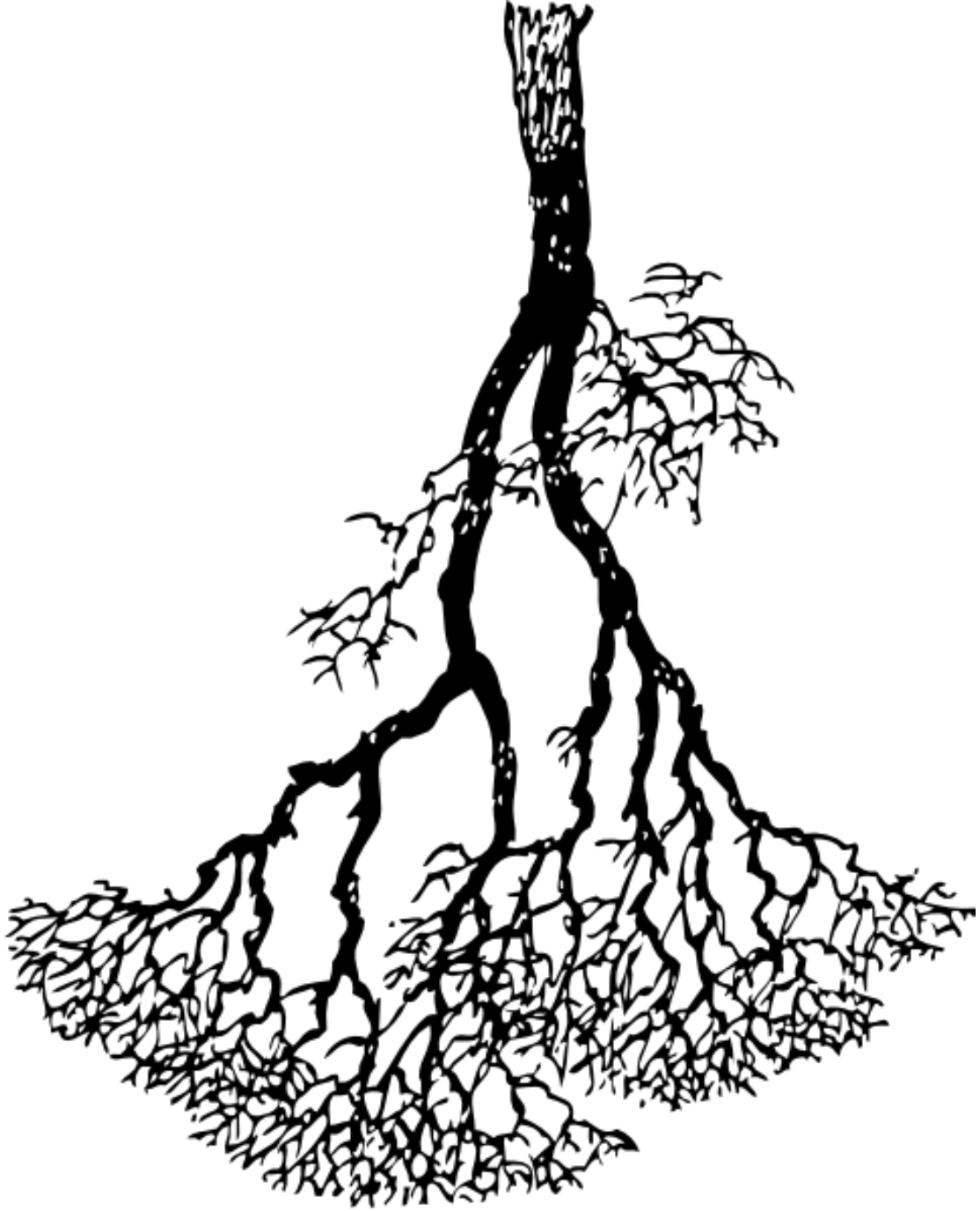
ALGORITMOS E ESTRUTURA DE DADOS

Curso de Engenharia de Software
Lucas Sampaio Leite

Árvores



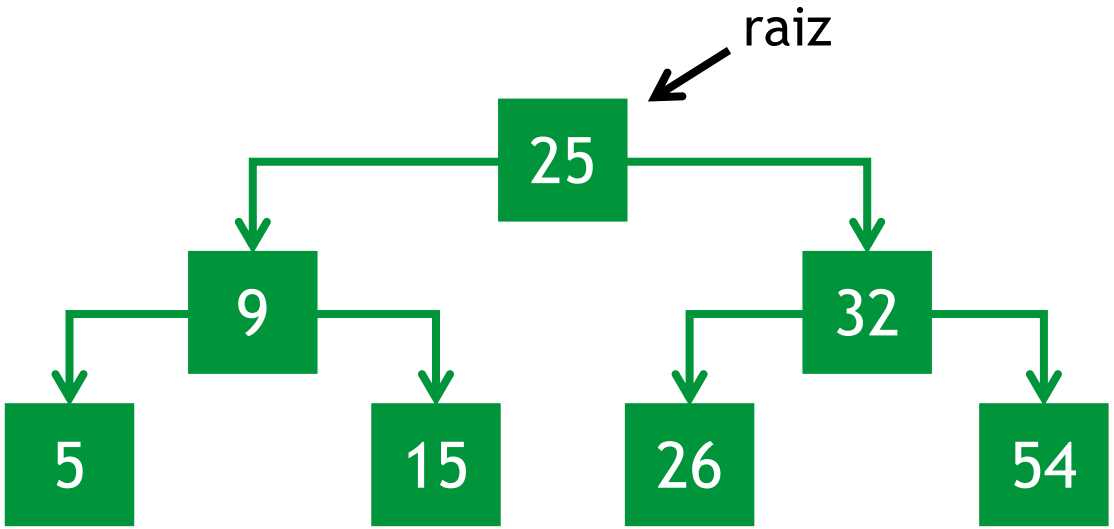
Fonte imagem: https://www.ime.usp.br/~pf/algoritmos/aulas/bint.html

INSTITUTO FEDERAL
Pernambuco

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

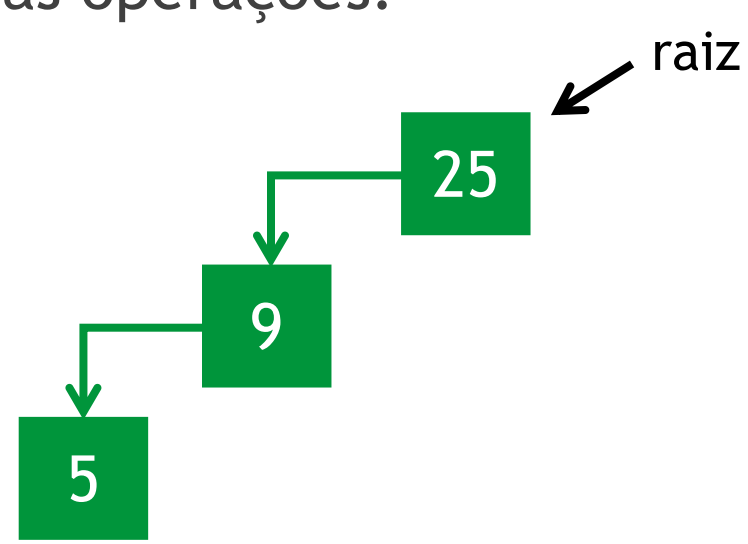
- Uma Árvore Binária de Busca (BST) organiza os dados de forma hierárquica, permitindo operações eficientes de busca, inserção e remoção.
  - Valores menores que o nó são armazenados à esquerda.
  - Valores maiores que o nó são armazenados à direita.

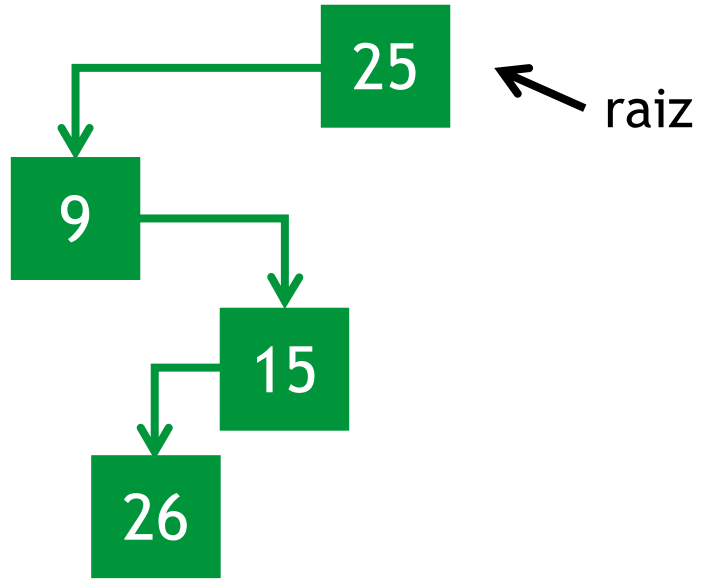


Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- O desempenho das operações em uma BST está diretamente relacionado à altura da árvore. Em uma BST bem estruturada (balanceada), essa altura tende a ser pequena, permitindo operações eficientes.
- No entanto, no pior caso, a árvore pode se tornar desbalanceada, assumindo uma forma linear equivalente a uma lista encadeada, o que degrada o desempenho das operações.



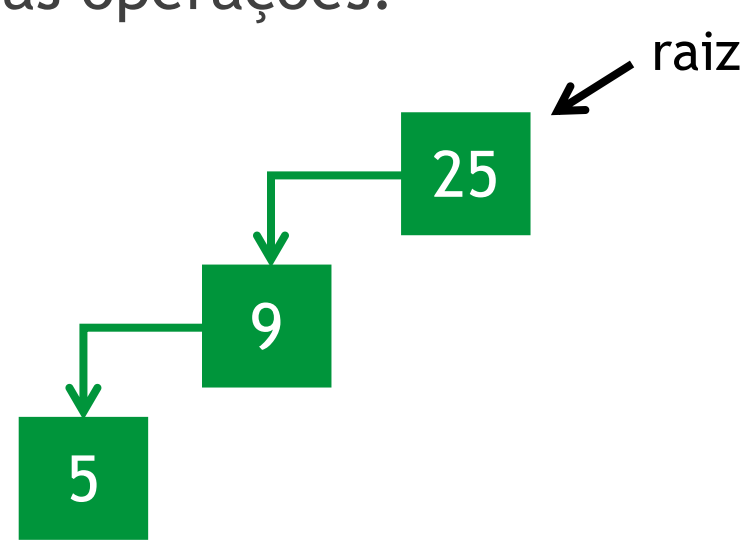


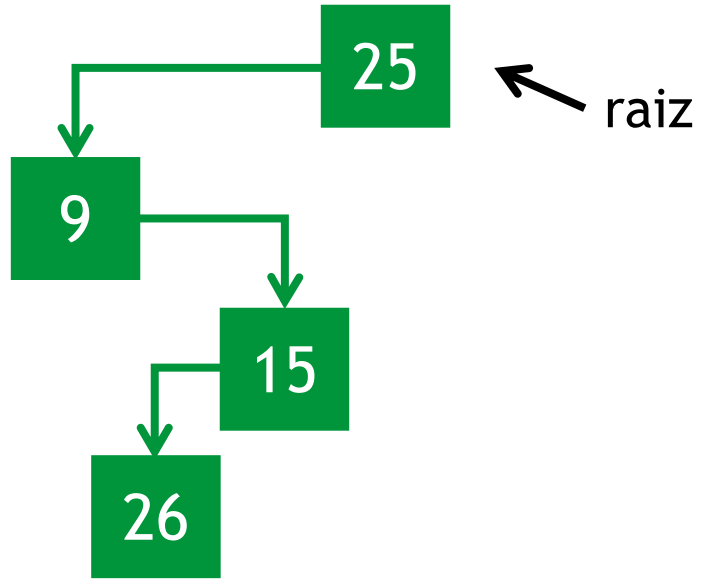
Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- O desempenho das operações em uma BST está diretamente relacionado à altura da árvore. Em uma BST bem estruturada (balanceada), essa altura tende a ser pequena, permitindo operações eficientes.
- No entanto, no pior caso, a árvore pode se tornar desbalanceada, assumindo uma forma linear equivalente a uma lista encadeada, o que degrada o desempenho das operações.

Uma árvore é degenerada se todos os seus nós internos possuem uma única subárvore.





Árvores AVL

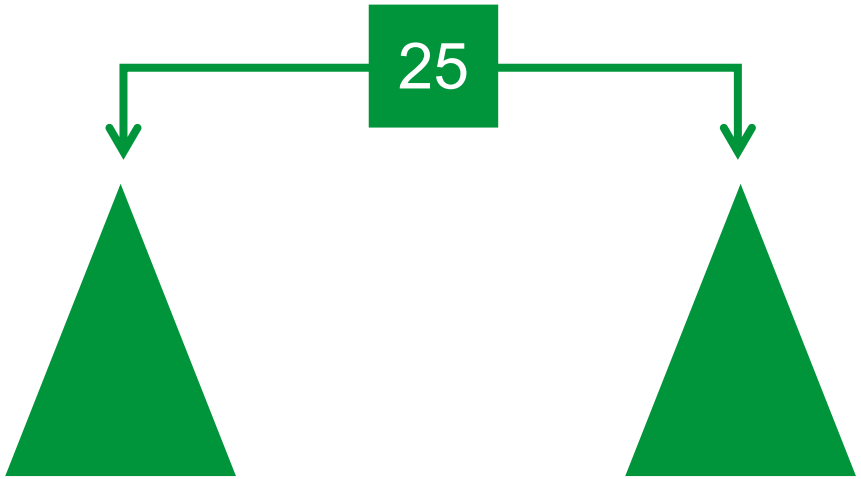
INSTITUTO FEDERAL
Pernambuco

- As árvores balanceadas em altura, também conhecidas como árvores AVL, foram introduzidas em 1962 por **A**delson-**V**elskii e **L**andis.
- Nessa estrutura, o balanceamento é mantido após cada operação, garantindo que a altura da árvore permaneça proporcional a log n.
- Como consequência, as operações de busca, inserção e remoção em uma árvore com n elementos podem ser realizadas em O(log n) mesmo no pior caso.

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Árvore Balanceada: é uma árvore binária em que a diferença de altura entre as subárvores esquerda e direita de qualquer nó é no máximo 1.



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- O fator de balanceamento (FB) é uma medida que indica o quanto um nó está balanceado em uma árvore AVL.
- Ele é calculado pela diferença entre as alturas das subárvores esquerda e direita do nó:

$$\mathrm{FB(n)} = \mathrm{altura(esquerda)} - \mathrm{altura(direita)}$$

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Interpretando o fator de balanceamento:

FB	Situação
0	Subárvores com a mesma altura
1	Subárvore esquerda é 1 nível mais alta
-1	Subárvore direita é 1 nível mais alta
>1	Nó desbalanceado para a esquerda
<-1	Nó desbalanceado para a direita

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Uma árvore AVL pode ser definida recursivamente da seguinte forma:
  - Uma árvore vazia é uma árvore AVL;
  - Seja T uma árvore binária de busca (BST) com subárvores esquerda L e direita R. A árvore T será uma árvore AVL se, e somente se:
    - L e R também forem árvores AVL;
    - a diferença entre as alturas das subárvores for no máximo 1, isto é:

$$|h(L) - h(R)| \leq 1$$

onde h(L) e h(R) representam, respectivamente, as alturas das subárvores esquerda e direita.

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

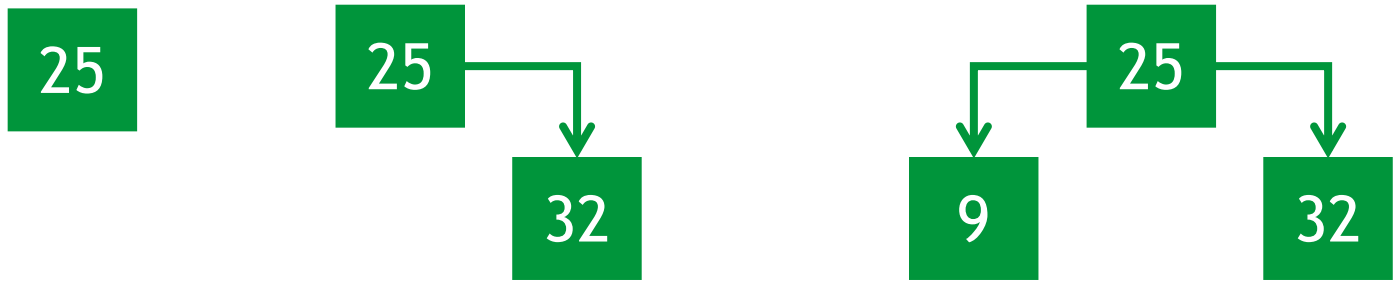
- Exemplos de AVLs:

Árvores AVL

- Exemplos de AVLs:

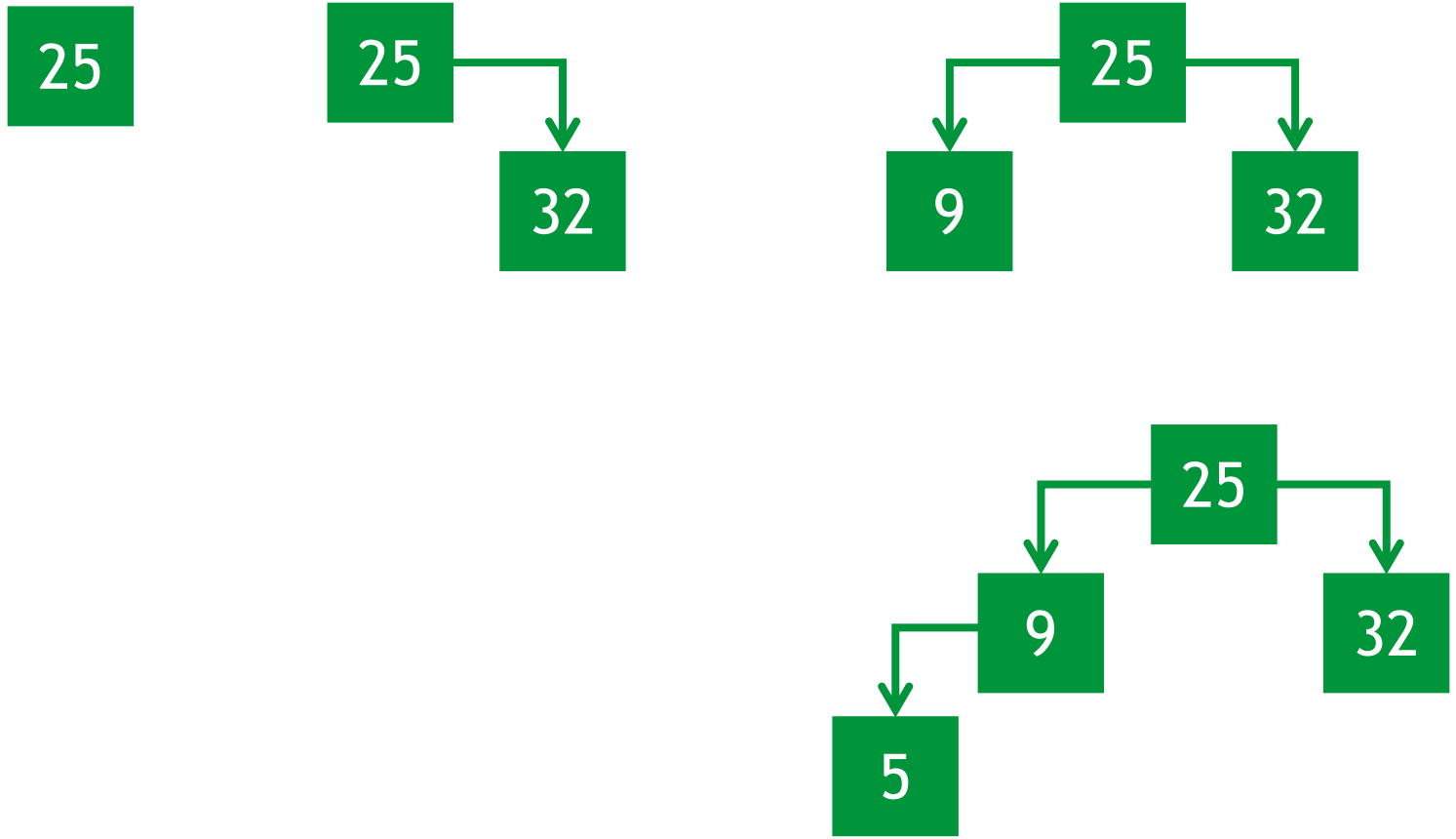
Árvores AVL

- Exemplos de AVLs:



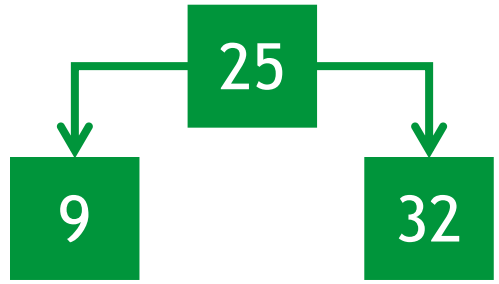
Árvores AVL

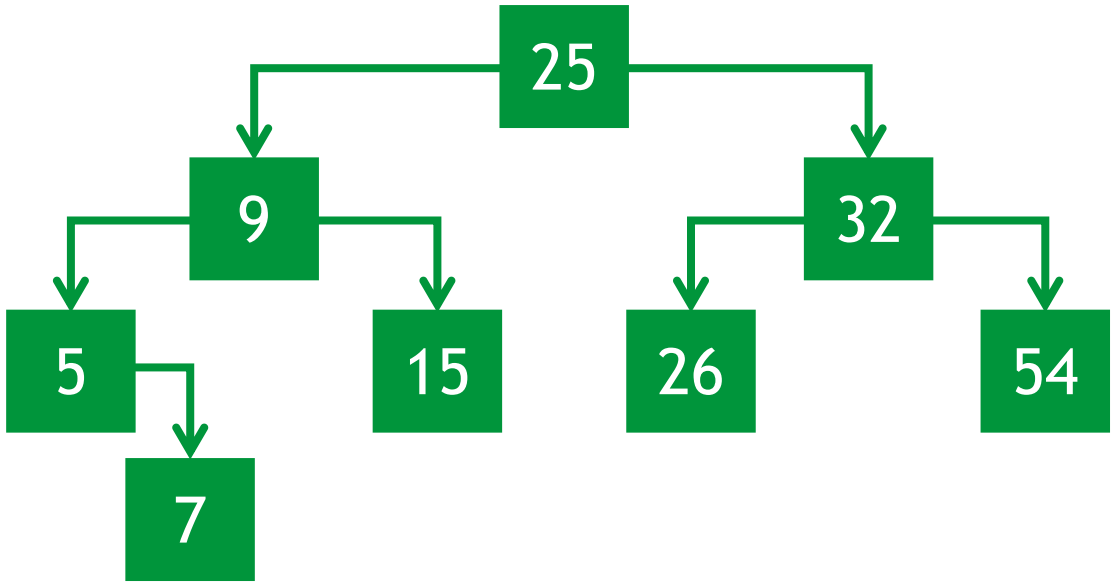
- Exemplos de AVLs:

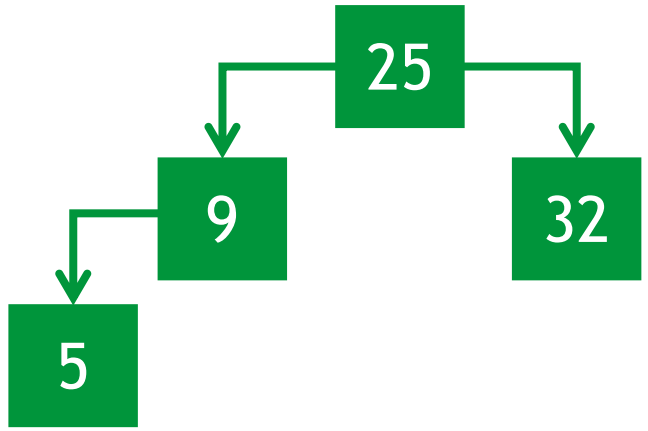


Árvores AVL

- Exemplos de AVLs:



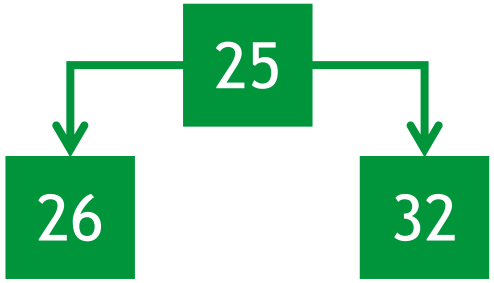




Árvores AVL

INSTITUTO FEDERAL
Pernambuco

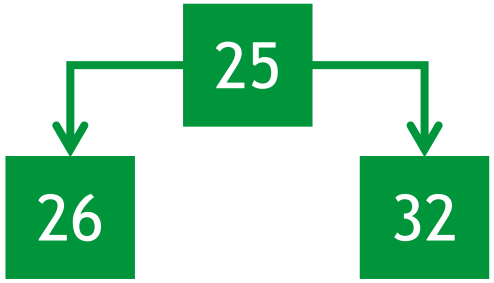
- Exemplos de não-AVLs:

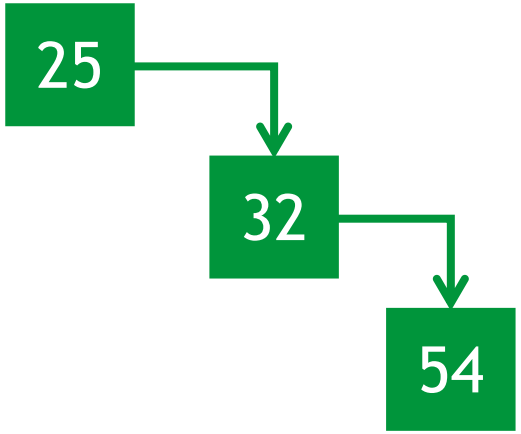


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Exemplos de não-AVLs:

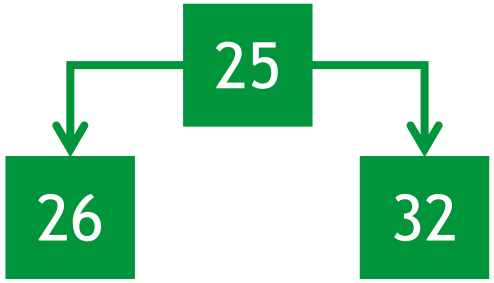


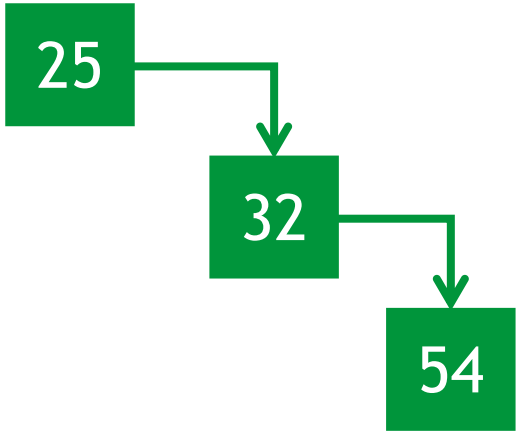


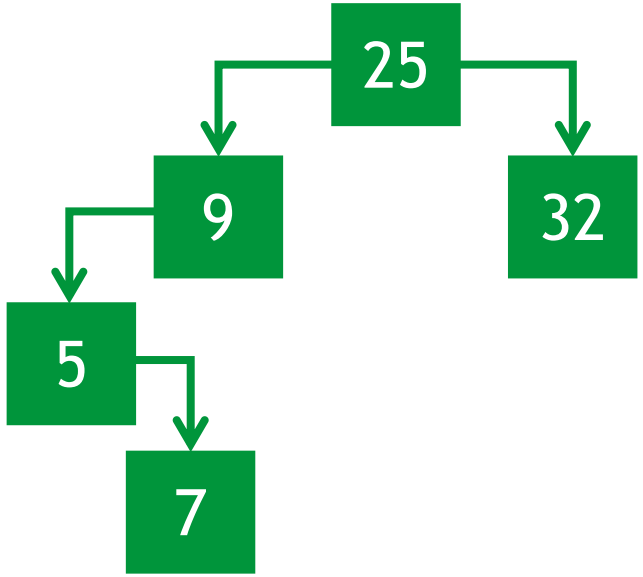
Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Exemplos de não-AVLs:







Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Representando um Nó em C:

```c
typedef struct No {
    int chave;
    int altura;
    struct No *esquerda;
    struct No *direita;
} No;
```

Armazenar a altura de cada nó permite calcular facilmente o fator de balanceamento e verificar se a árvore ainda está balanceada após uma operação.

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Criando um Nó em C:

```c
No *criarNo(int chave) {
    No *novo = (No *) malloc(sizeof(No));

    if (novo == NULL)
        return NULL;

    novo->chave = chave;
    novo->altura = 0;
    novo->esquerda = NULL;
    novo->direita = NULL;

    return novo;
}
```

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Obtendo o fator de balanceamento:

```
int fatorBalanceamento(No *n) {
    if (n == NULL)
        return 0;

    return altura(n->esquerda) - altura(n->direita);
}
```

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Obtendo o fator de balanceamento:

```c
int fatorBalanceamento(No *n) {
    if (n == NULL)
        return 0;

    return altura(n->esquerda) - altura(n->direita);
}
```

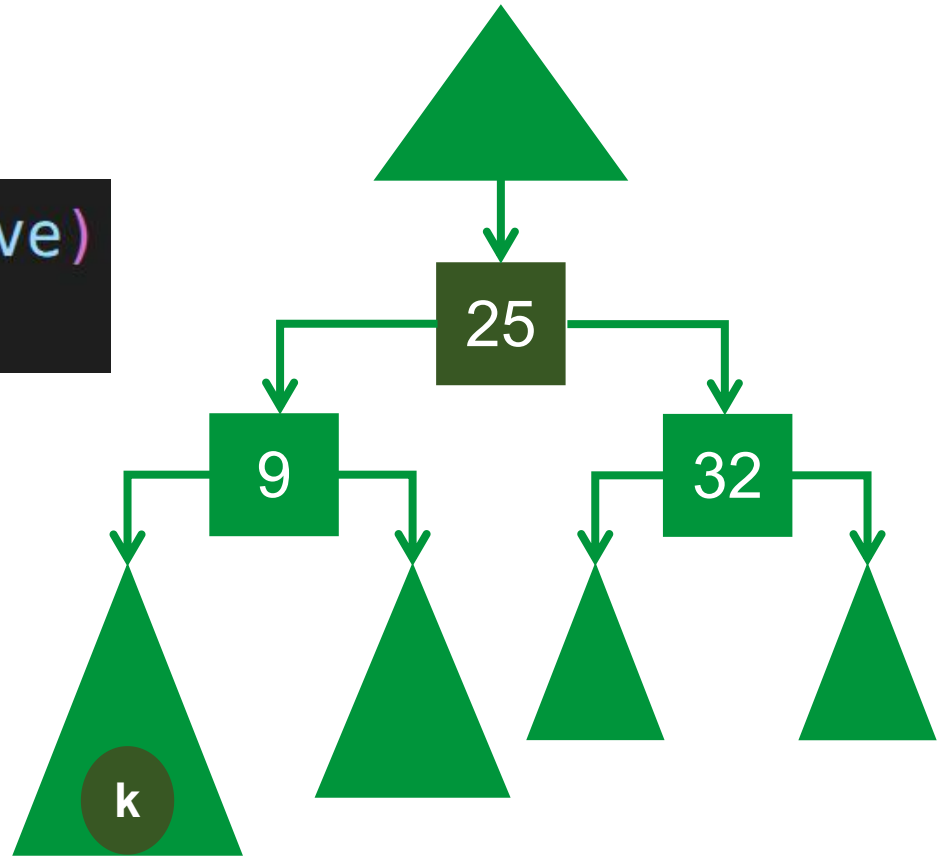
```c
int altura(No *n) {
    if (n == NULL)
        return -1;
    return n->altura;
}
```

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Rotação à direita (R-rotation): ocorre quando a inserção é feita na subárvore esquerda da subárvore esquerda de um nó desbalanceado.
- Esse caso caracteriza o padrão de desbalanceamento esquerda-esquerda (LL).

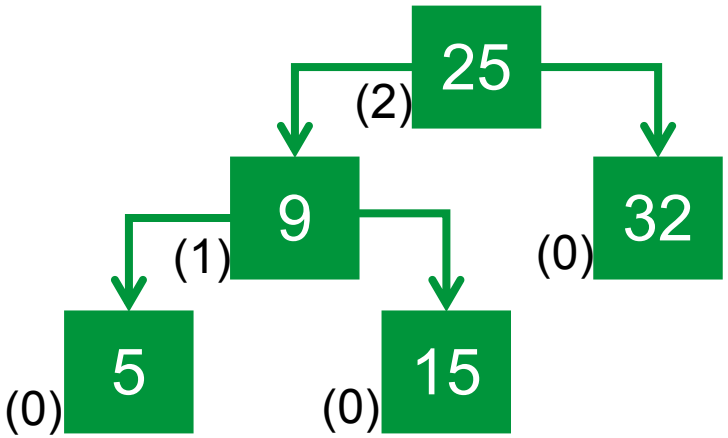
```
if (fb > 1 && chave < raiz->esquerda->chave)
    return rotacaoDireita(raiz);
```



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

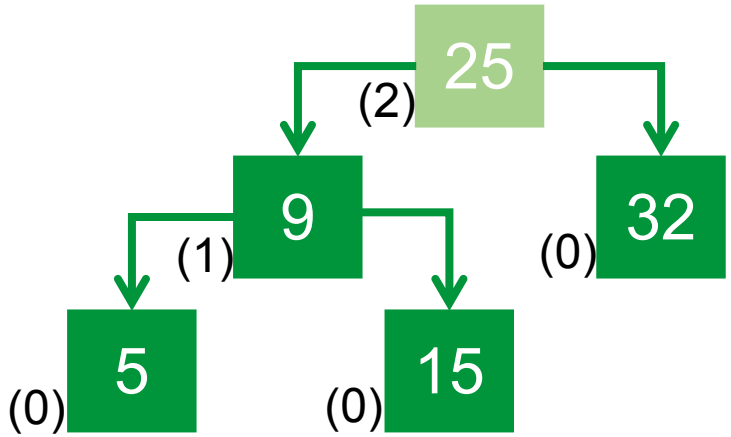
- Inserindo o nó 3:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

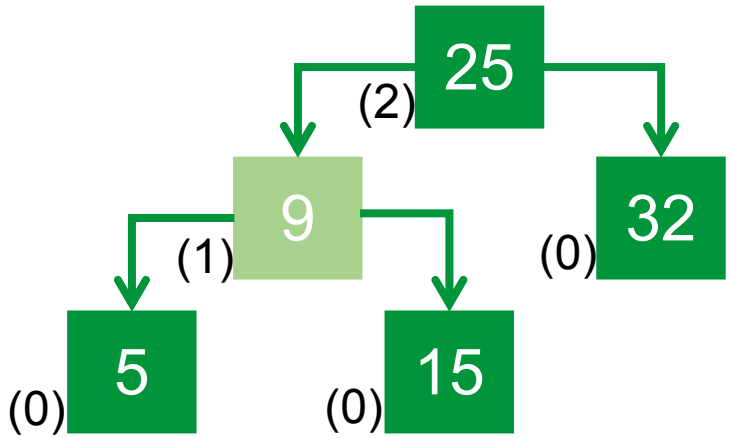
- Inserindo o nó 3:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

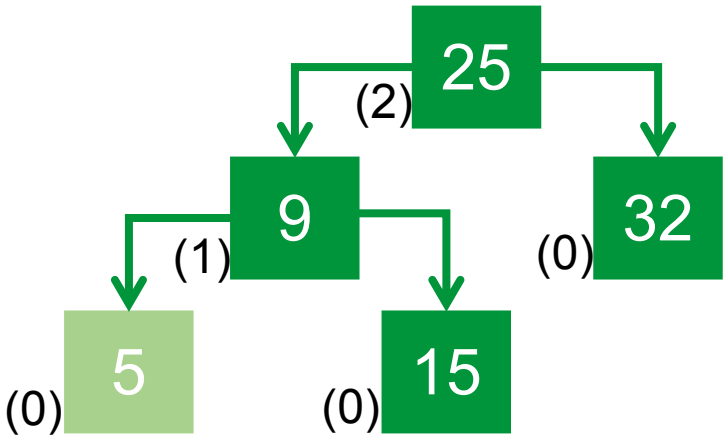
- Inserindo o nó 3:



Árvores AVL

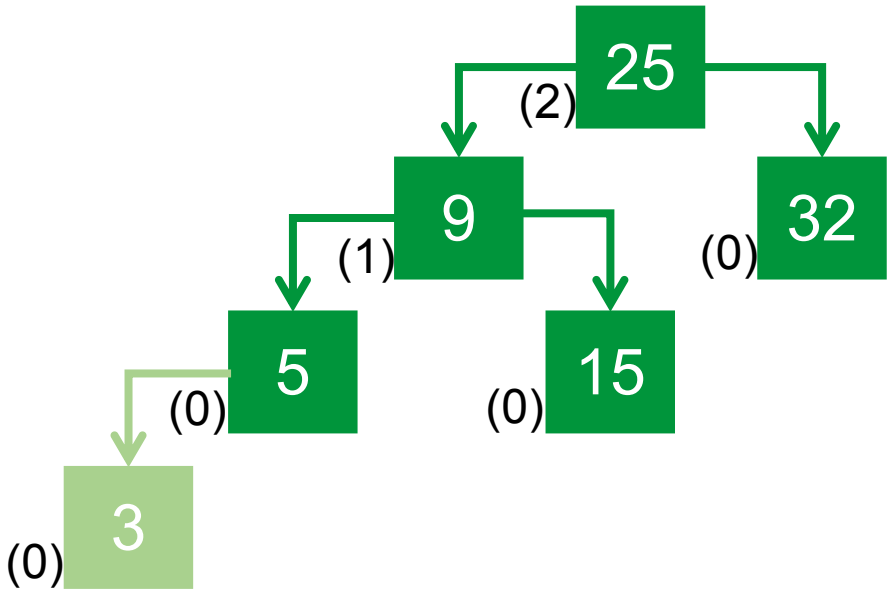
INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 3:



Árvores AVL

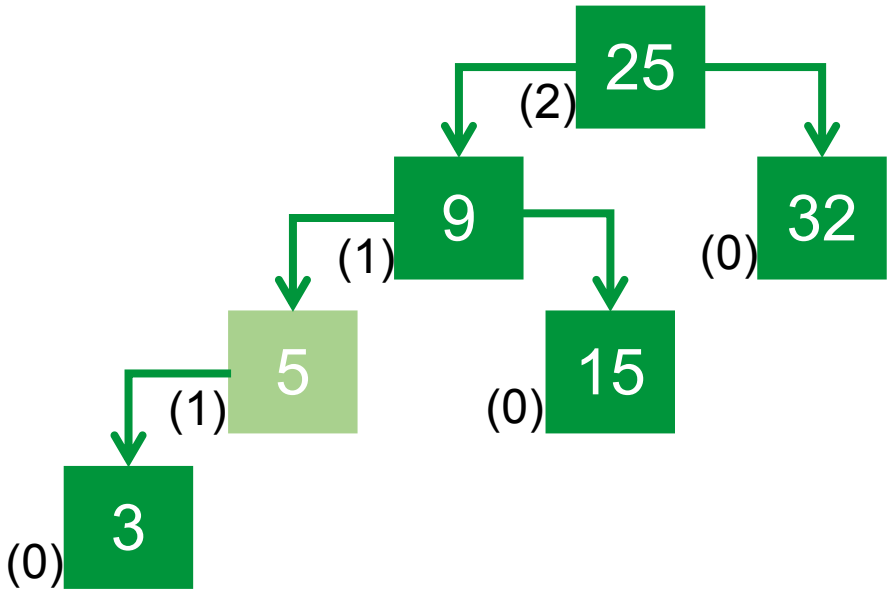
- Inserindo o nó 3:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

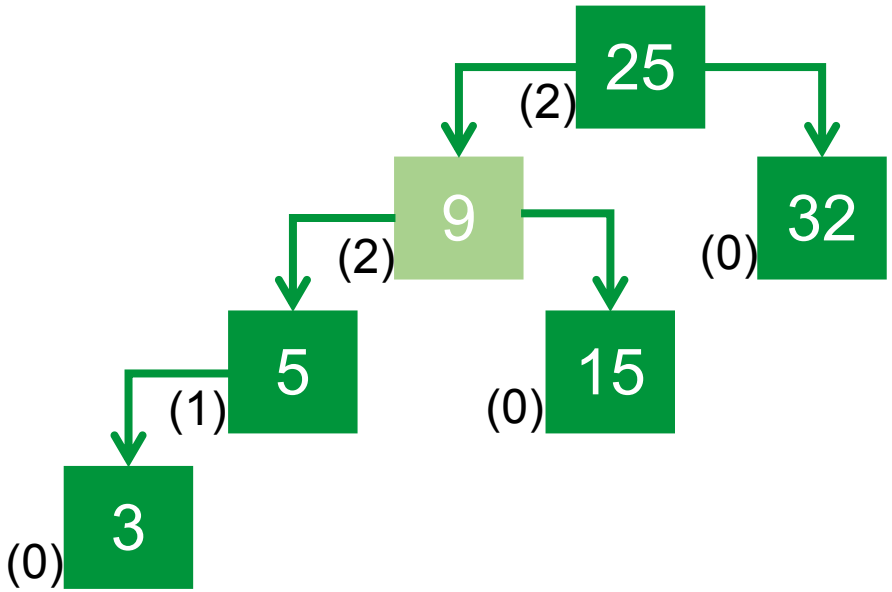
- Inserindo o nó 3:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

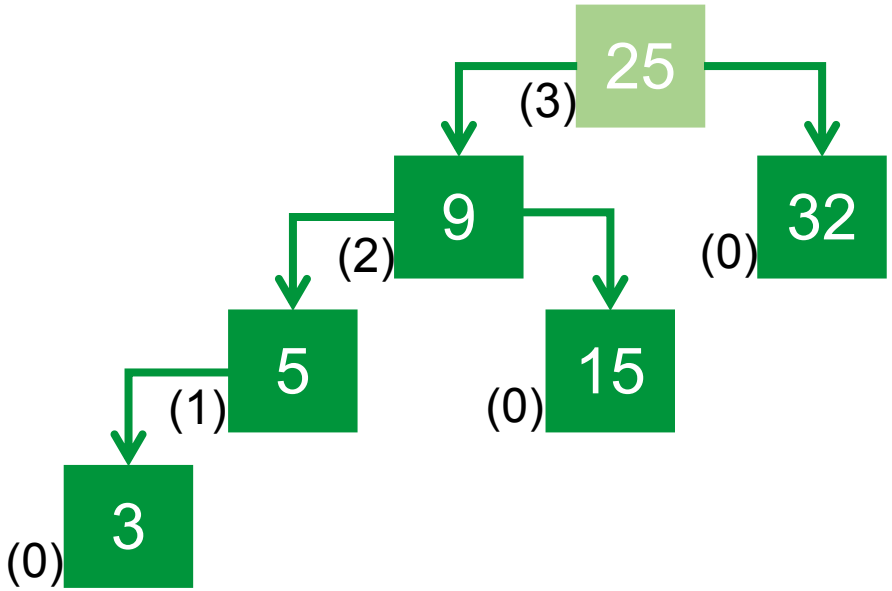
- Inserindo o nó 3:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 3:

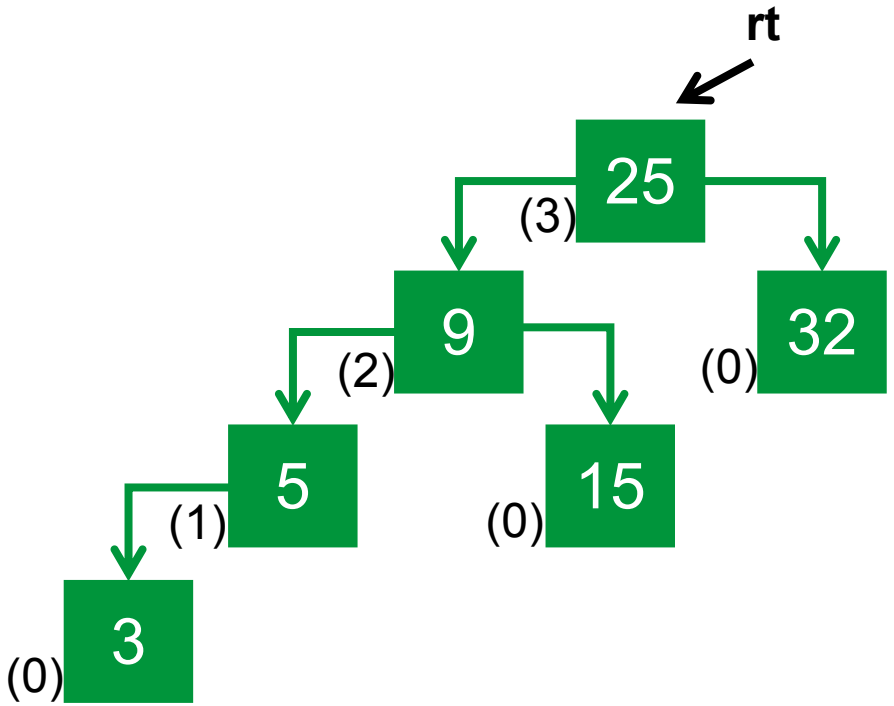



```
if (fb > 1 && chave < raiz->esquerda->chave)
    return rotacaoDireita(raiz);
```

