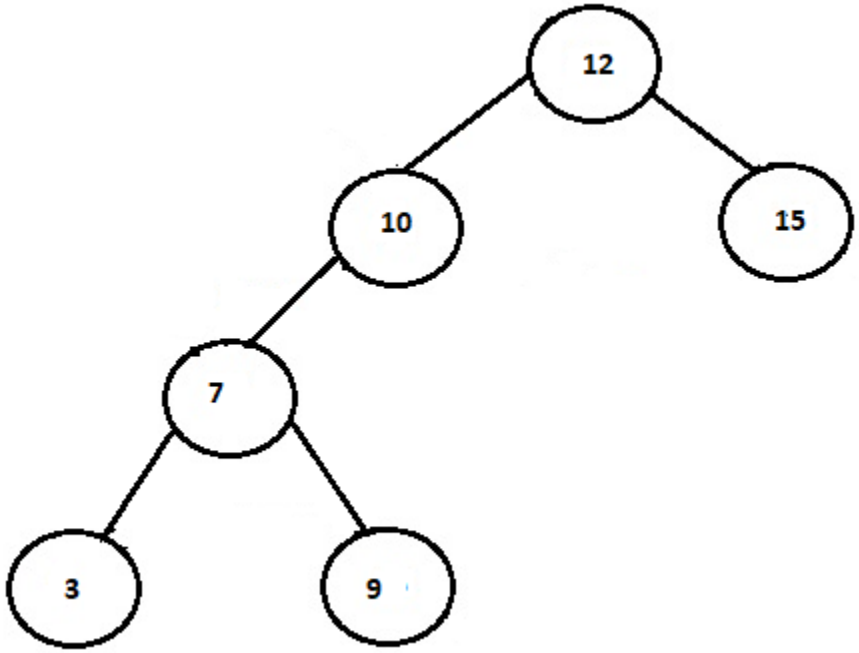
Rotação para Esquerda em Relação ao Nó 7

```
public ARVORE rotacaoEsquerda(ARVORE p) {
// faz rotação para esquerda em relação ao nó apontado por p
    ARVORE q,temp;
    q = p.dir;
    temp = q.esq;
    q.esq = p;
    p.dir = temp;
    return q;
    }
```



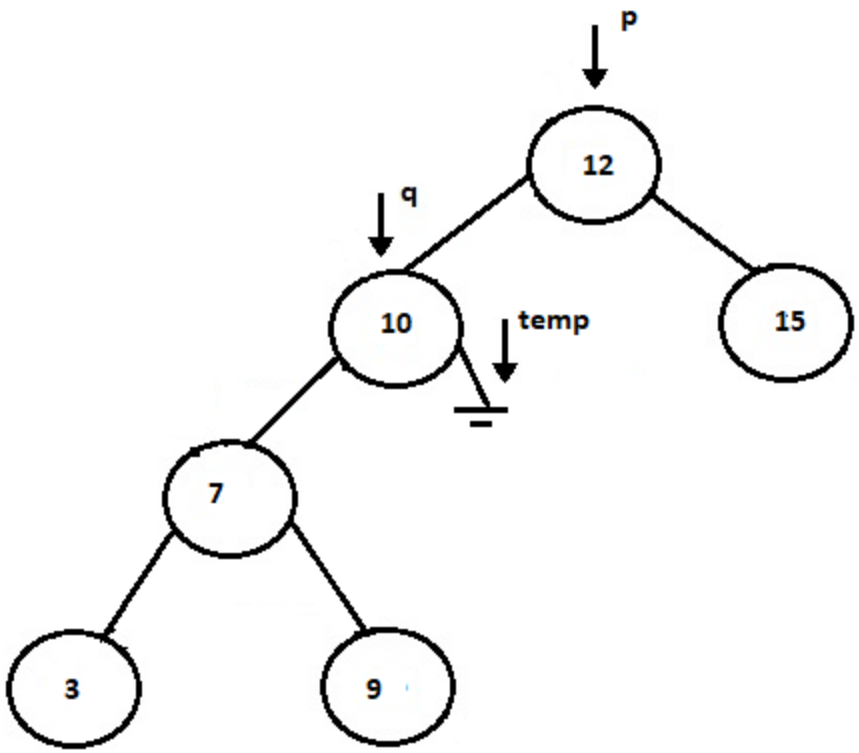
Rotação para Esquerda em Relação ao Nó 7

```
public ARVORE rotacaoEsquerda(ARVORE p) {
// faz rotação para esquerda em relação ao nó apontado por p
    ARVORE q,temp;
    q = p.dir;
    temp = q.esq;
    q.esq = p;
    p.dir = temp;
    return q;
    }
```