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

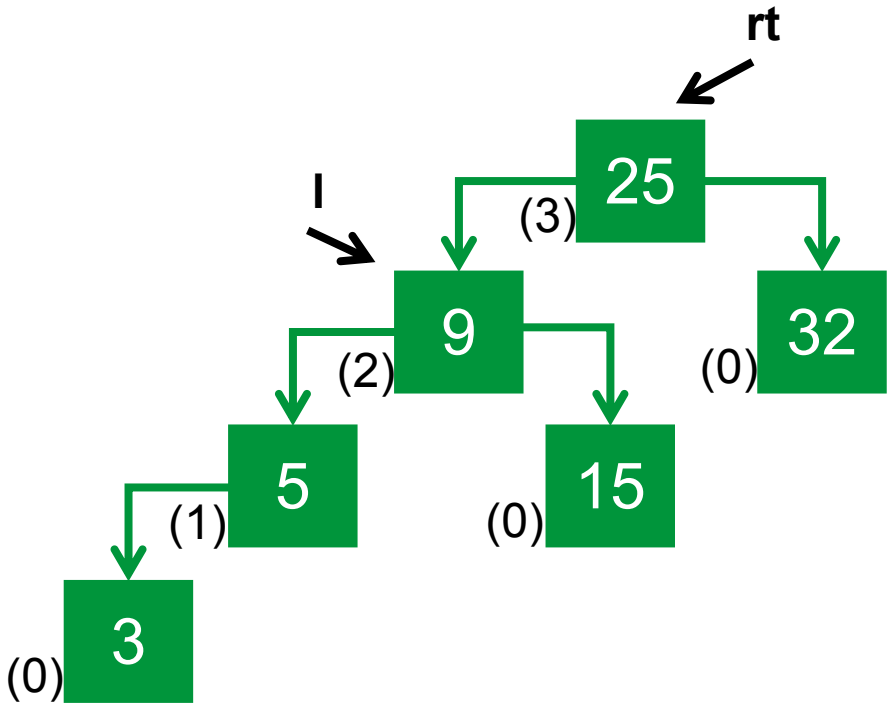
- Inserindo o nó 3:




```c
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

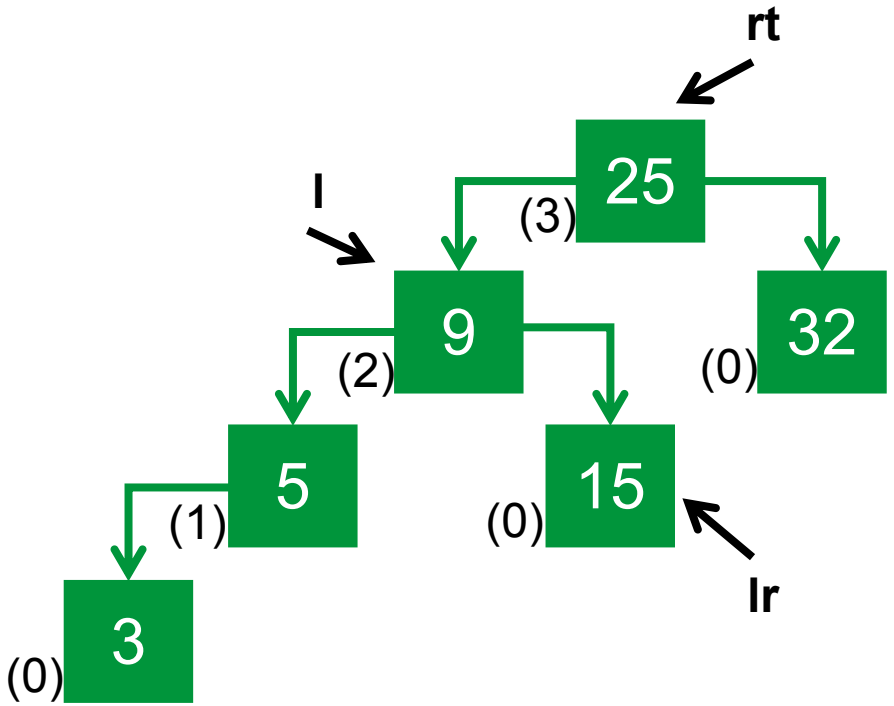
- Inserindo o nó 3:




```c
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

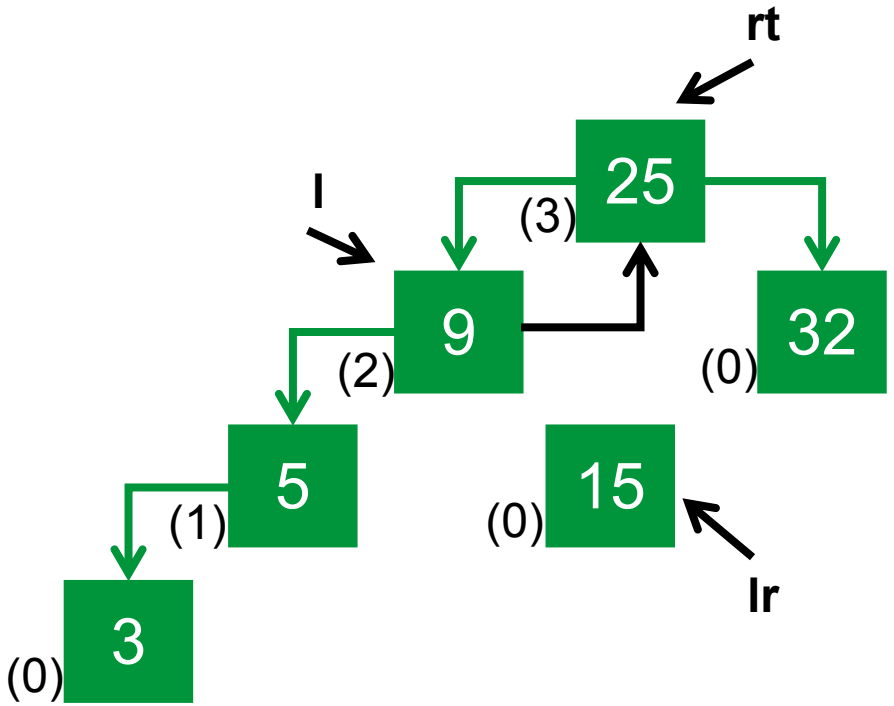
- Inserindo o nó 3:




```c
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

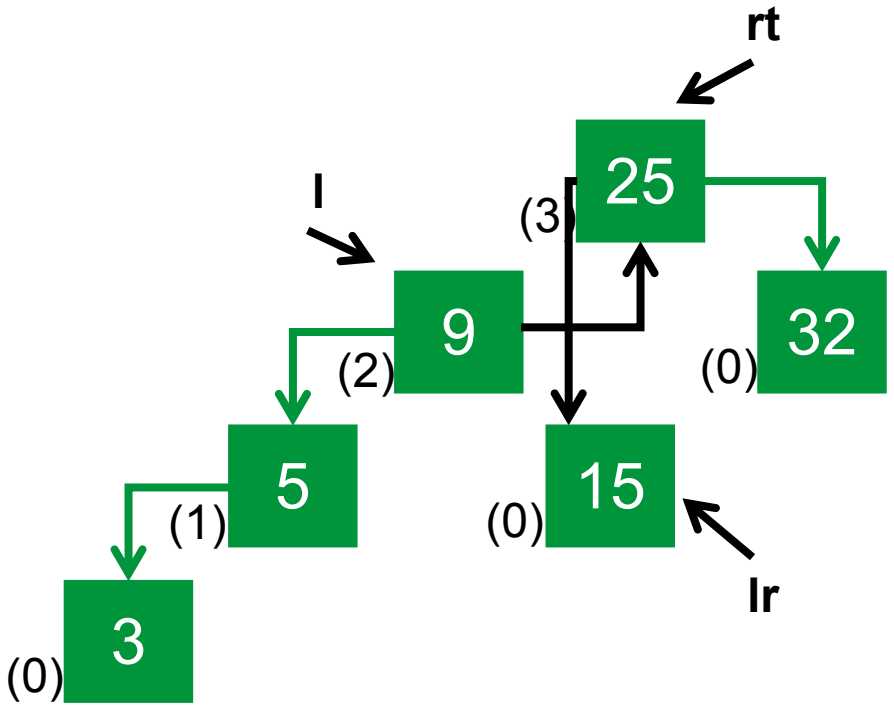
- Inserindo o nó 3:




```c
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

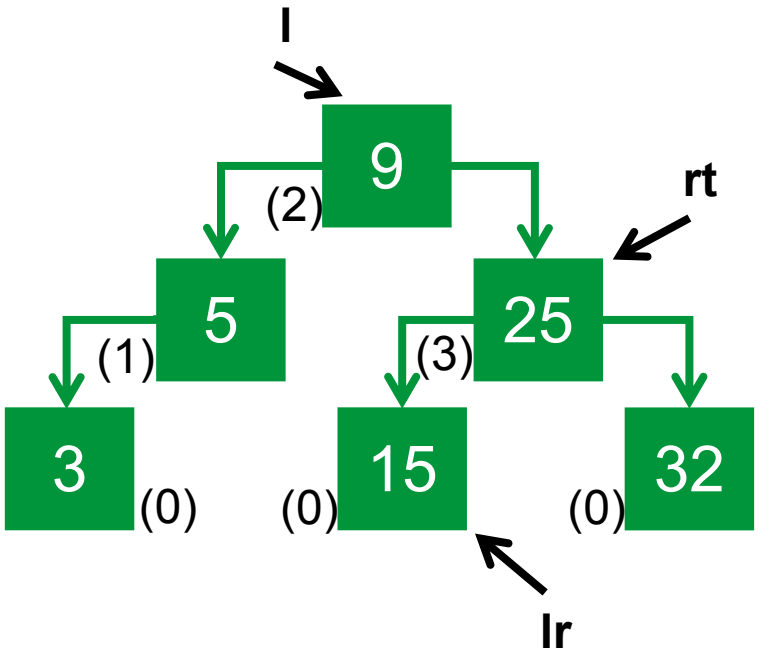
- Inserindo o nó 3:




```c
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

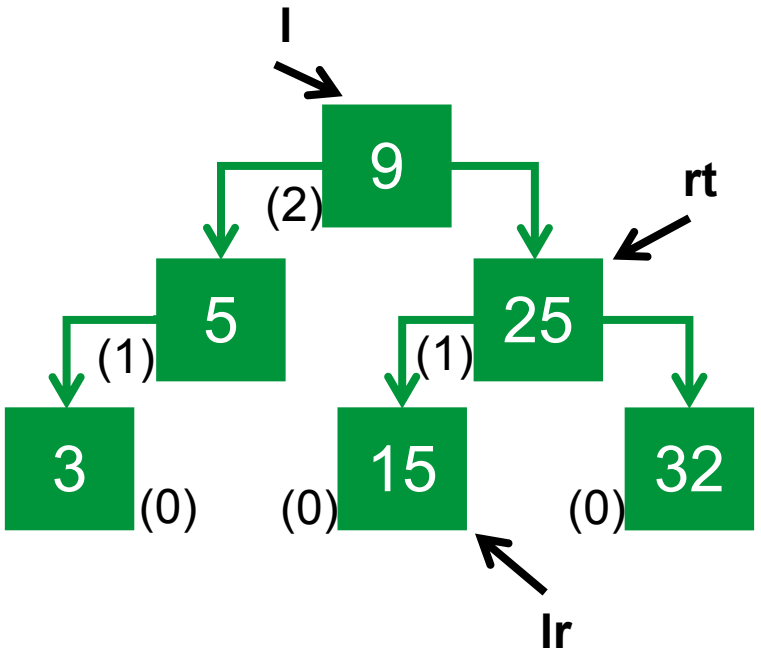
- Inserindo o nó 3:




```c
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

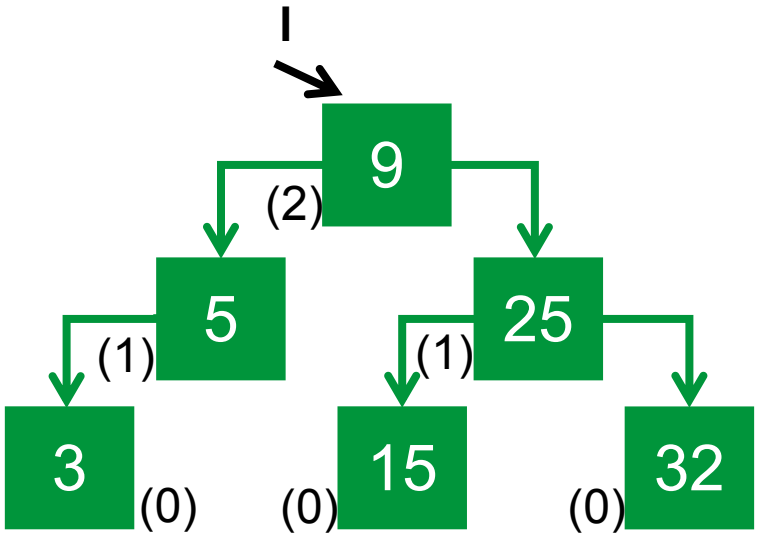
- Inserindo o nó 3:




```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

- Inserindo o nó 3:



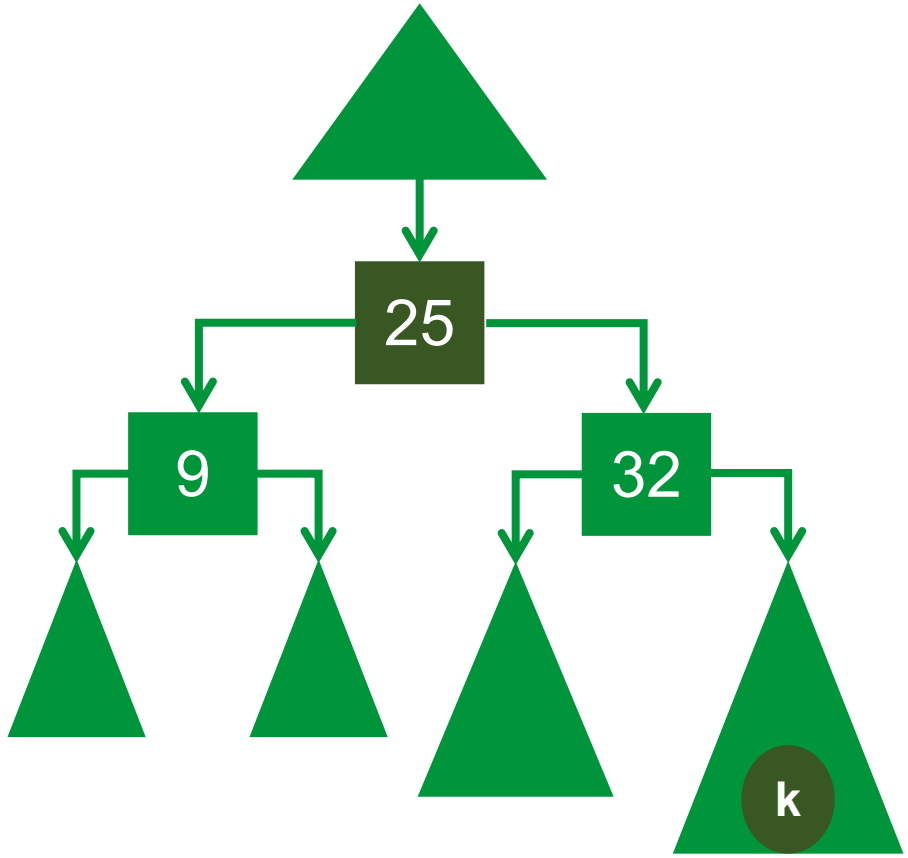

```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Rotação à esquerda (L-rotation): ocorre quando a inserção é feita na subárvore direita da subárvore dureita de um nó desbalanceado.
- Esse caso caracteriza o padrão de desbalanceamento direita-direita (RR).

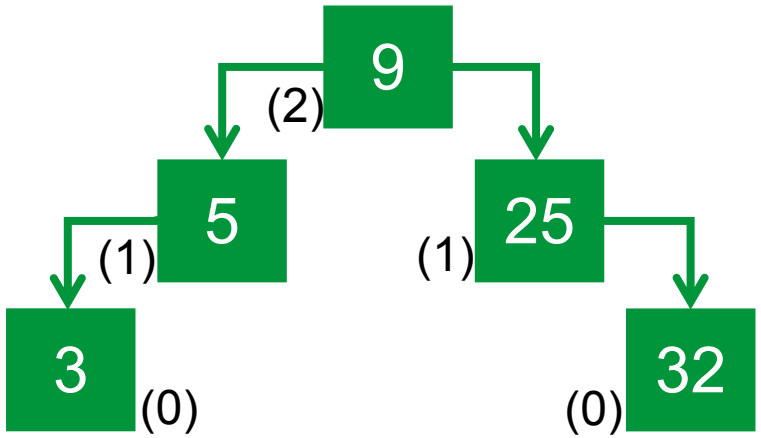
```
if (fb < -1 && chave > raiz->direita->chave)
    return rotacaoEsquerda(raiz);
```



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

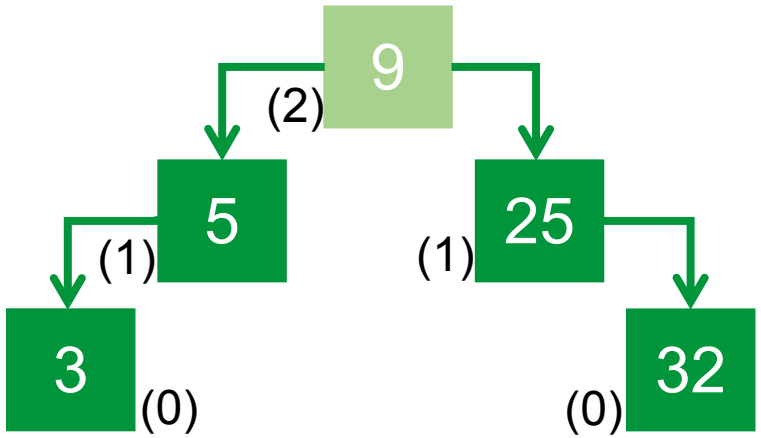
- Inserindo o nó 35:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

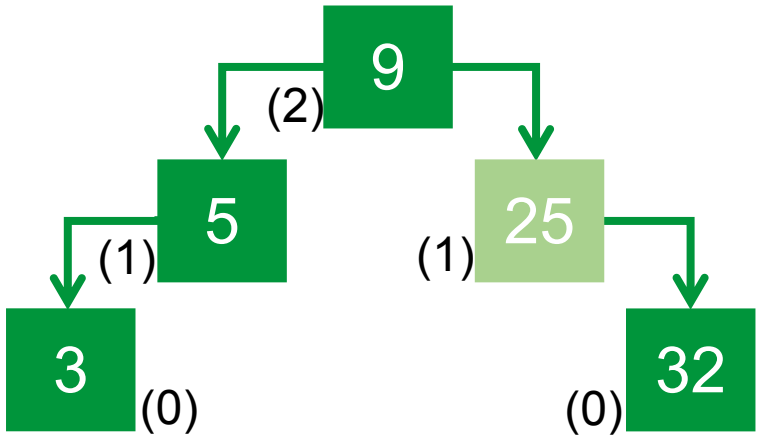
- Inserindo o nó 35:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

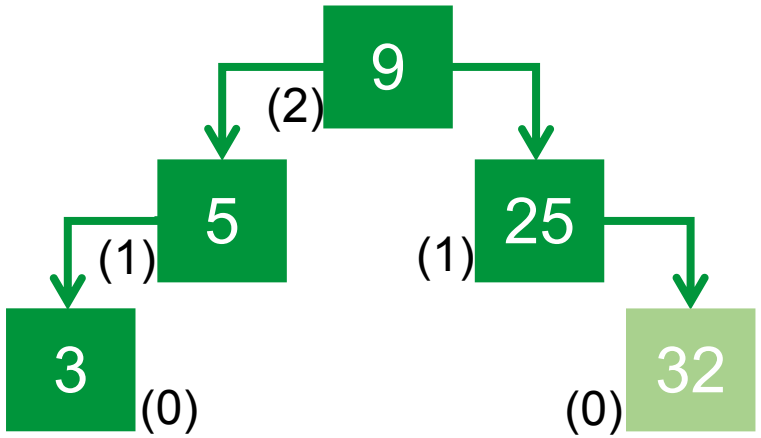
- Inserindo o nó 35:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

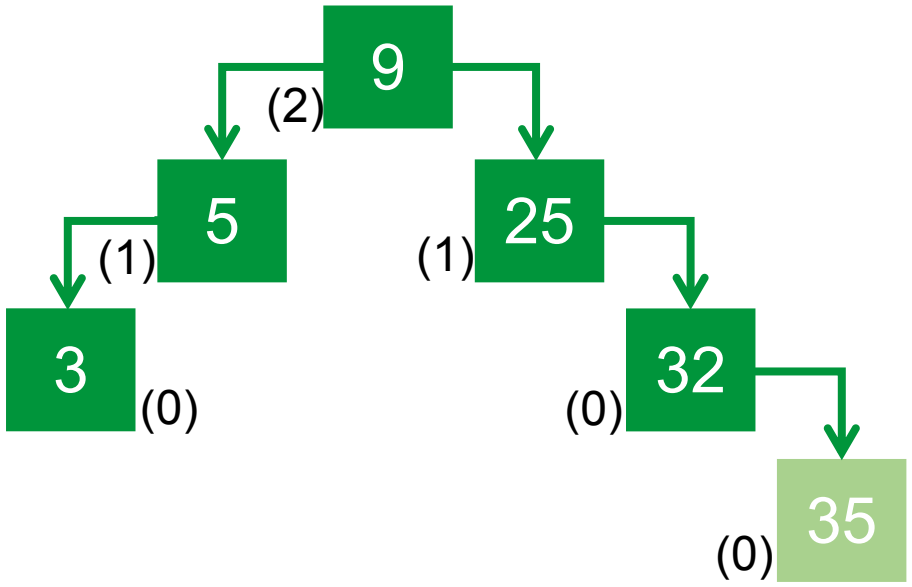
- Inserindo o nó 35:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

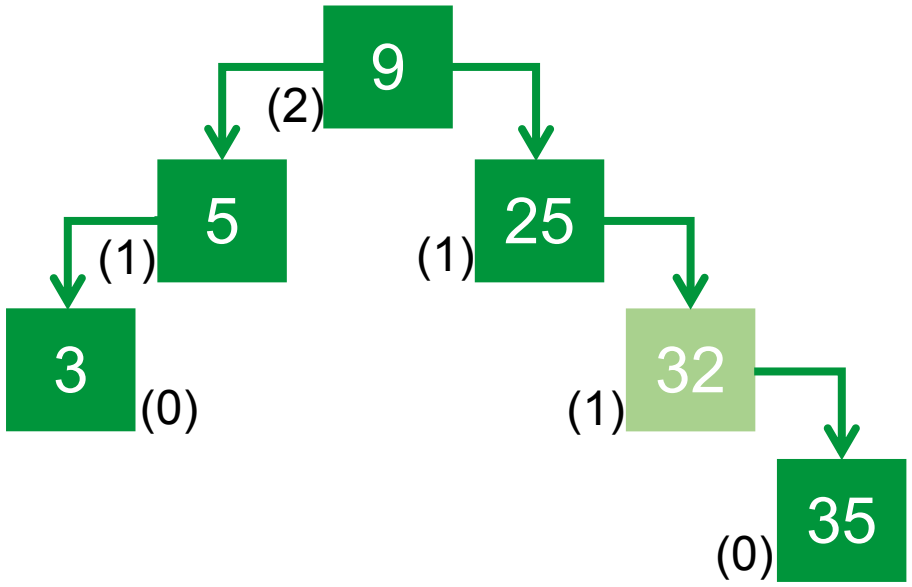
- Inserindo o nó 35:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 35:

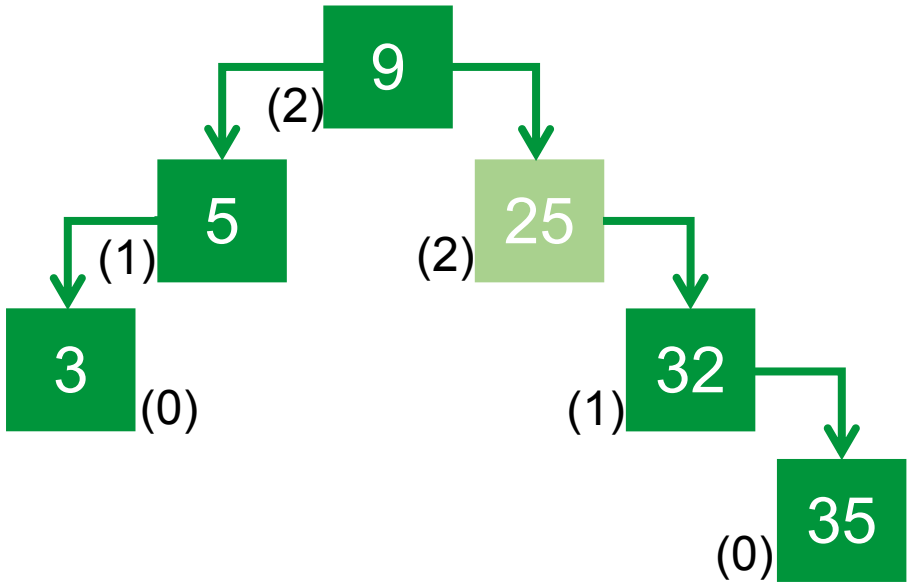


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 35:

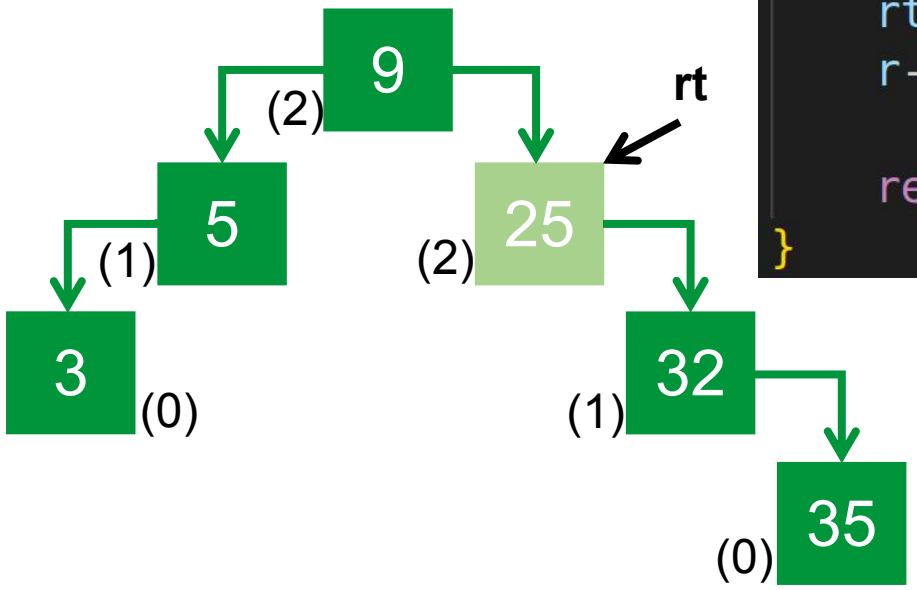
```
if (fb < -1 && chave > raiz->direita->chave)
    return rotacaoEsquerda(raiz);
```



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 35:

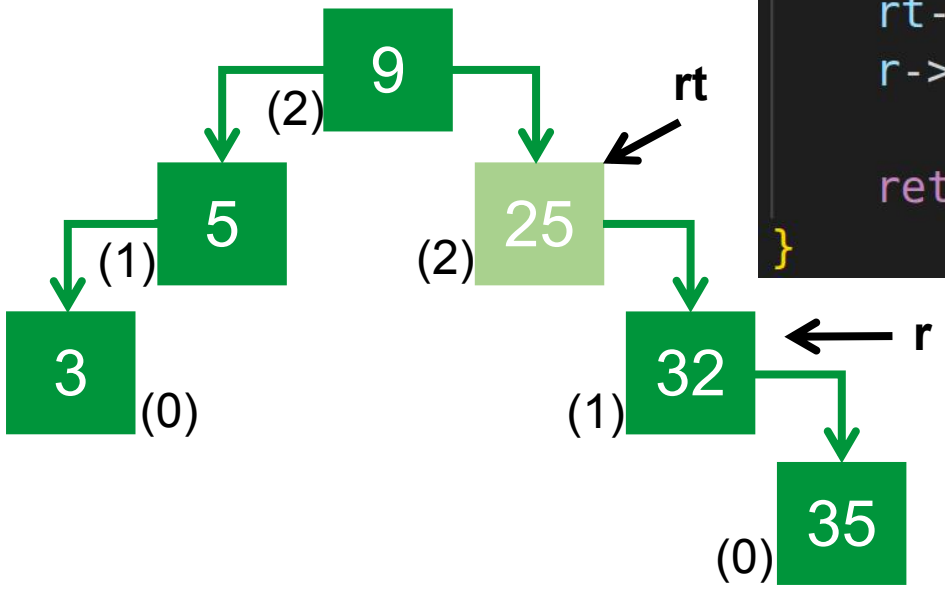



```c
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```

Árvores AVL

- Inserindo o nó 35:

```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```

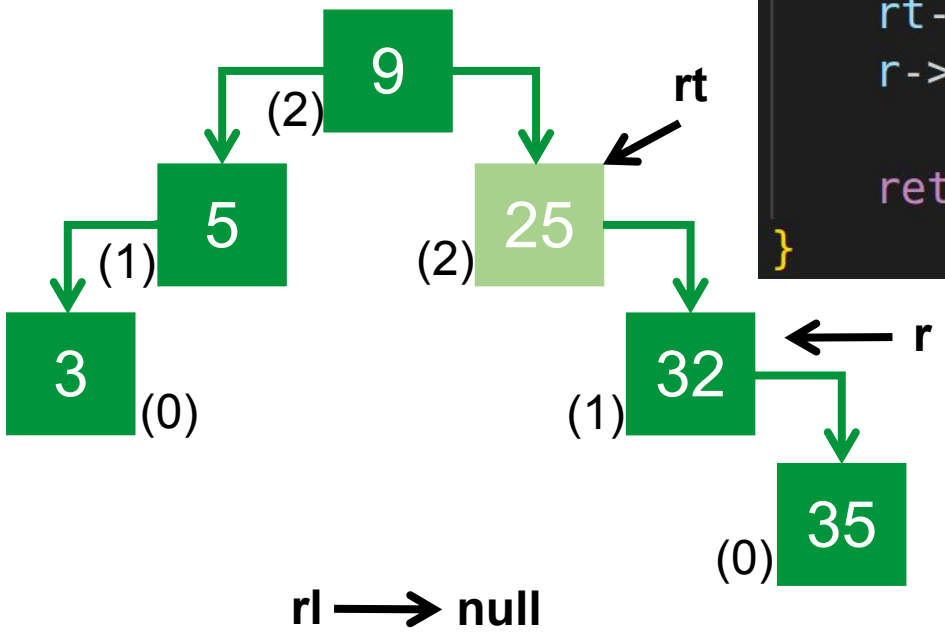


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 35:

```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```

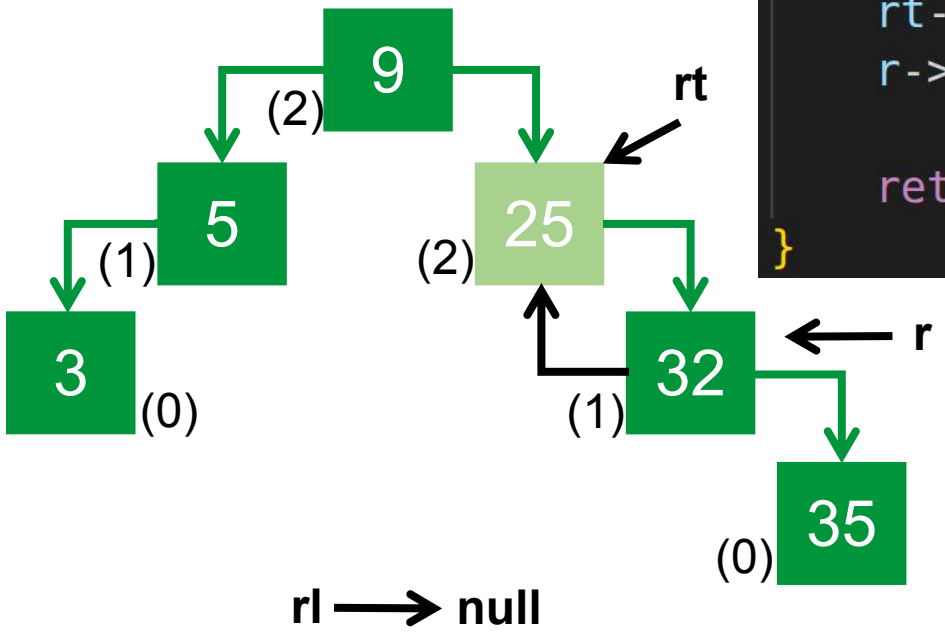


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 35:

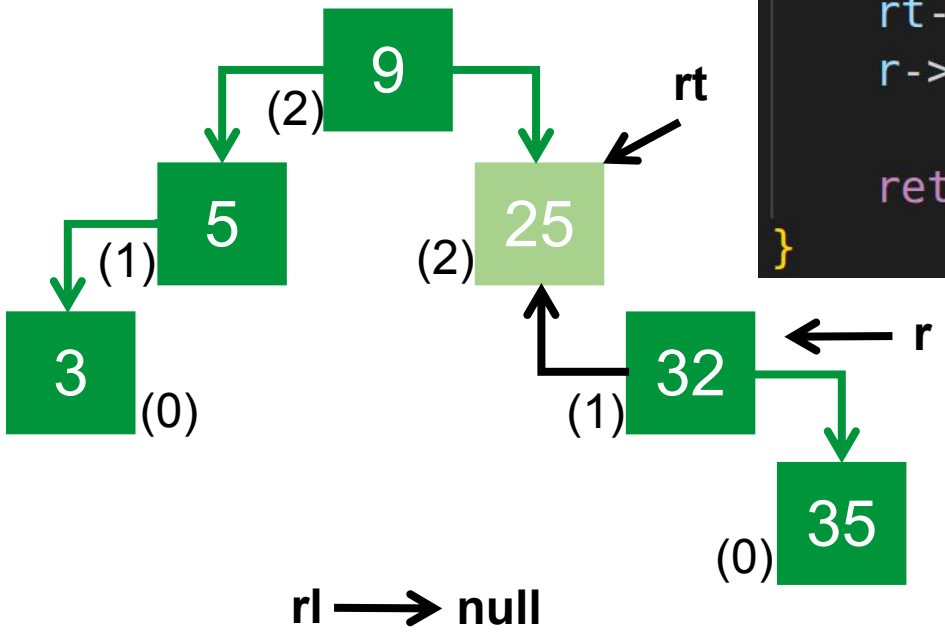
```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 35:

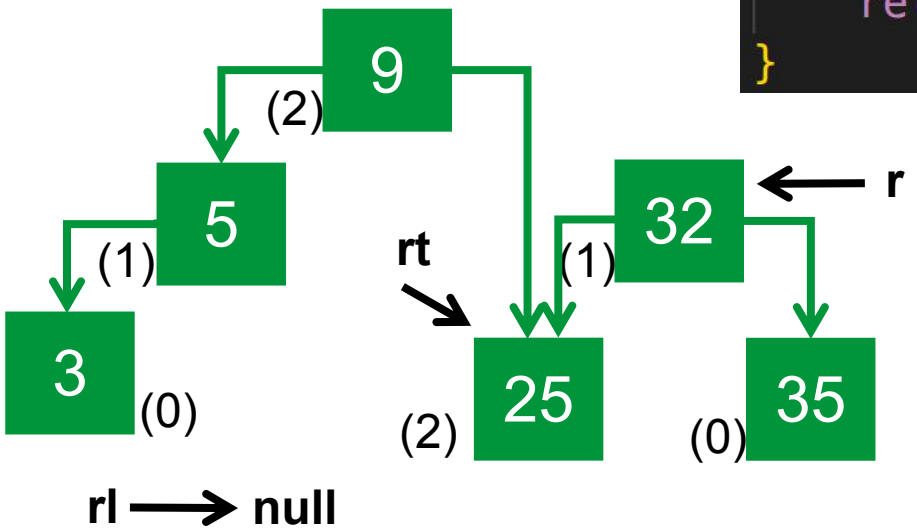
```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 35:

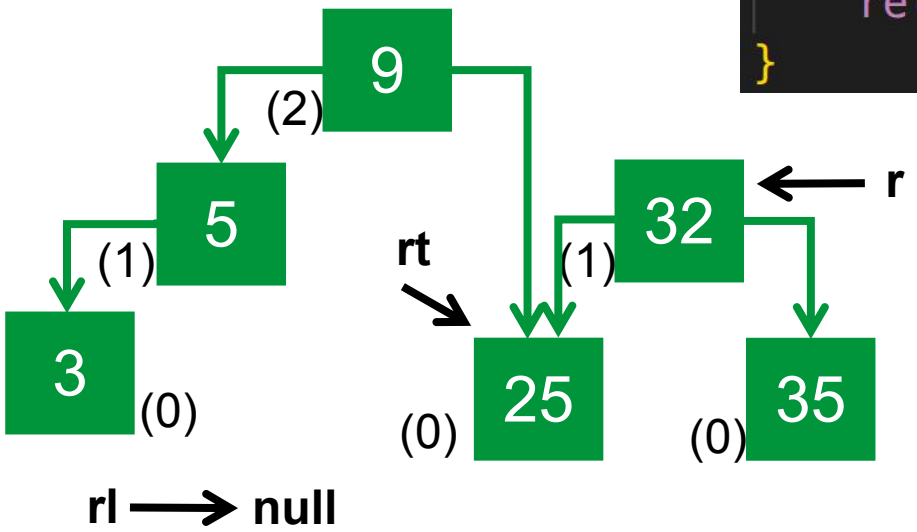
```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 35:

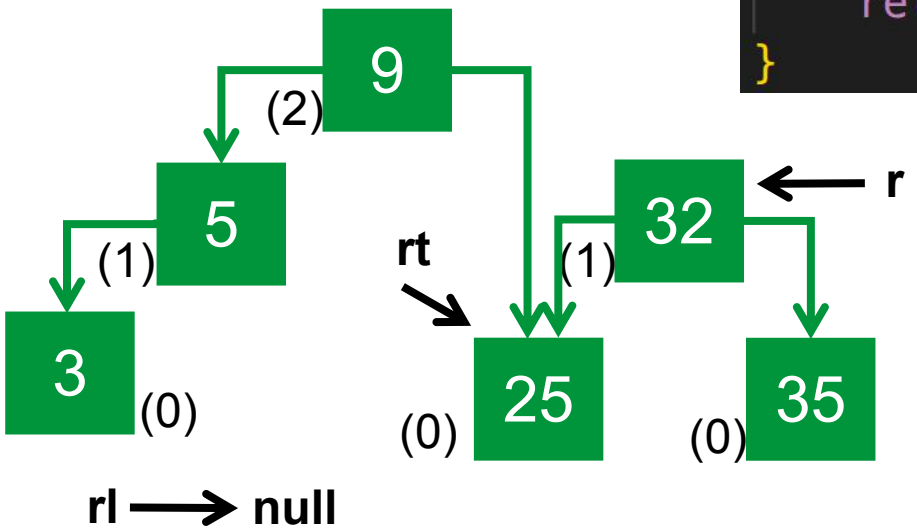
```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 35:

```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```

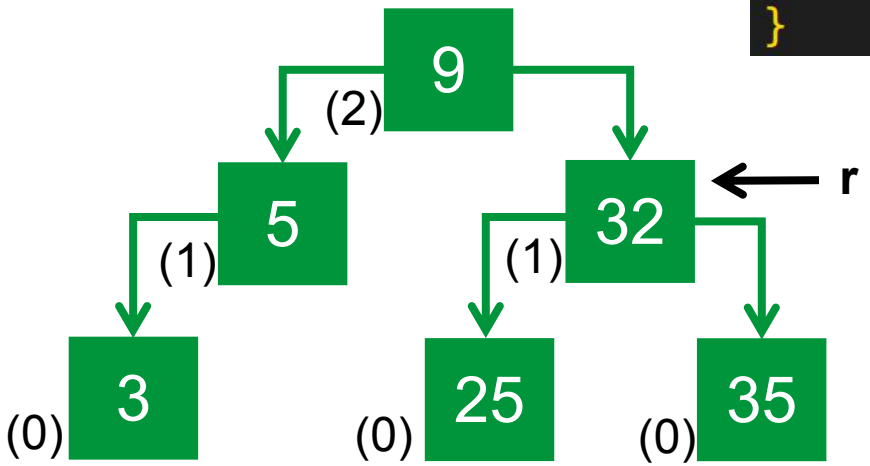


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 35:

```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```

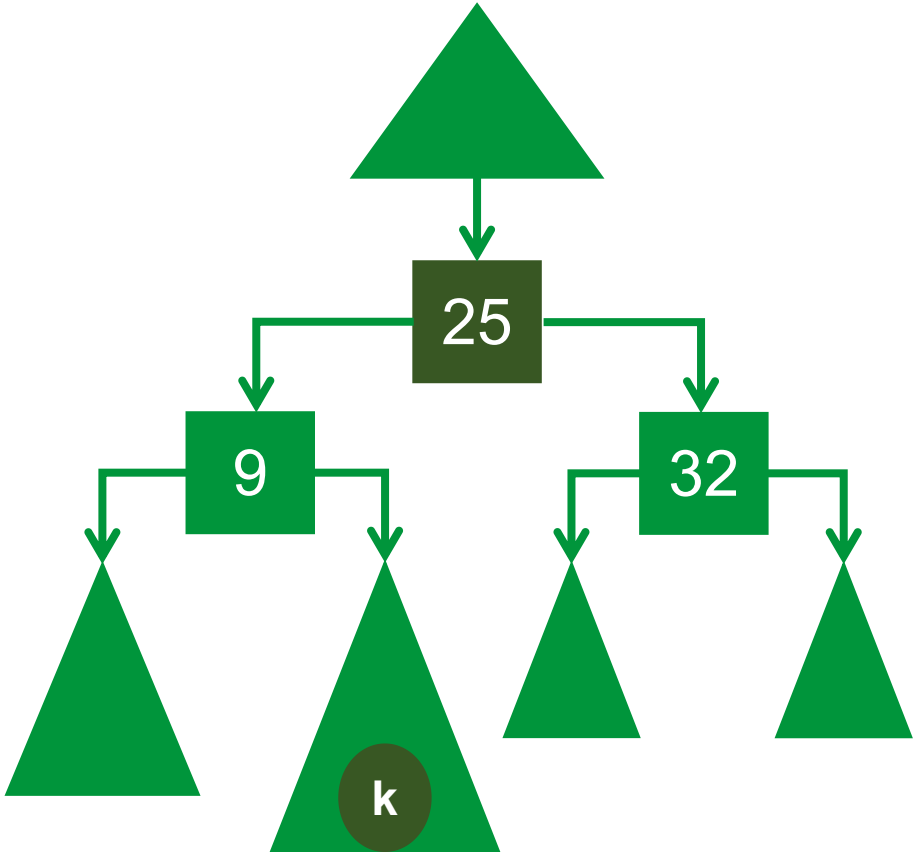


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Rotação à esquerda-direita (LR-rotation): ocorre quando a inserção é realizada na subárvore direita do filho esquerdo de um nó desbalanceado.
- Esse caso caracteriza o padrão de desbalanceamento esquerda-direita (LR).

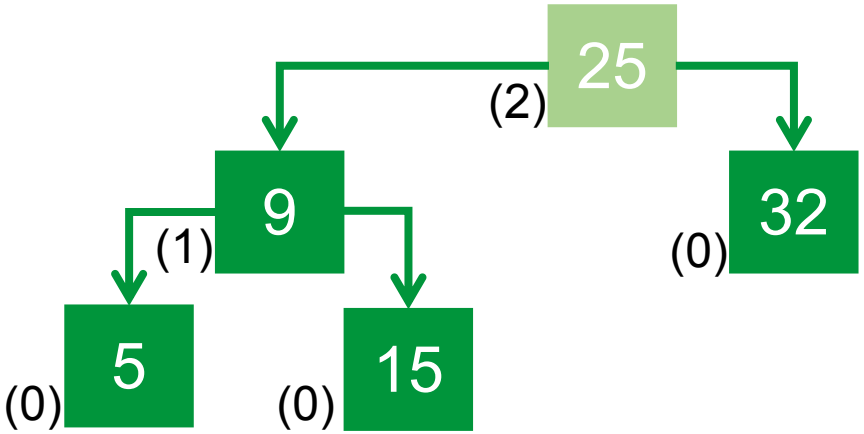
```
if (fb > 1 && chave > raiz->esquerda->chave) {
    raiz->esquerda = rotacaoEsquerda(raiz->esquerda);
    return rotacaoDireita(raiz);
}
```



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

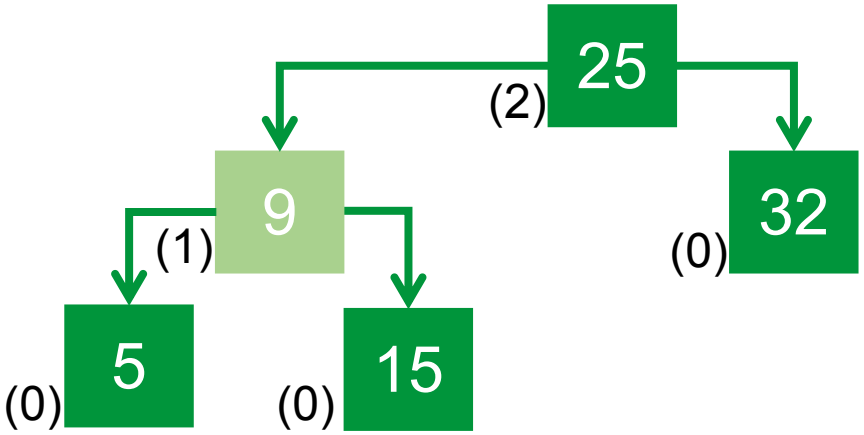
- Inserindo o nó 16:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

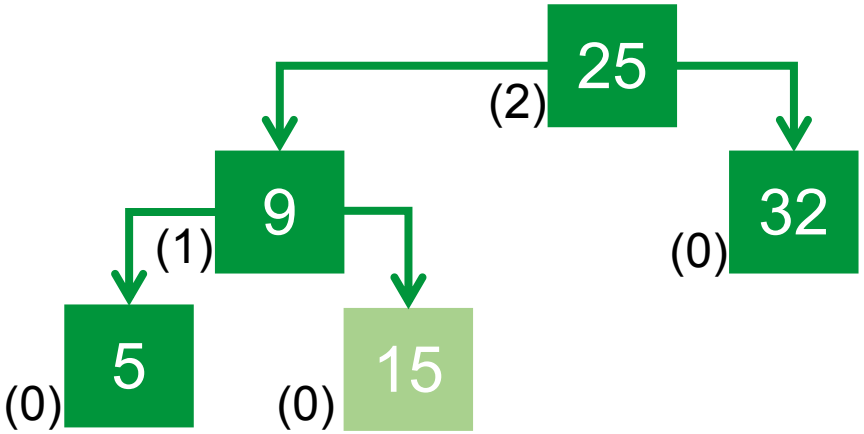
- Inserindo o nó 16:



Árvores AVL

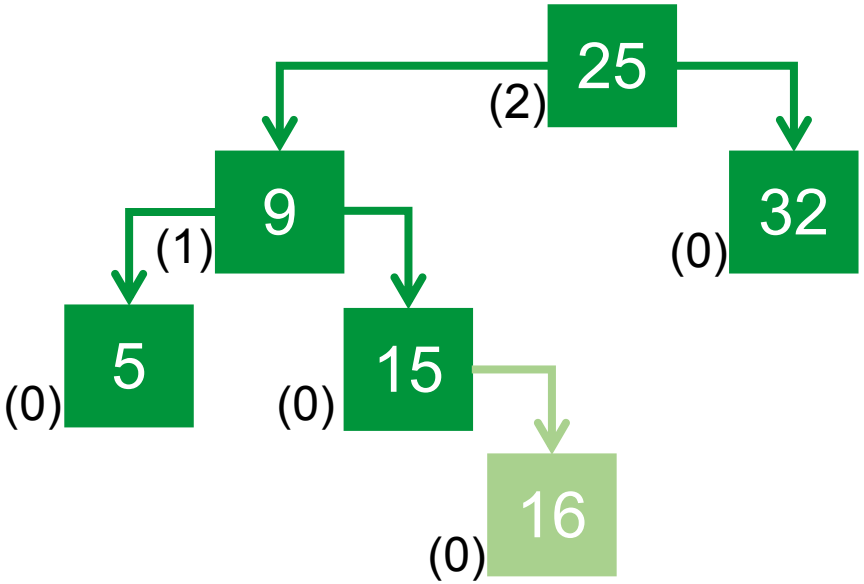
INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:



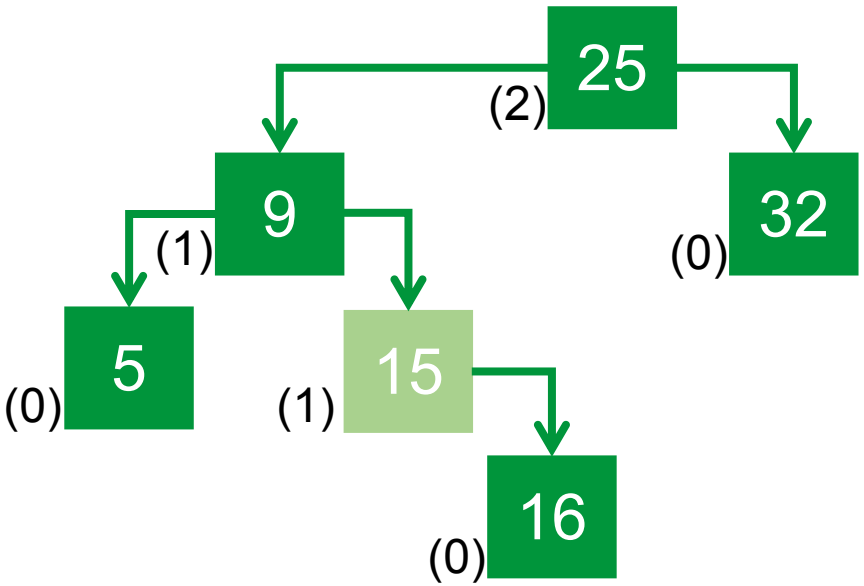
Árvores AVL

- Inserindo o nó 16:



Árvores AVL

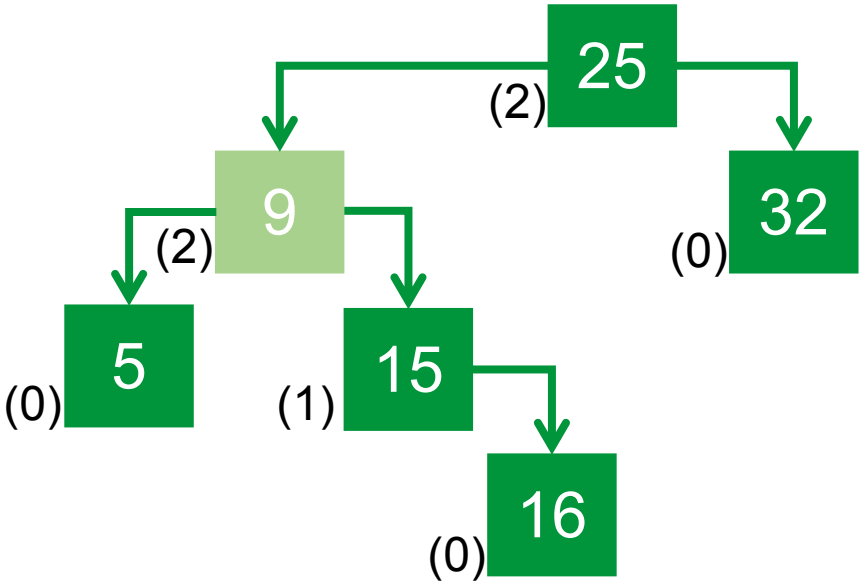
- Inserindo o nó 16:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

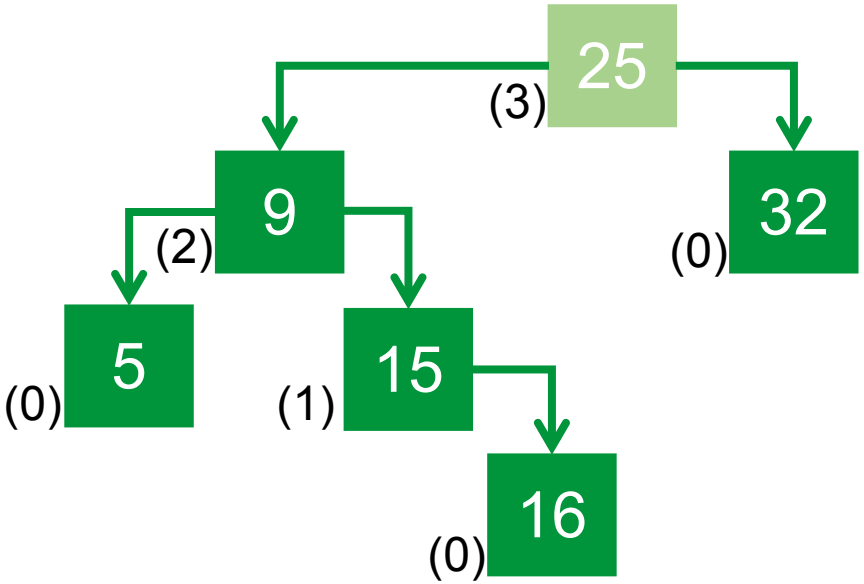


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

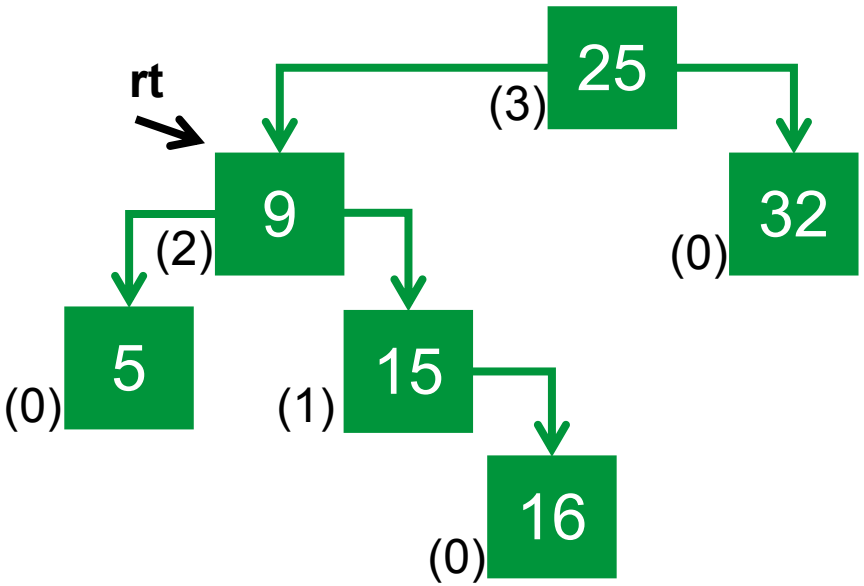
```
if (fb > 1 && chave > raiz->esquerda->chave) {
    raiz->esquerda = rotacaoEsquerda(raiz->esquerda);
    return rotacaoDireita(raiz);
}
```



Árvores AVL

- Inserindo o nó 16:

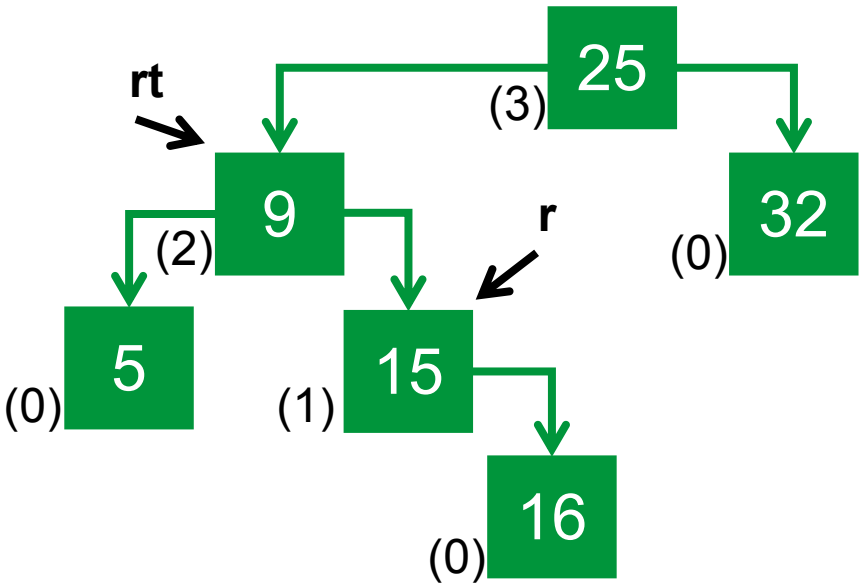
```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 16:

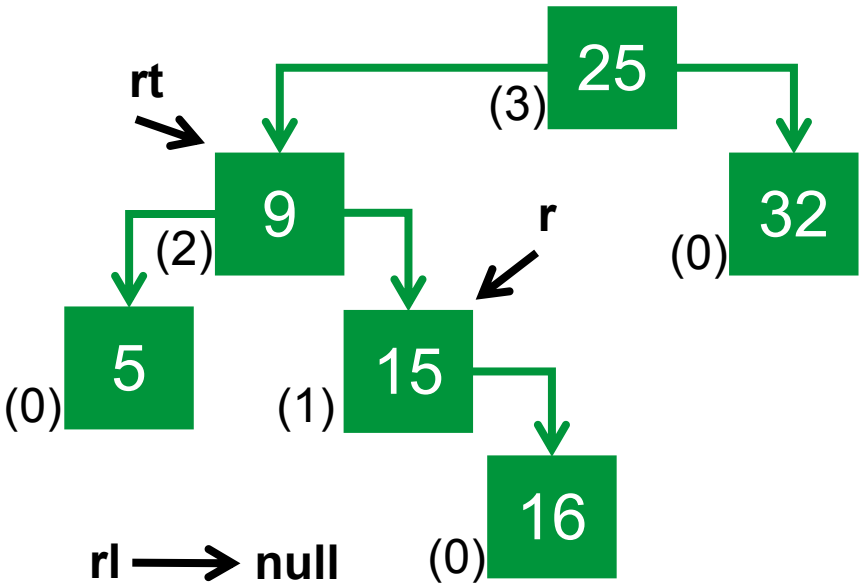
```c
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 16:

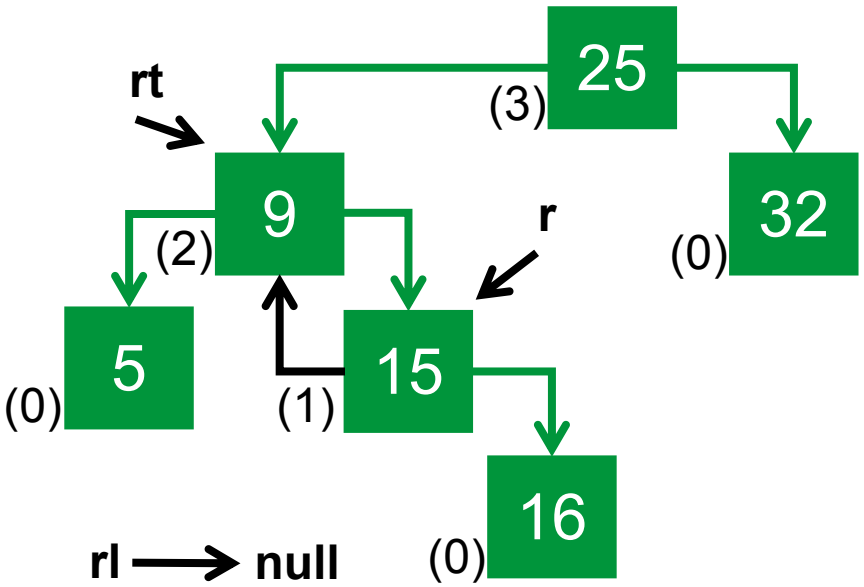
```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 16:

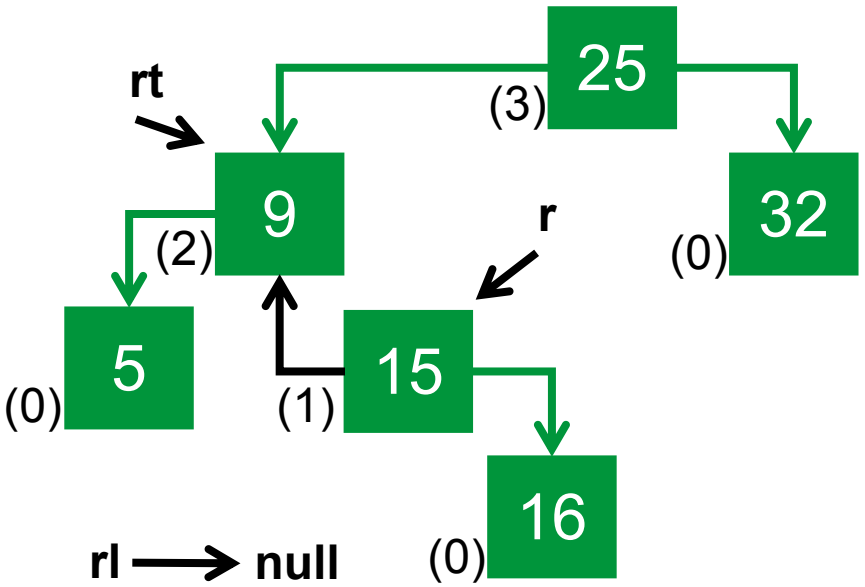
```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 16:

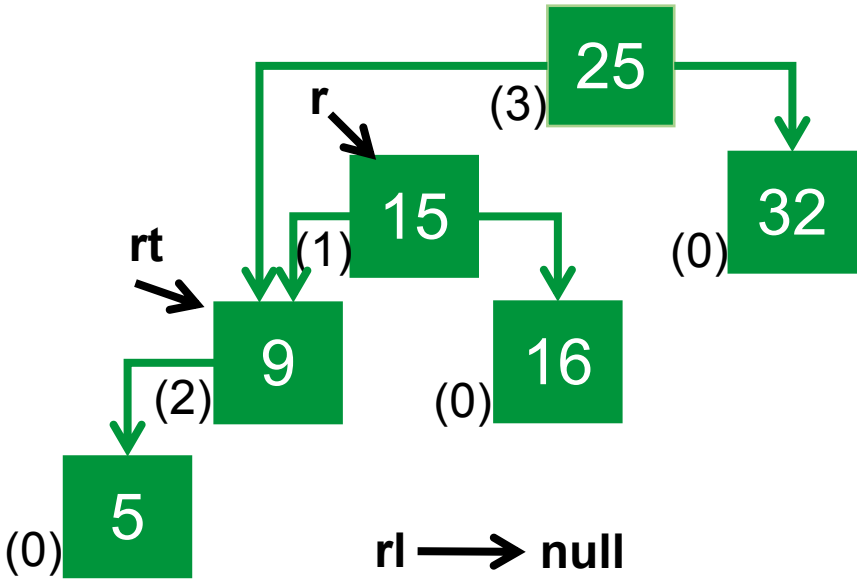
```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 16:

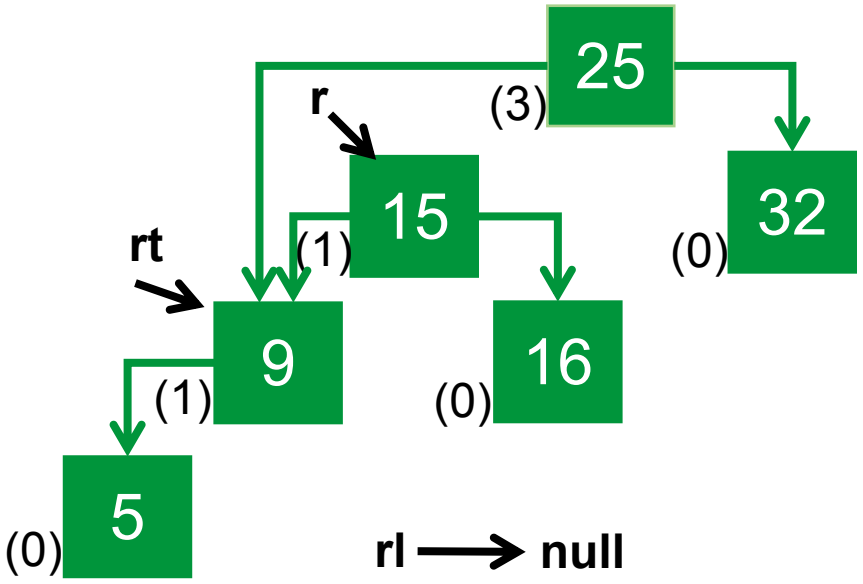
```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 16:

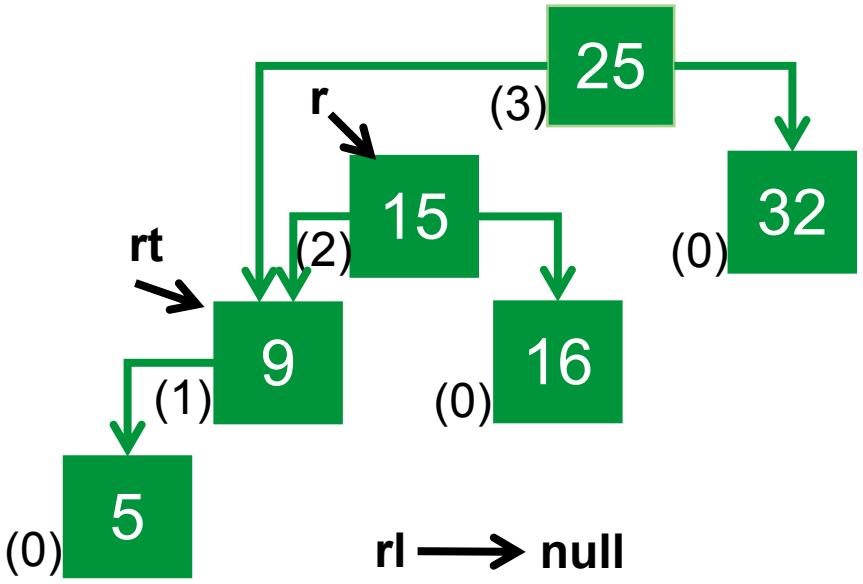
```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

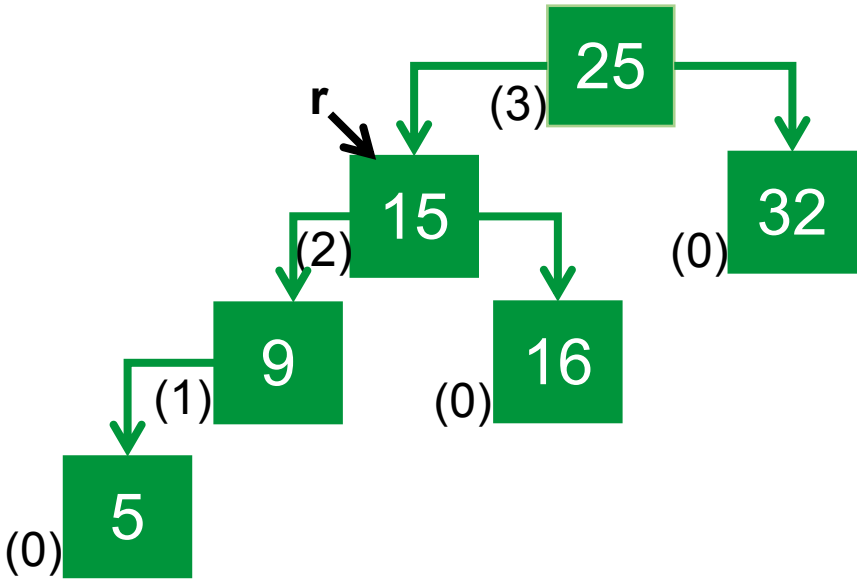
- Inserindo o nó 16:

```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 16:



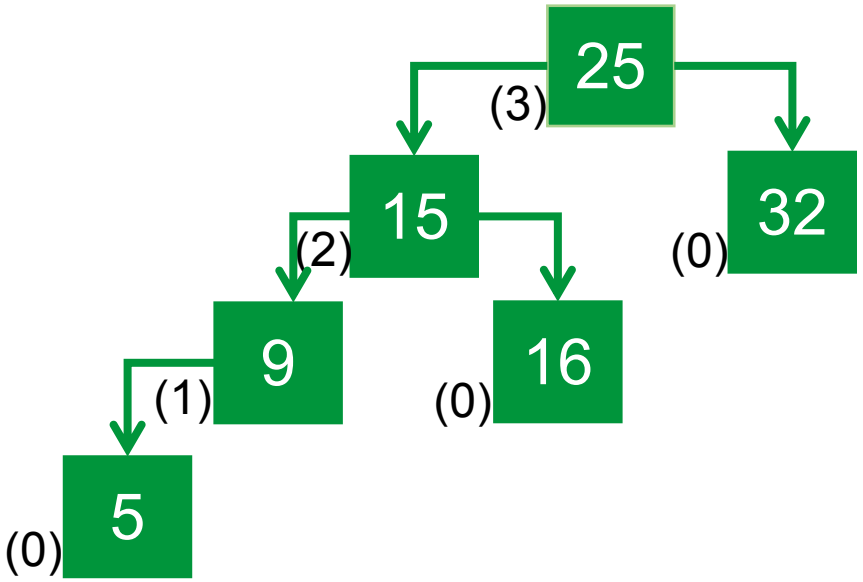