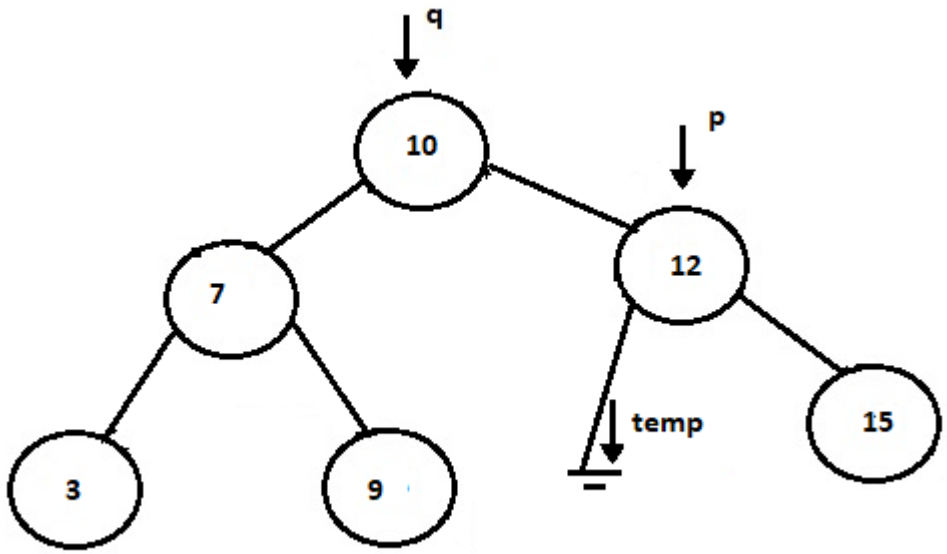
Rotação para Direita em Relação ao Nó 12

```
public ARVORE rotacaoDireita (ARVORE p){
// faz rotação para direita em relação ao nó apontado por p
    ARVORE q,temp;
    q = p.esq;
    temp = q.dir;
    q.dir = p;
    p.esq = temp;
    return q;
}
```



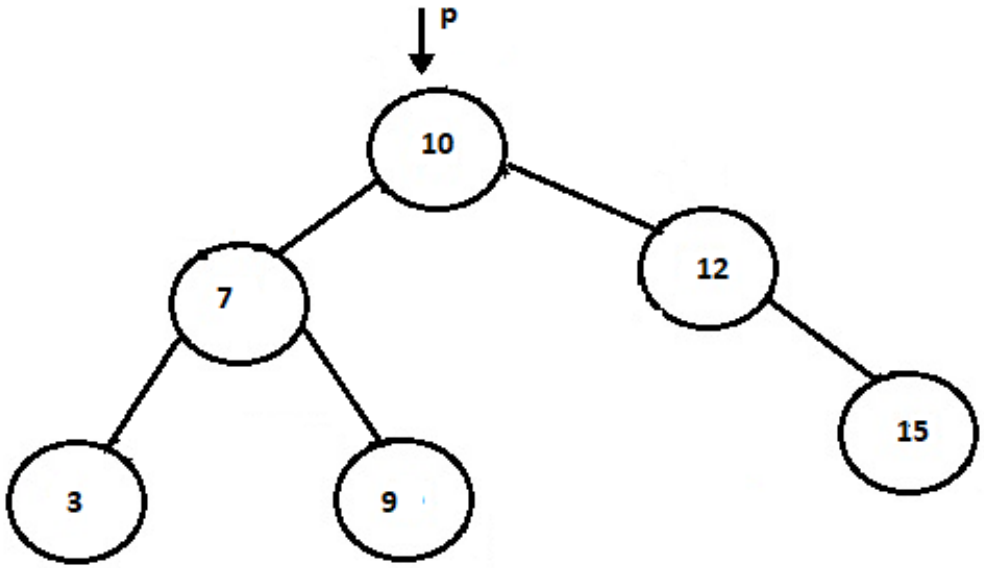
Rotação para Direita em Relação ao Nó 12

```
public ARVORE rotacaoDireita (ARVORE p){
// faz rotação para direita em relação ao nó apontado por p
    ARVORE q,temp;
    q = p.esq;
    temp = q.dir;
    q.dir = p;
    p.esq = temp;
    return q;
}
```



Rotação para Direita em Relação ao Nó 12

```
public ARVORE rotacaoDireita (ARVORE p){
// faz rotação para direita em relação ao nó apontado por p
    ARVORE q,temp;
    q = p.esq;
    temp = q.dir;
    q.dir = p;
    p.esq = temp;
    return q;
}
```



Regras para Decisão de Rotações AVL

FB do Nó	FB do Nó Filho da sub-árvore com maior altura	Operações
2	1	Rotação para esquerda em relação ao nó pai
	0	
	-1	Rotação para direita em relação ao nó filho Rotação para esquerda em relação ao nó pai
-2	1	Rotação para esquerda em relação ao nó filho Rotação para direita em relação ao nó pai
	0	Rotação para direita em relação ao nó pai
	-1	

Método balanceamento()

```
public ARVORE balanceamento (ARVORE p) {
// analisa FB e realiza rotações necessárias para balancear árvore
int FB = p.hDir - p.hEsq;
if (FB > 1) {
    int fbFilhoDir = p.dir.hDir - p.dir.hEsq;
    if (fbFilhoDir >= 0)
          p = rotacaoEsquerda(p);
    else {
          p.dir = rotacaoDireita(p.dir);
          p = rotacaoEsquerda(p);
          }
    }
else {
    if (FB < -1) {
        int fbFilhoEsq = p.esq.hDir - p.esq.hEsq;
        if (fbFilhoEsq <= 0)
            p = rotacaoDireita(p);
        else {
            p.esq = rotacaoEsquerda(p.esq);
            p = rotacaoDireita(p);
            }
        }
    }
return p;
}
```