```c
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

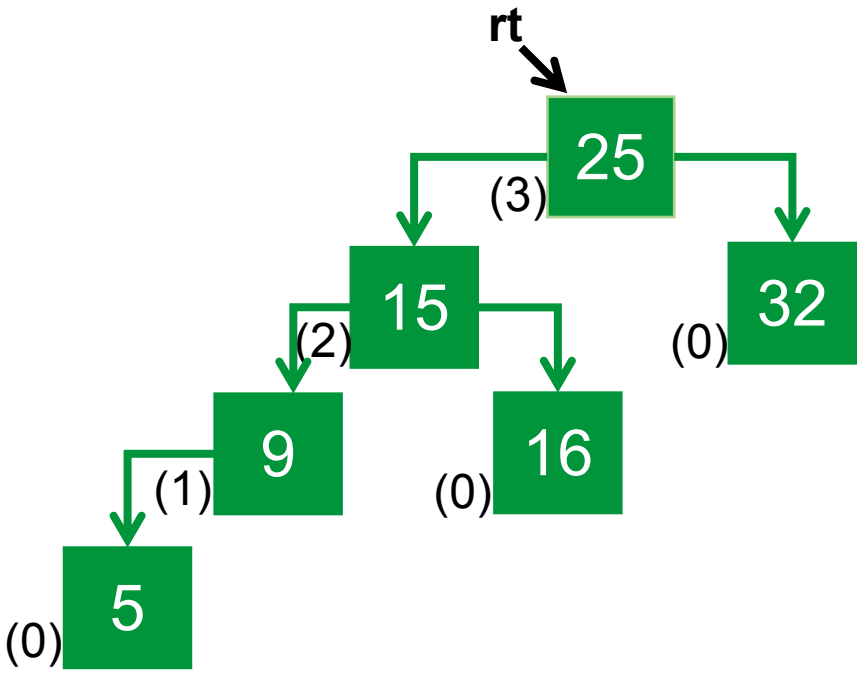
- Inserindo o nó 16:

```
if (fb > 1 && chave > raiz->esquerda->chave) {
    raiz->esquerda = rotacaoEsquerda(raiz->esquerda);
    return rotacaoDireita(raiz);
}
```



Árvores AVL

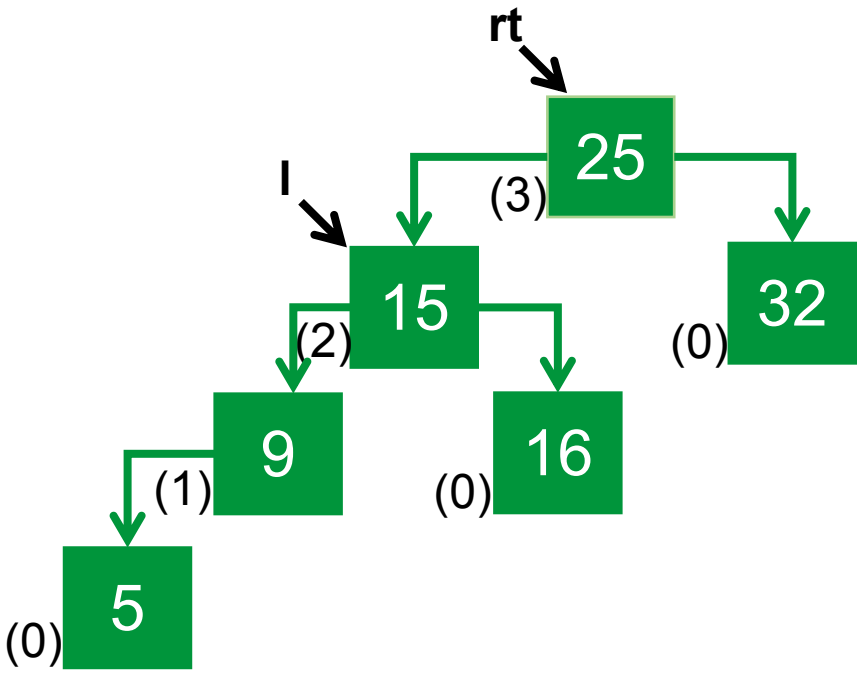
- Inserindo o nó 16:




```c
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

- Inserindo o nó 16:

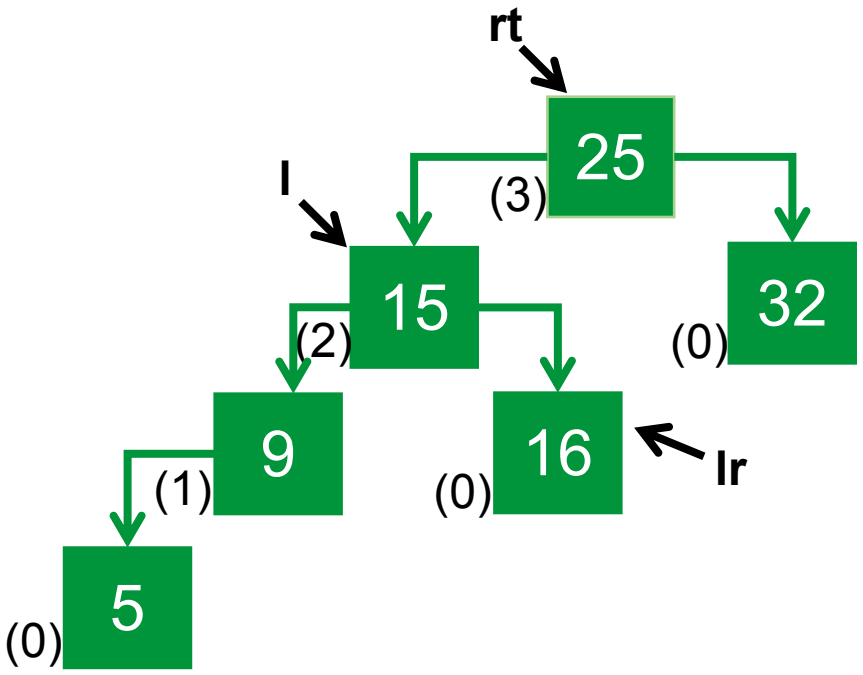



```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

INSTITUTO FEDERAL
Pernambuco

Árvores AVL

- Inserindo o nó 16:

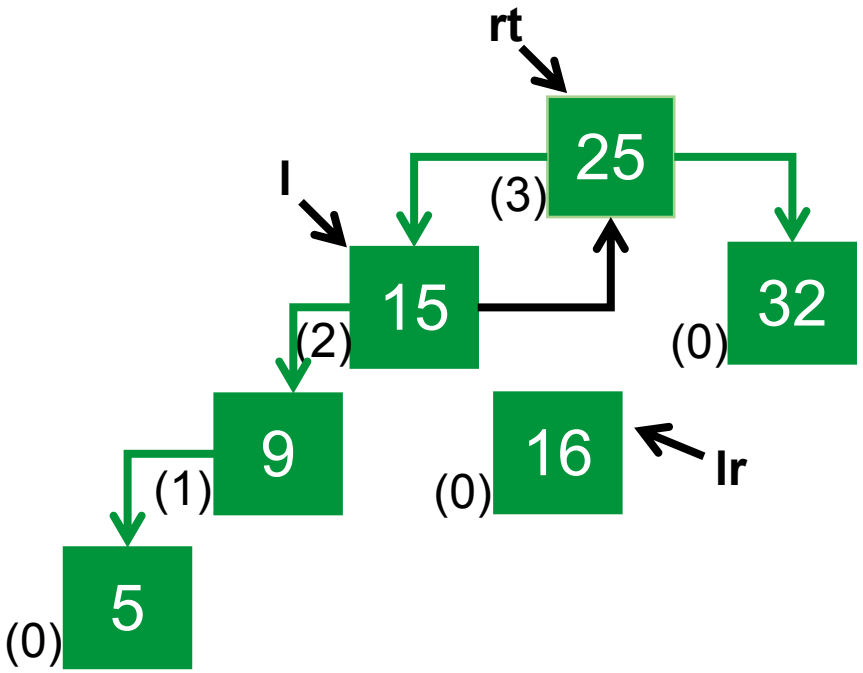



```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

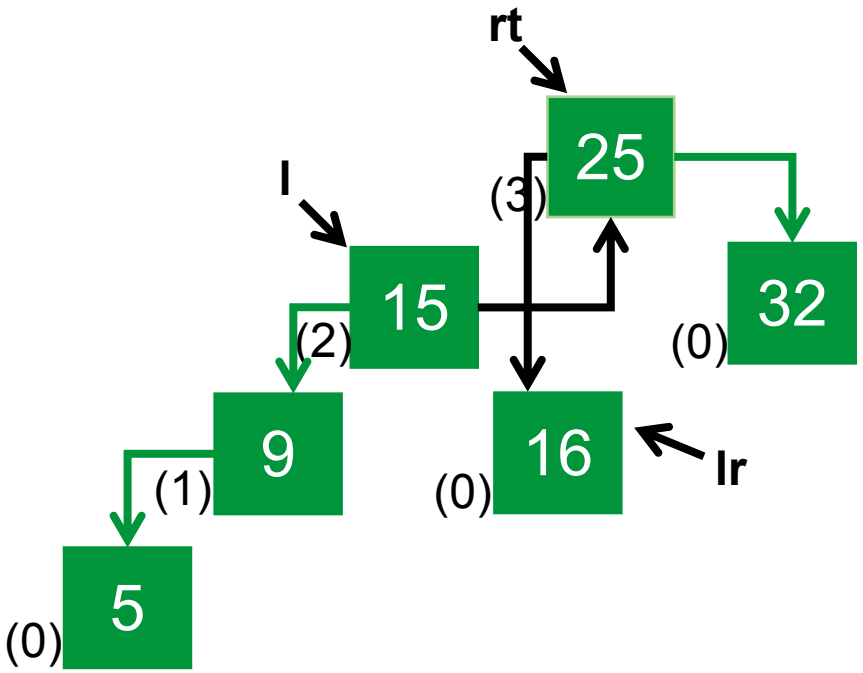
- Inserindo o nó 16:




```c
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

- Inserindo o nó 16:

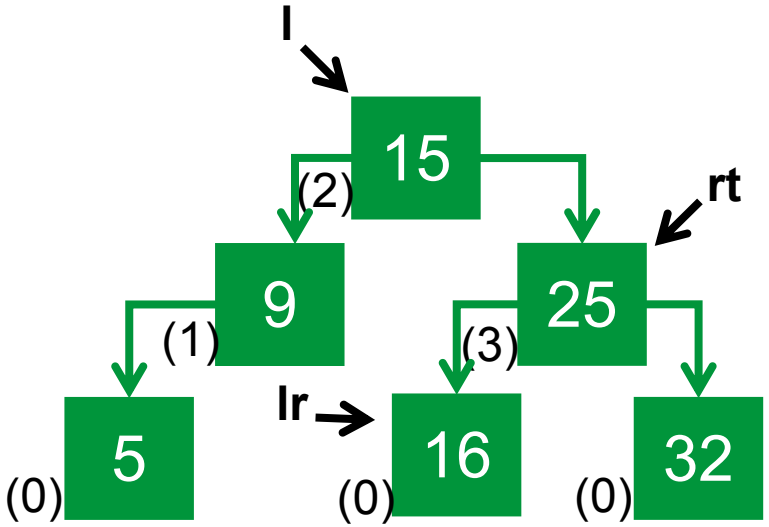



```c
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

- Inserindo o nó 16:

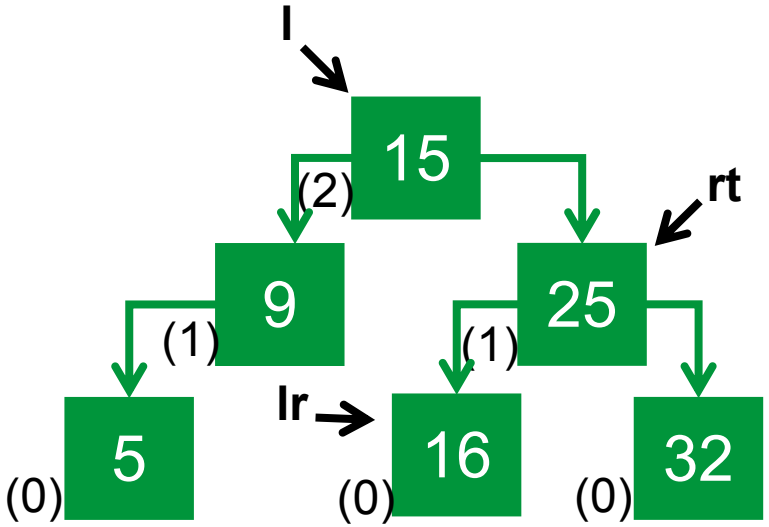
```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```



Árvores AVL

- Inserindo o nó 16:

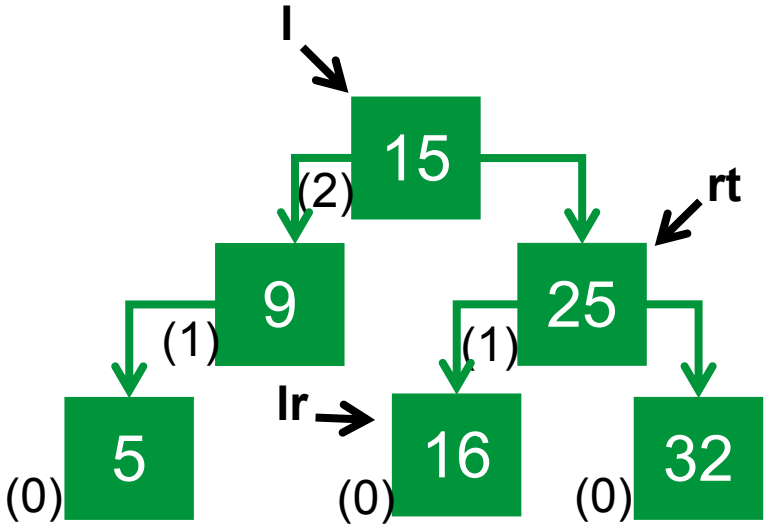
```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```



Árvores AVL

- Inserindo o nó 16:

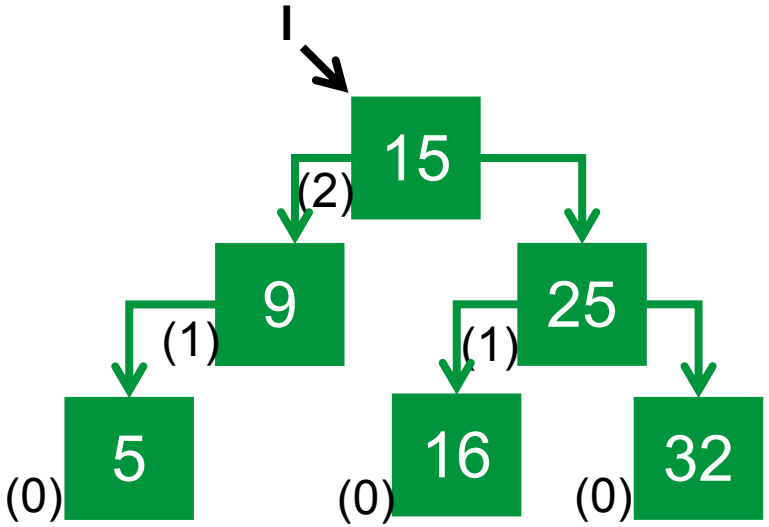
```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```