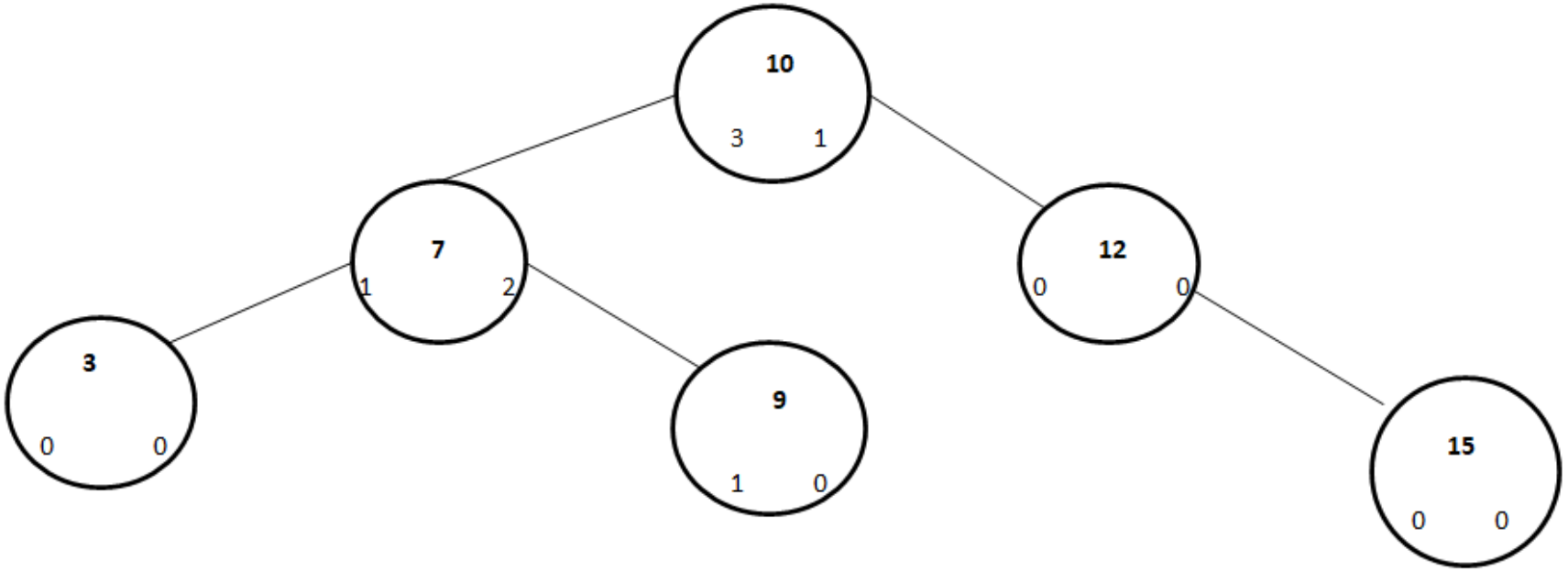
Mas, quando devemos aplicar o método de balanceamento?

Toda vez que for inserido um novo nó!!!

Método inserirAVL()

```
public ARVORE inserirAVL(ARVORE p, int info) {
if (p == null) { //nó inserido sempre será nó folha
          p=new ARVORE();
          p.dado = info;
          p.esq = null; p.dir = null;
          p.hDir=0; p.hEsq=0;
    }
else if (p.dado > info){
          p.esq= inserirAVL (p.esq, info);
          if (p.esq.hDir > p.esq.hEsq) //Altura do nó será a maior
              p.hEsq = p.esq.hDir + 1; //altura dos seus filhos
          else
              p.hEsq = p.esq.hEsq + 1;
        p = balanceamento (p);
          }
     else {
          p.dir=inserirAVL(p.dir, info);
          if (p.dir.hDir > p.dir.hEsq)
              p.hDir = p.dir.hDir + 1;
          else
              p.hDir = p.dir.hEsq + 1;
        }
        p = balanceamento (p);
}
return p;
}
```

Exemplo 3: Configuração Final Depois do Balanceamento



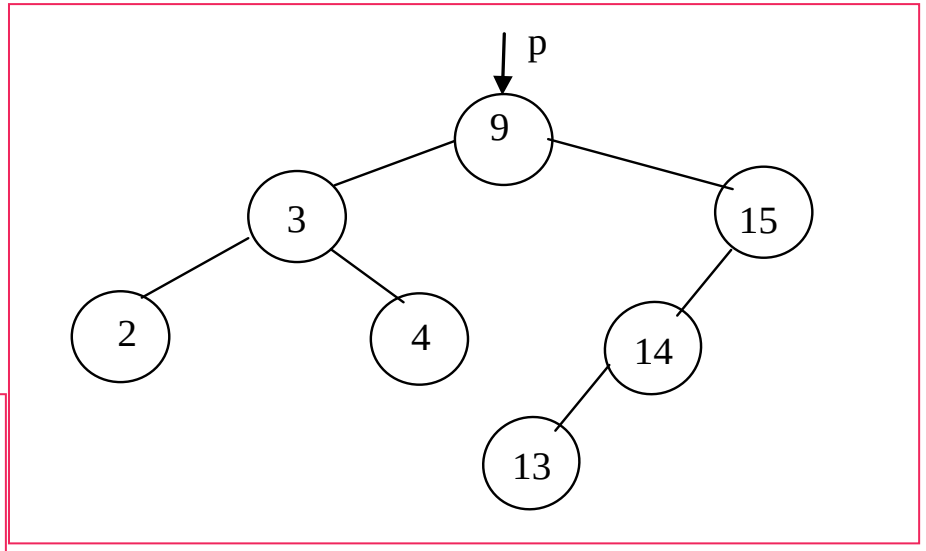
Note que as alturas de cada ramo em cada nó não foram atualizadas

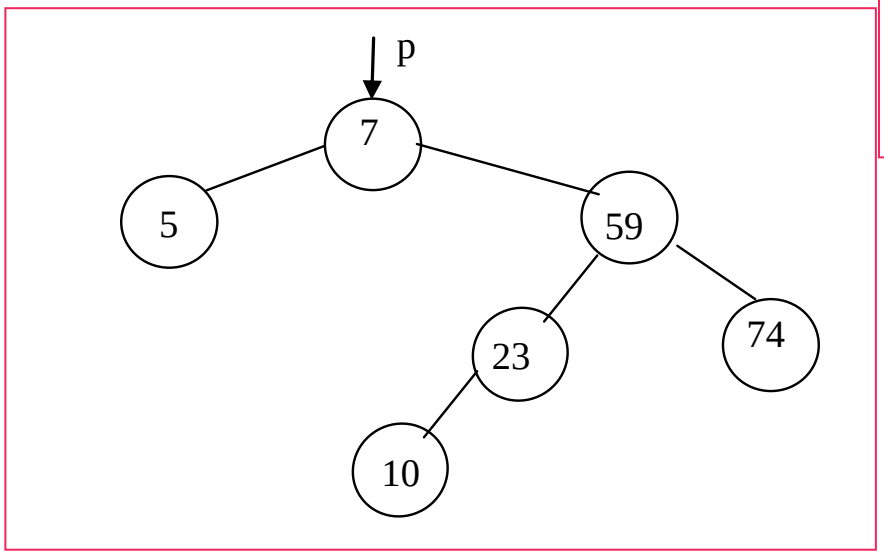
Método atualizaAlturas()

```
public void atualizaAlturas(ARVORE p) {
/*atualiza informação da altura de cada nó depois da remoção percorre a árvore
usando percurso pós-ordem para ajustar primeiro os nós folhas (profundidade
maior) e depois os níveis acima */
  if( p != null) {
        atualizaAlturas(p.esq);
        if (p.esq == null)
                p.hEsq = 0;
        else  if (p.esq.hEsq > p.esq.hDir)
                  p.hEsq = p.esq.hEsq+1;
             else
                  p.hEsq = p.esq.hDir+1;
        atualizaAlturas(p.dir);
        if (p.dir == null)
                p.hDir = 0;
        else if (p.dir.hEsq > p.dir.hDir)
                  p.hDir = p.dir.hEsq+1;
             else
                  p.hDir = p.dir.hDir+1;
        }
}
```