Árvores AVL

- Inserindo o nó 16:

```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

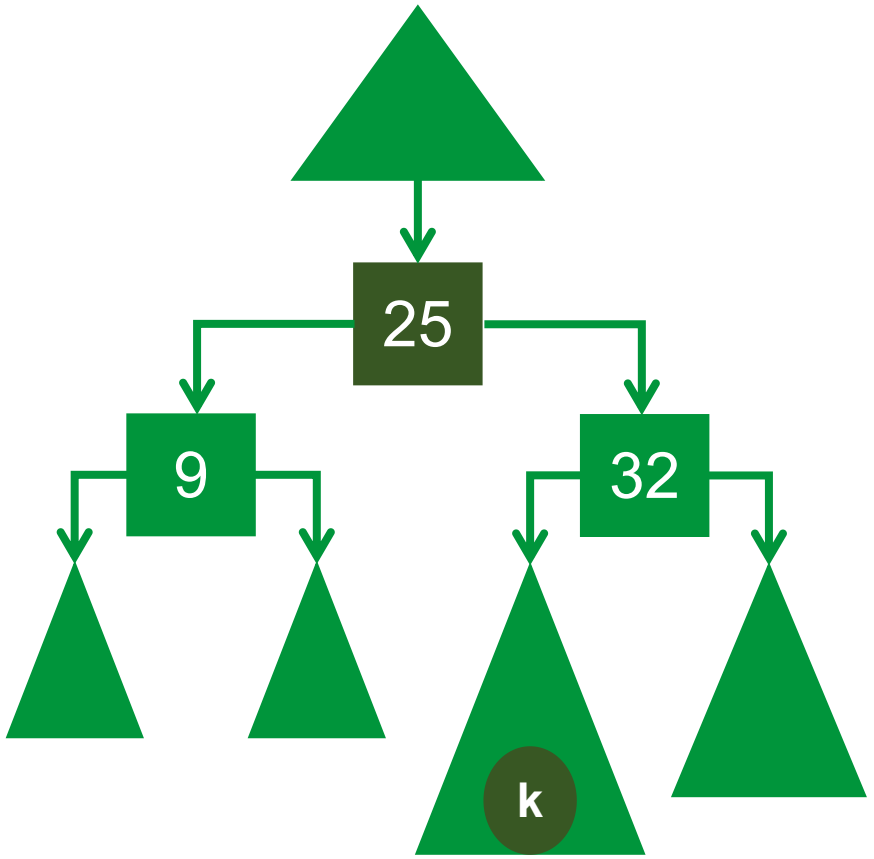


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Rotação à direita-esquerda (RL-rotation): ocorre quando a inserção é realizada na subárvore esquerda do filho direito de um nó desbalanceado.
- Esse caso caracteriza o padrão de desbalanceamento direita-esquerda (RL).

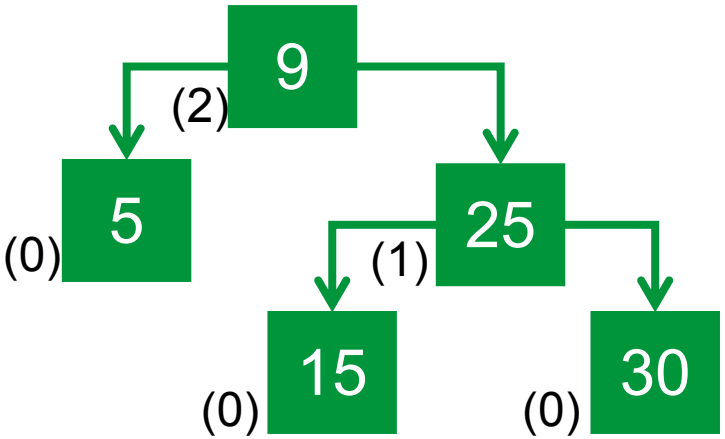
```
if (fb < -1 && chave < raiz->direita->chave) {
    raiz->direita = rotacaoDireita(raiz->direita);
    return rotacaoEsquerda(raiz);
}
```



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

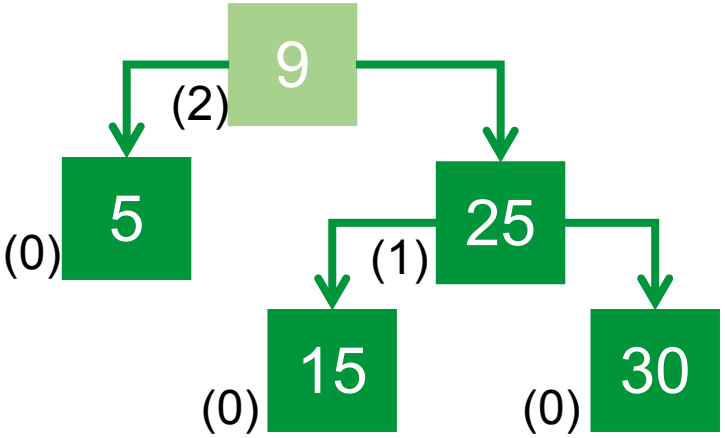
- Inserindo o nó 16:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

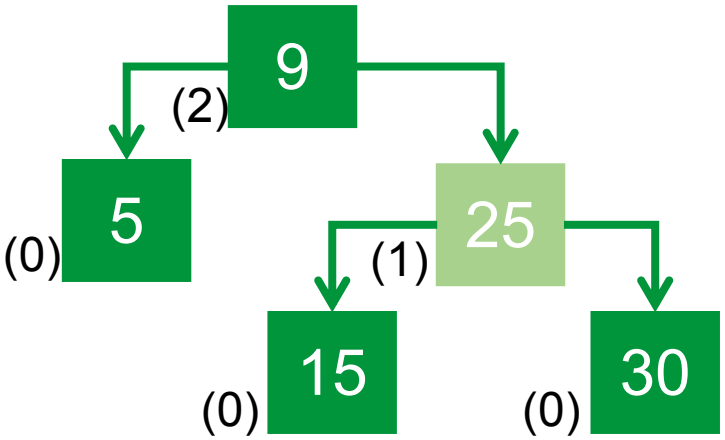
- Inserindo o nó 16:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

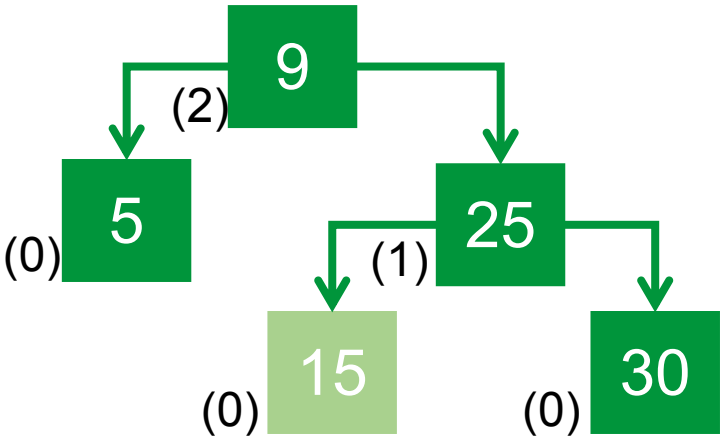
- Inserindo o nó 16:



Árvores AVL

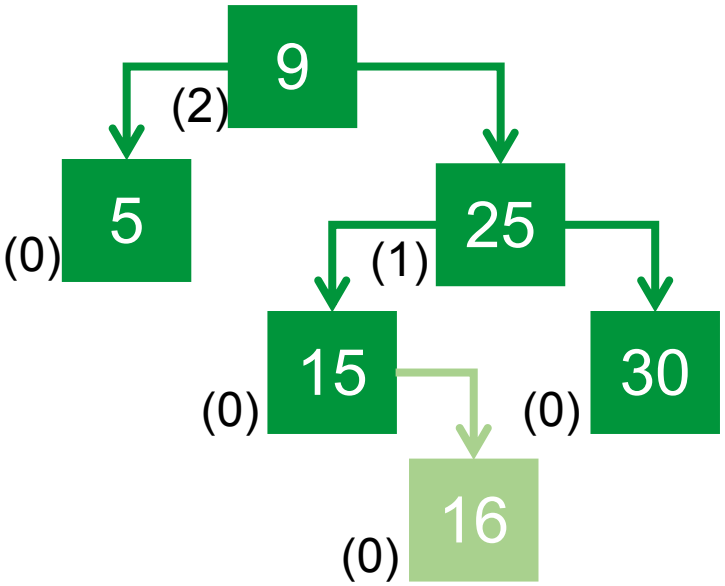
INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:



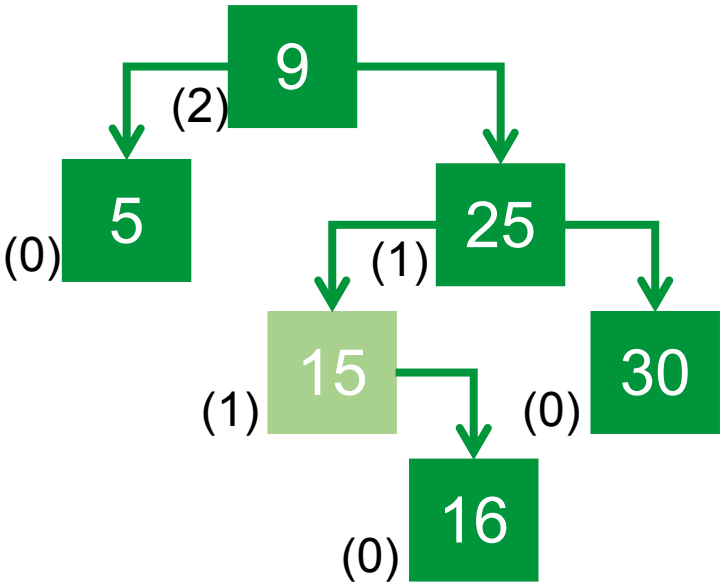
Árvores AVL

- Inserindo o nó 16:



Árvores AVL

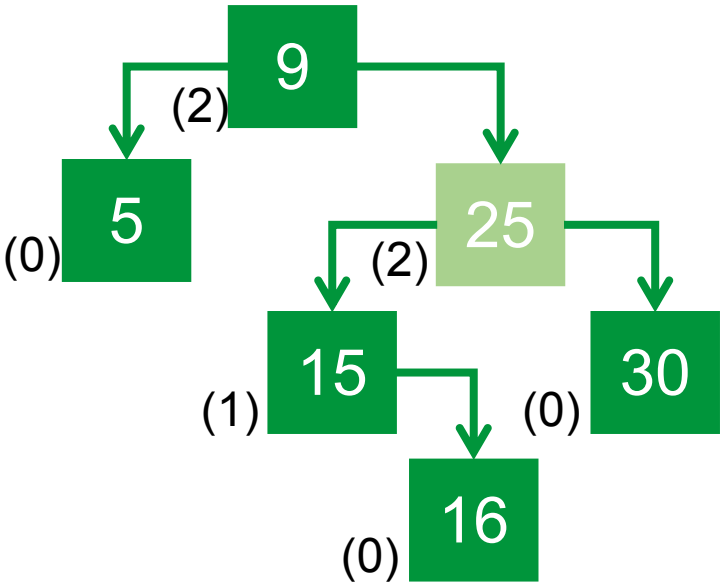
- Inserindo o nó 16:



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

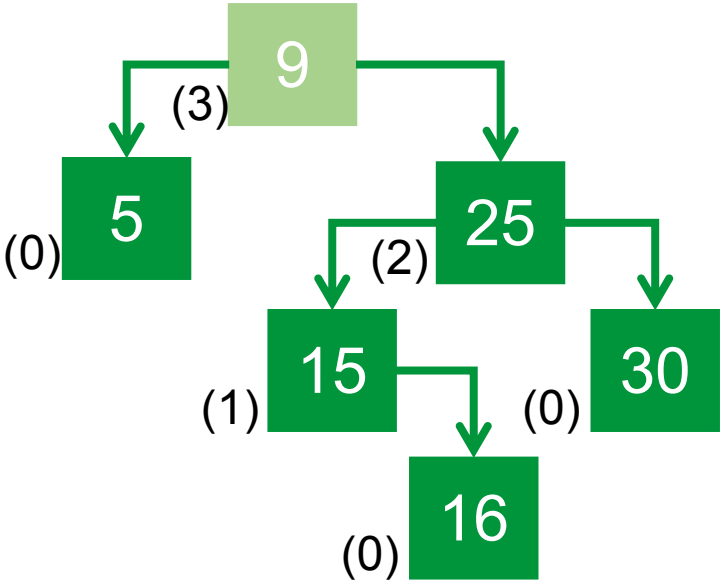


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

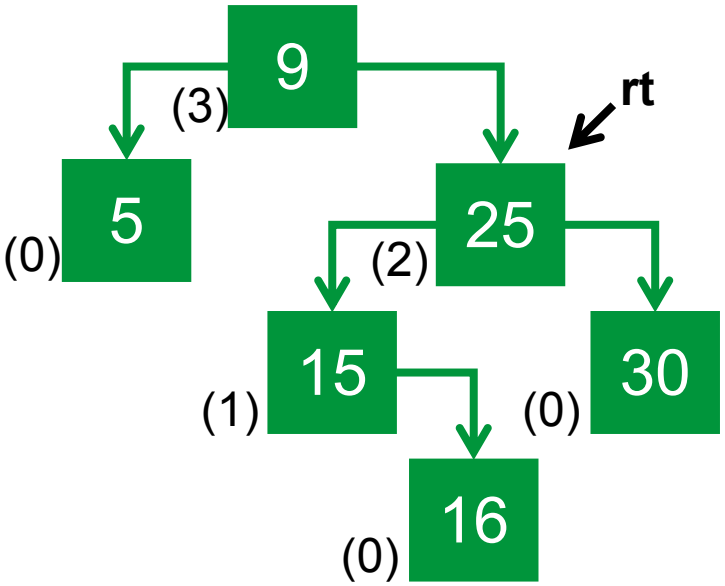
```
if (fb < -1 && chave < raiz->direita->chave) {
    raiz->direita = rotacaoDireita(raiz->direita);
    return rotacaoEsquerda(raiz);
}
```



Árvores AVL

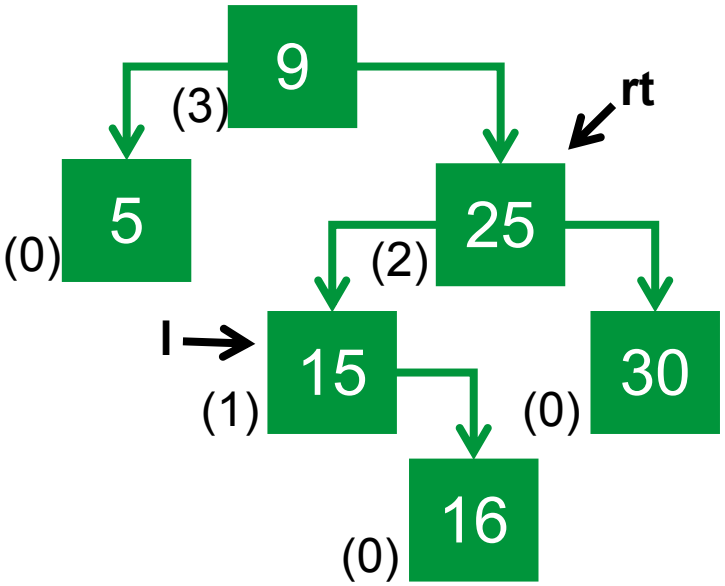
- Inserindo o nó 16:

```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```



Árvores AVL

- Inserindo o nó 16:

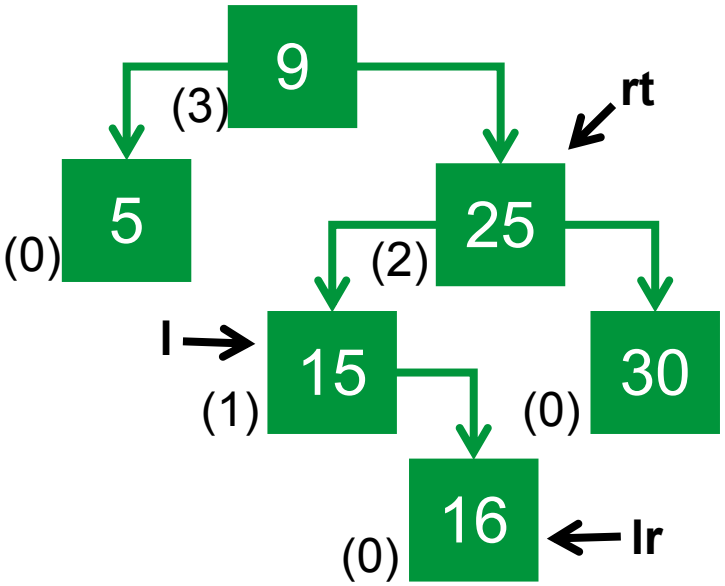



```c
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

- Inserindo o nó 16:

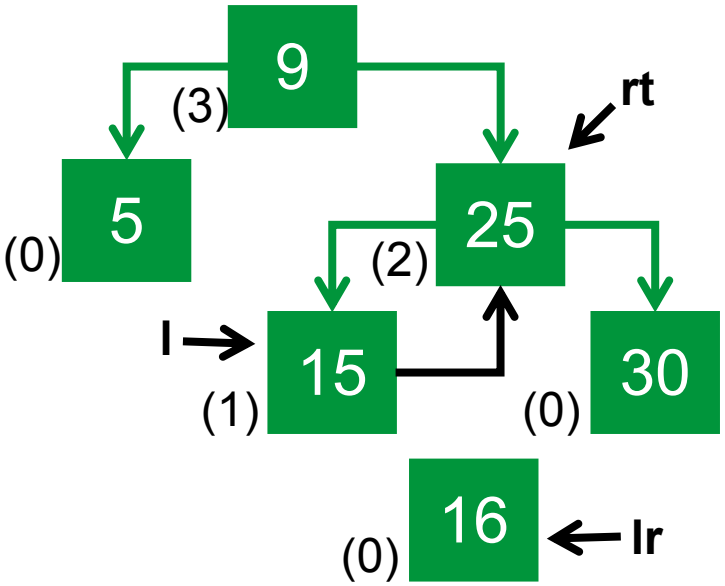
```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```



Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

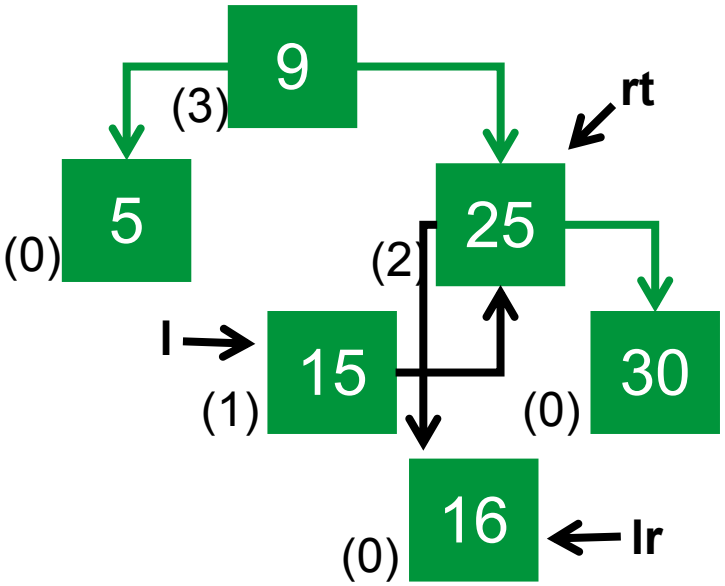



```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

Árvores AVL

- Inserindo o nó 16:

```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

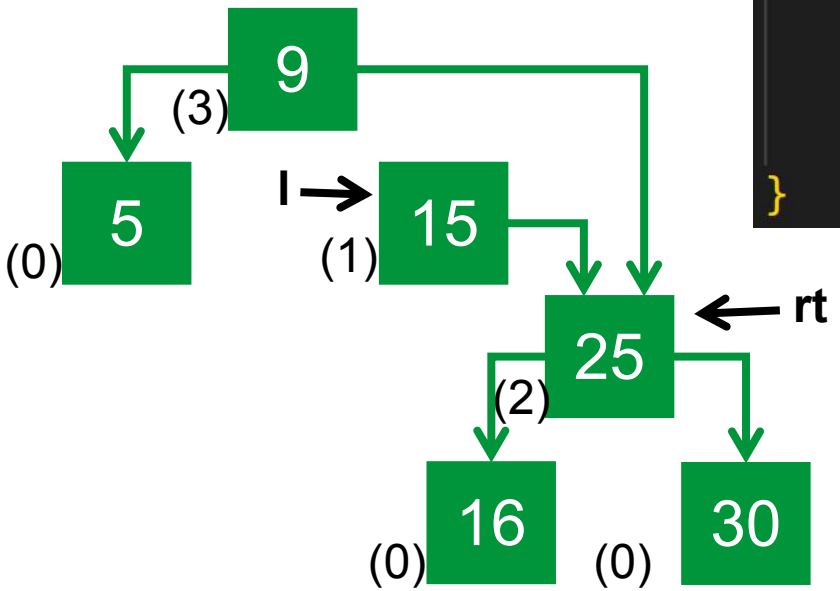


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

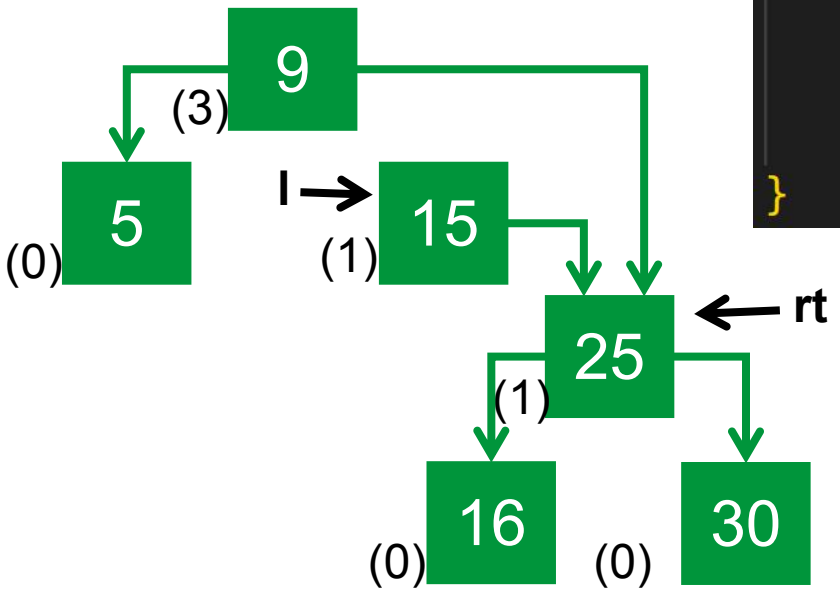
```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```



Árvores AVL

- Inserindo o nó 16:

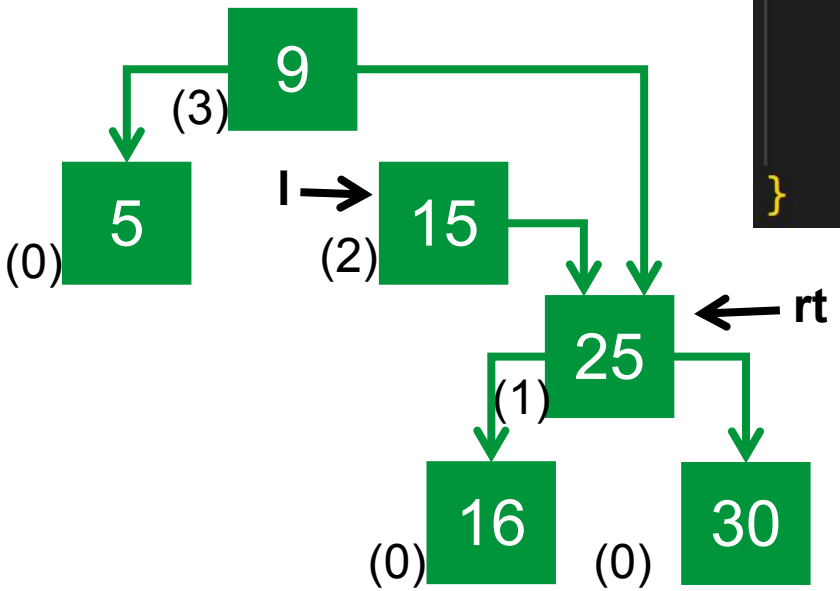
```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```



Árvores AVL

- Inserindo o nó 16:

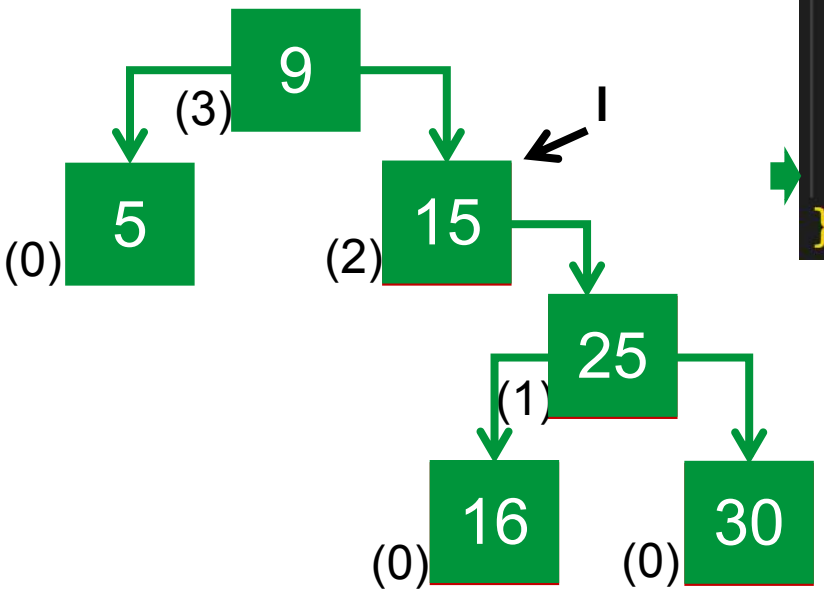
```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```



Árvores AVL

- Inserindo o nó 16:

```
No *rotacaoDireita(No *rt) {
    No *l = rt->esquerda;
    No *lr = l->direita;

    l->direita = rt;
    rt->esquerda = lr;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    l->altura = 1 + max(altura(l->esquerda), altura(l->direita));


    return l;
}
```

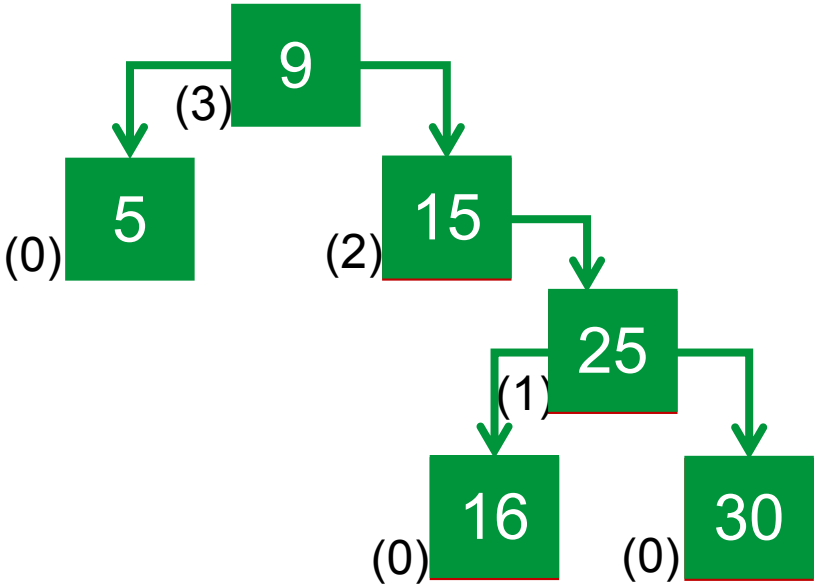


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

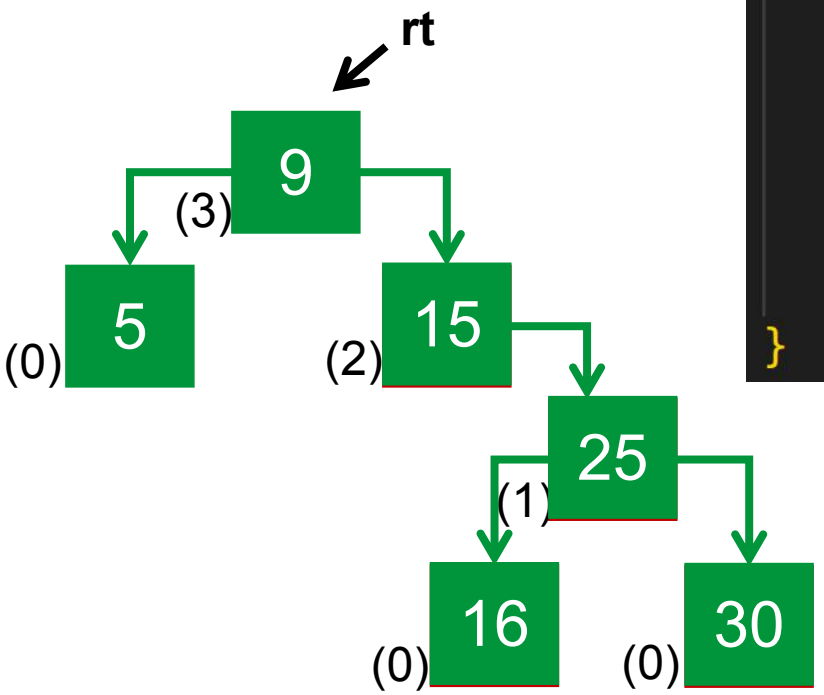
```
if (fb < -1 && chave < raiz->direita->chave) {
    raiz->direita = rotacaoDireita(raiz->direita);
    return rotacaoEsquerda(raiz);
}
```



Árvores AVL

- Inserindo o nó 16:

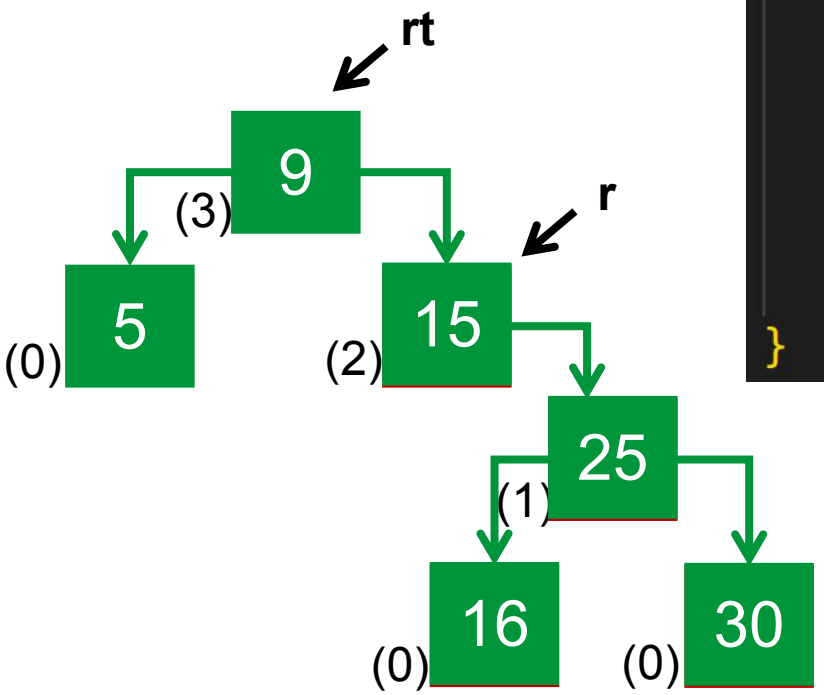
```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 16:

```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```

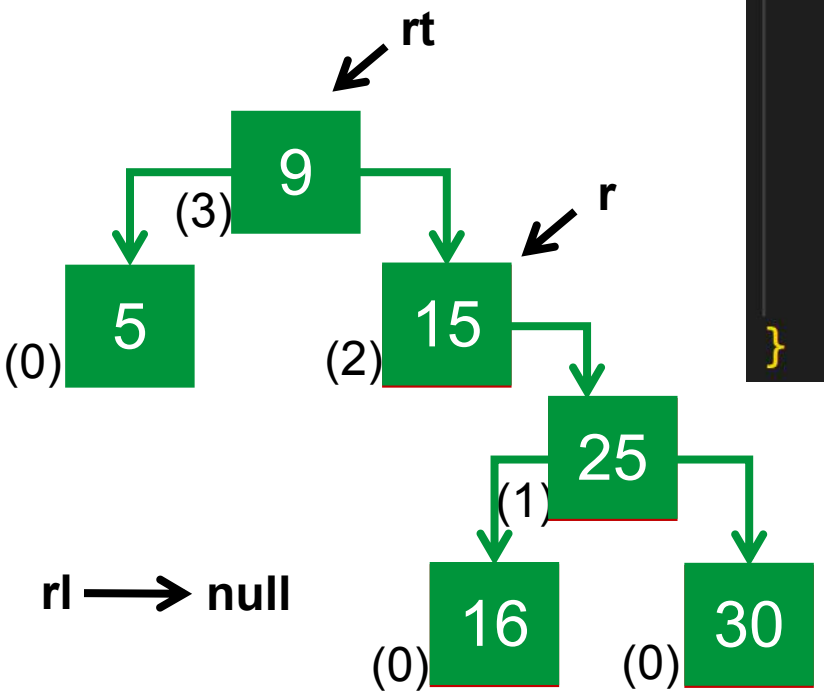


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```

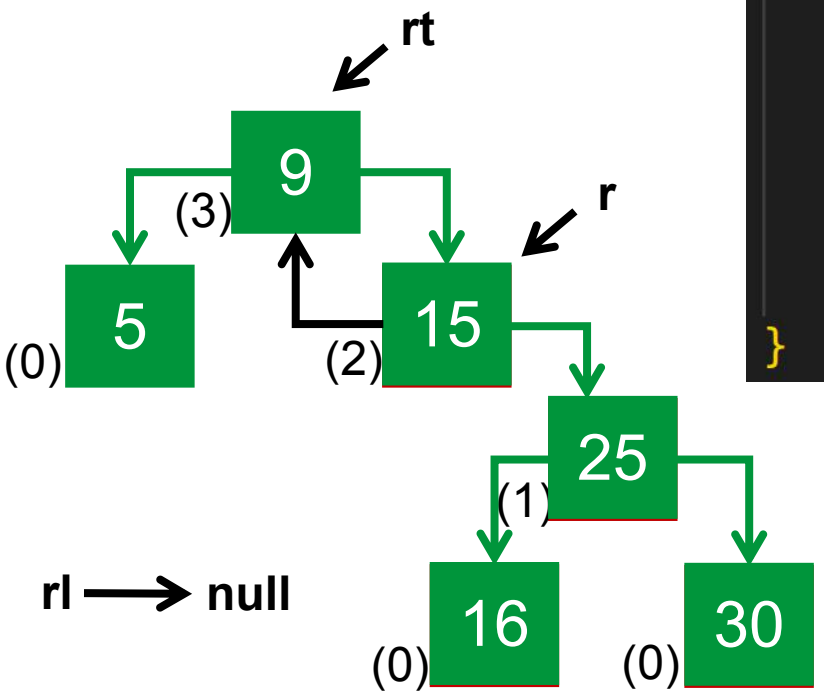


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

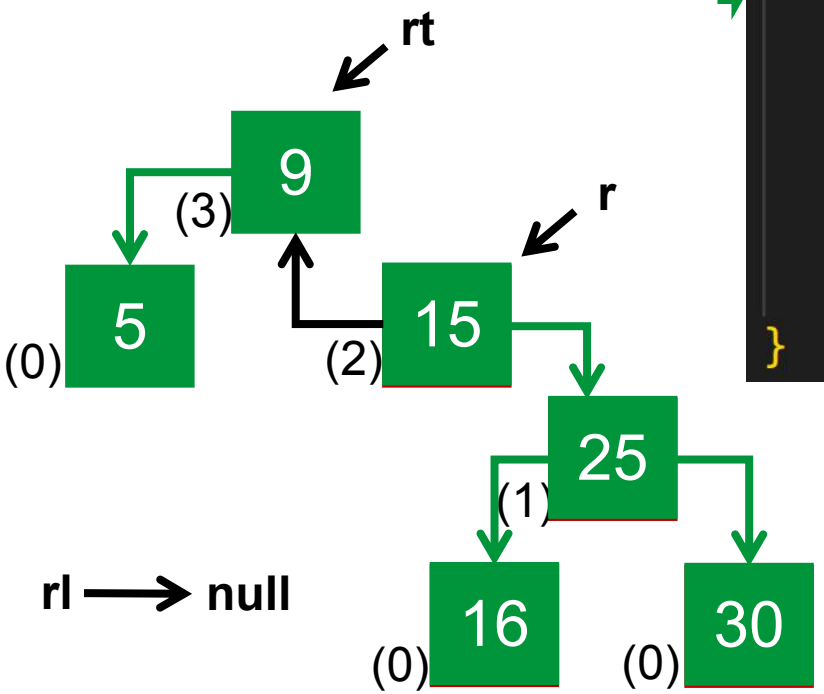
```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Inserindo o nó 16:

```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```

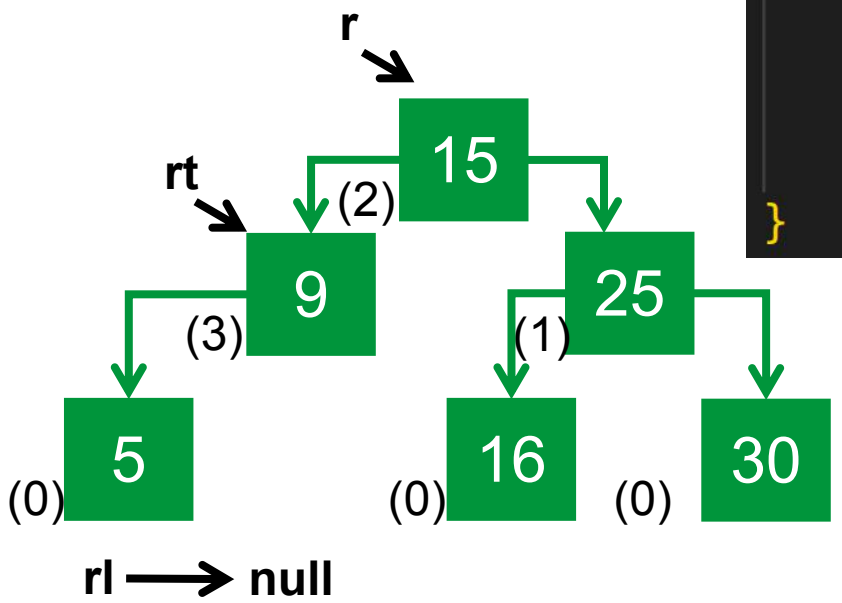


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```

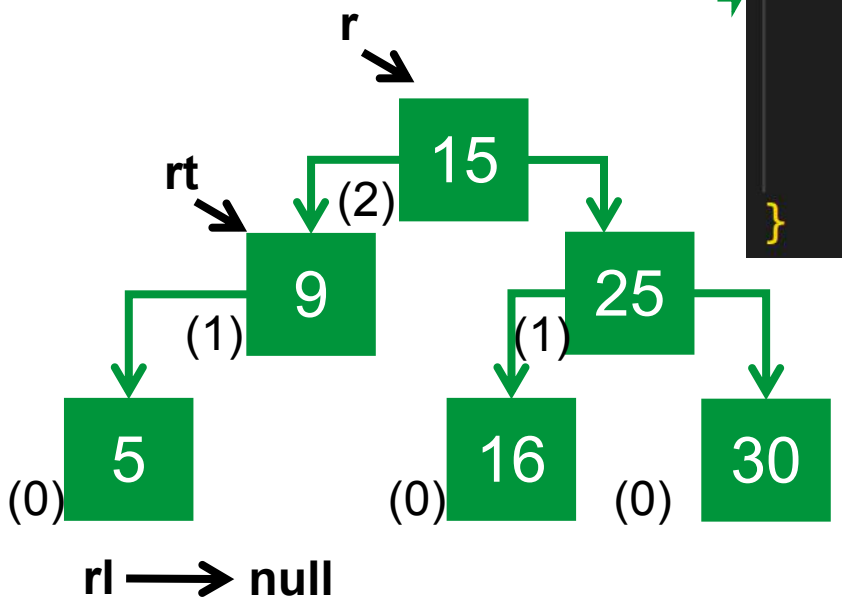


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

```
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```

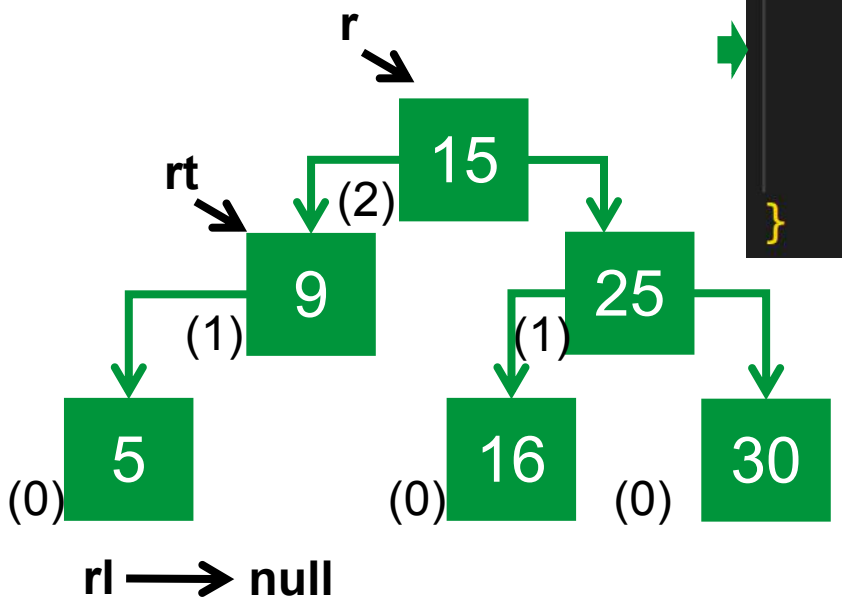


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

```c
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```

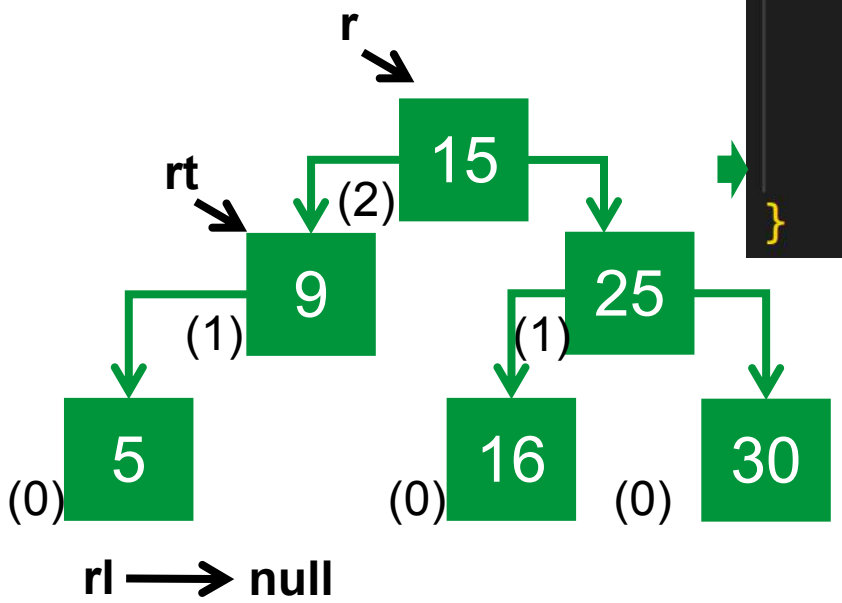


Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Inserindo o nó 16:

```c
No *rotacaoEsquerda(No *rt) {
    No *r = rt ->direita;
    No *rl = r->esquerda;

    r->esquerda = rt;
    rt->direita = rl;

    rt->altura = 1 + max(altura(rt->esquerda), altura(rt->direita));
    r->altura = 1 + max(altura(r->esquerda), altura(r->direita));

    return r;
}
```



Árvores AVL

- Implementação da inserção:

```c
No *inserir(No *raiz, int chave) {

    if (raiz == NULL)
        return criarNo(chave);

    if (chave < raiz->chave)
        raiz->esquerda = inserir(raiz->esquerda, chave);
    else if (chave > raiz->chave)
        raiz->direita = inserir(raiz->direita, chave);
    else
        return raiz;

    atualizarAltura(raiz);

    int fb = fatorBalanceamento(raiz);

    // LL
    if (fb > 1 && chave < raiz->esquerda->chave)
        return rotacaoDireita(raiz);
    // RR
    if (fb < -1 && chave > raiz->direita->chave)
        return rotacaoEsquerda(raiz);
    // LR
    if (fb > 1 && chave > raiz->esquerda->chave) {
        raiz->esquerda = rotacaoEsquerda(raiz->esquerda);
        return rotacaoDireita(raiz);
    }
    // RL
    if (fb < -1 && chave < raiz->direita->chave) {
        raiz->direita = rotacaoDireita(raiz->direita);
        return rotacaoEsquerda(raiz);
    }
    return raiz;
}
```

INSTITUTO FEDERAL
Pernambuco

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

```
No *inserir(No *raiz, int chave) {

    if (raiz == NULL)
        return criarNo(chave);

    if (chave < raiz->chave)
        raiz->esquerda = inserir(raiz->esquerda, chave);
    else if (chave > raiz->chave)
        raiz->direita = inserir(raiz->direita, chave);
    else
        return raiz;

    atualizarAltura(raiz);
    int fb = fatorBalanceamento(raiz);
```

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

```
    // LL
    if (fb > 1 && chave < raiz->esquerda->chave)
        return rotacaoDireita(raiz);
    // RR
    if (fb < -1 && chave > raiz->direita->chave)
        return rotacaoEsquerda(raiz);
    // LR
    if (fb > 1 && chave > raiz->esquerda->chave) {
        raiz->esquerda = rotacaoEsquerda(raiz->esquerda);
        return rotacaoDireita(raiz);
    }
    // RL
    if (fb < -1 && chave < raiz->direita->chave) {
        raiz->direita = rotacaoDireita(raiz->direita);
        return rotacaoEsquerda(raiz);
    }
    return raiz;
}
```

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

```
void atualizarAltura(No *n) {
    n->altura = 1 + max(altura(n->esquerda), altura(n->direita));
}
```

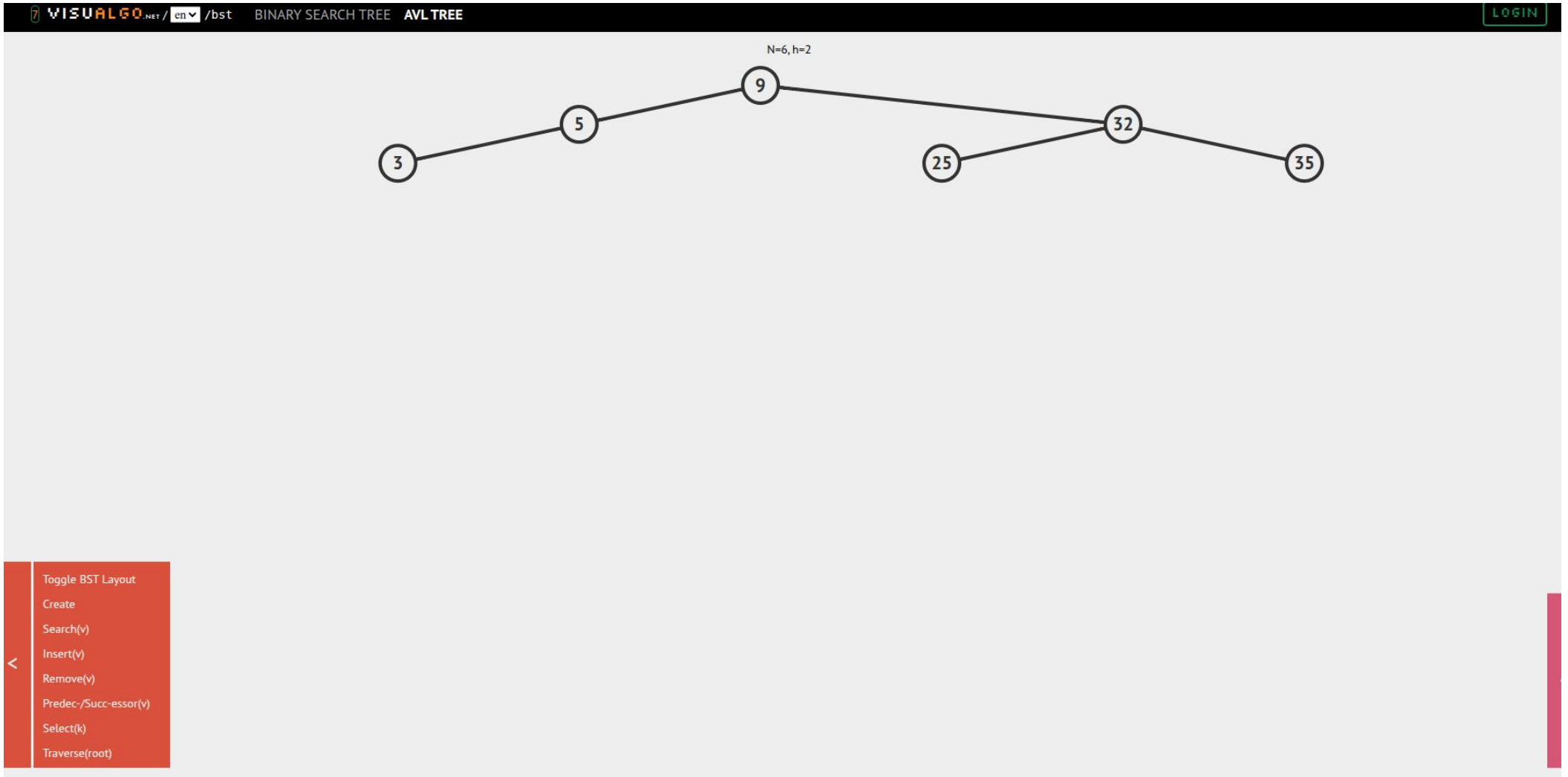
Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- A implementação da remoção em árvores AVL segue os mesmos passos da remoção em uma árvore de busca binária (BST). Entretanto, após a remoção, pode ser necessário realizar rotações para restaurar o balanceamento da árvore.
- Após a remoção, as alturas dos nós ancestrais devem ser atualizadas e o fator de balanceamento deve ser verificado. Caso algum nó fique desbalanceado, aplicam-se as rotações apropriadas (LL, RR, LR ou RL) para restaurar as propriedades da árvore AVL.

Árvores AVL

INSTITUTO FEDERAL
Pernambuco



https://visualgo.net/en/bst

Árvores AVL

INSTITUTO FEDERAL
Pernambuco

- Vantagens
  - Mantém a árvore sempre balanceada.
  - Busca, inserção e remoção com complexidade O(log n).
  - Melhor desempenho em operações de busca quando comparada a uma BST desbalanceada.
  - Garante altura O(log n), evitando casos degenerados.
- Desvantagens
  - Implementação mais complexa que uma BST.
  - Inserções e remoções exigem atualizar as alturas e, eventualmente, rotações.
  - Cada nó armazena informações adicionais, aumentando o consumo de memória.

Outras árvores balanceadas

INSTITUTO FEDERAL
Pernambuco

- Árvore Vermelho-Preta (Red-Black Tree)
- Árvore B (B-Tree)
- Árvore B+ (B+ Tree)
- Árvore Splay (Splay Tree)
- Árvore 2-3 (2-3 Tree)
- Árvore 2-3-4 (2-3-4 Tree)
- Árvore AA (AA Tree)
- Árvore Treap
- Árvore Peso-Balanceada (Weight-Balanced Tree)

Exercício

1. Implemente a operação de remoção em uma árvore AVL. Após a implementação, utilize um simulador de árvores AVL para testar diferentes casos de remoção e verificar se a árvore permanece balanceada após cada operação.

Dúvidas

INSTITUTO FEDERAL
Pernambuco

ALGORITMOS E ESTRUTURA DE DADOS

Curso de Engenharia de Software
Lucas Sampaio Leite