Exercícios Conceitual

2. Aplique os critérios de balanceamento de uma árvore AVL conforme descrito neste texto para as seguintes árvores:





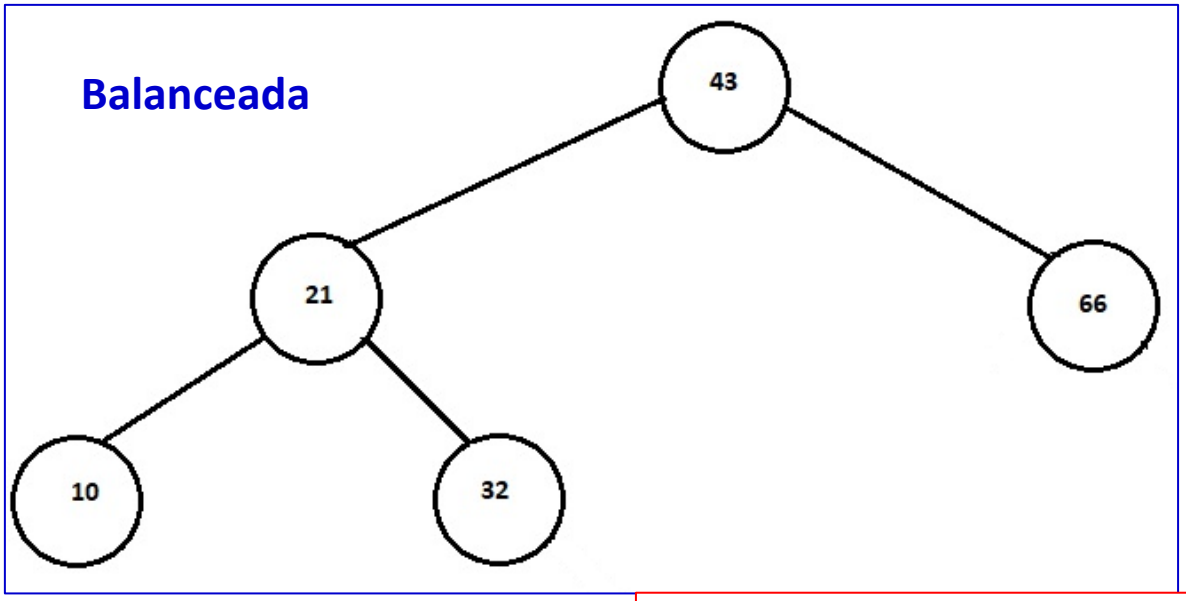
Remoção de um Nó em uma Árvore AVL

- Até agora sempre “causamos” o desbalanceamento da árvore binária pela inserção de um novo nó.
- Quando removemos um determinado nó podemos ou não deixá-la desbalanceada.
- Da mesma forma que na inserção, é o FB de cada nó que determina se será necessário efetuar rotações para devolver a árvore no estado balanceada.
- Porém, há uma importante diferença na identificação do balanceamento, pois na inserção o novo nó sempre será inserido na extremidade, ou seja, como nó folha. Já na remoção de um nó, podemos retirar esse nó em um nível intermediário da sub-árvore.
- Por causa dessa possibilidade, a forma de atualizar as informações de alturas hDir e hEsq de cada nó deve ser feita na árvore como toda e, depois realizar a operação de balanceamento.
- O método **`atualizaAlturaBalanceamento()`** precisa realizar além da atualização de hDir e hEsq de cada nó efetua o balanceamento.

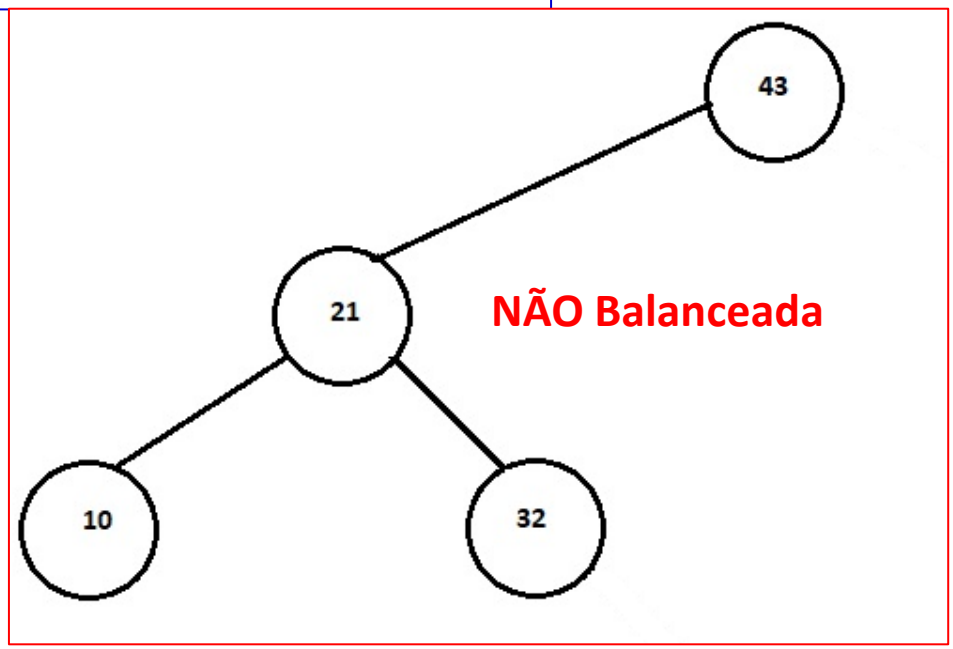
Método atualizaAlturaBalanceamento()

```
public ARVORE atualizaAlturaBalanceamento (ARVORE p) {
/*atualiza informação da altura de cada nó depois da remoção percorre a árvore
usando percurso pós-ordem para ajustar primeiro os nós folhas (profundidade
maior) e depois os níveis acima */
        if( p != null) {
                p.esq = atualizaAlturaBalanceamento (p.esq);
                if (p.esq == null)
                        p.hEsq = 0;
                else  if (p.esq.hEsq > p.esq.hDir)
                          p.hEsq = p.esq.hEsq+1;
                      else
                          p.hEsq = p.esq.hDir+1;
                p.dir = atualizaAlturaBalanceamento (p.dir);
                if (p.dir == null)
                        p.hDir = 0;
                else if (p.dir.hEsq > p.dir.hDir)
                          p.hDir = p.dir.hEsq+1;
                     else
                        p.hDir = p.dir.hDir+1;
                p = balanceamento(p);
        }
        return p;
}
```

Remoção de um Nó em uma Árvore AVL



Remoção do Nó com valor **66**



Exercícios

3. Construa um programa que crie uma AVL para armazenar valores inteiros e que possua um menu com as seguintes opções:
   - 0 - Sair do programa
   - 1 - Insere 1 valor na AVL;
   - 2 - Apresenta pós ordem os nós da árvore apresentando também o FB do nó;
   - 3- Remove um nó escolhido por seu conteúdo
   - 4- Apresenta a altura da árvore
   - 5- Apresenta o número de nós presentes na árvore
4. Elabore um programa que crie uma AVL para armazenar produtos de um supermercado. Cada produto possui os seguintes atributos: código (inteiro), descrição, preço. O programa deve possuir um menu que permita:
   - 0 - Sair do programa
   - 1 – Inserir 1 novo produto no cadastro da loja;
   - 2 - Apresenta todos os produtos com preço menores do que um valor escolhido;
   - 3- Remove um produto do cadastro da loja escolhido por seu código
   - 4- Consulta pelo código um produto, apresentando a descrição e preço,

Referências Bibliográficas

•ASCÊNCIO, A.F.G; ARAUJO, G.S. **Estruturas de Dados: Algoritmos, Análise de Complexidade e Implementações em JAVA e C/C++. São Paulo, Ed.Pearson Prentice Hall, 2010.**

•TENEMBAUM, A.M et al.; **Estruturas de Dados usando C. Makron Books Ltda, 1995.**

•DEITEL, P; J.; Deitel, H.M., **Java: como programar - 8ª edição, São Paulo, Ed.Pearson Prentice Hall, 2010.**