

Árvore Binária de Pesquisa Balanceada

Adelson-Velskii Landis (AVL)

**Implementação com ligações simples
(usando ponteiros e memória dinâmica)**

Definição

O equilíbrio perfeito de uma árvore

- Acontece quando as subárvores esquerda e direita têm a mesma altura
- Exigiria algoritmos de inserção e de remoção muito complexos
- É conveniente definir uma condição de equilíbrio que seja fácil de manter

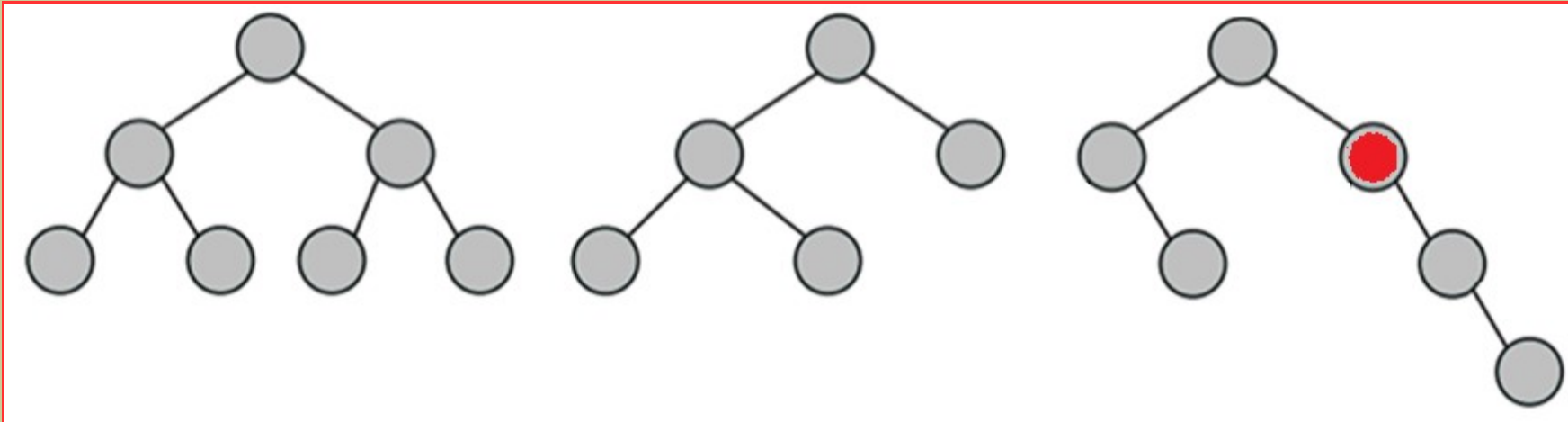
Uma boa condição de equilíbrio consiste na seguinte definição

- Uma árvore equilibrada em altura é uma árvore em que a diferença das alturas entre as subárvores esquerda e direita, em cada nodo, não seja superior a uma unidade

Assim, a árvore com a “melhor altura”

- É aquela que tem o maior número de nodos para uma qualquer altura (a árvore binária de pesquisa completa)

Exemplos

Árvore **AVL**Árvore **AVL**Árvore **não AVL**

Operações sobre uma Árvore AVL

- Seguem as mesmas estratégias usadas para as ABP, com exceção das operações para **inserir** e **remover**
- Depois de se **inserir** ou **remover** um elemento de uma árvore, é necessário
 - atualizar as alturas
 - reequilibrar a árvore, caso tenha ficado desequilibrada

Operações básicas sobre uma Árvore AVL

Criar uma Árvore AVL (vazia)

Libertar/destruir uma Árvore AVL

Verificar se uma Árvore AVL está vazia

Mostrar os elementos duma Árvore AVL: pré-ordem, em-ordem e pós-ordem

Pesquisar um elemento numa Árvore AVL

Consultar um elemento duma Árvore AVL

Atualizar um elemento duma Árvore AVL

Inserir um elemento numa Árvore AVL

Remover um elemento duma Árvore AVL

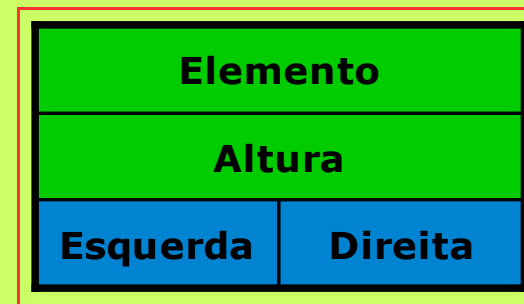
Nodos e ligações

Manipulação de uma ABP do tipo AVL

- Para manipular uma árvore AVL é preciso guardar, em cada nodo, a sua **altura**, que representa a altura da subárvore com raiz nesse nodo
- Assim, o nodo de uma árvore AVL é idêntico ao nodo de uma ABP, à qual se juntou o campo **Altura**

Definição de um nodo de uma árvore AVL

```
struct NodoAVL {  
    DadosAVL Elemento;  
    int Altura;  
    struct NodoAVL *Esquerda;  
    struct NodoAVL *Direita;  
};  
typedef struct NodoAVL *PNodoAVL;
```



Operações sobre a estrutura DadosAVL

Mostrar os dados de um elemento do tipo DadosAVL

```
void mostrarElementoAVL (DadosAVL X)
```

- operação que mostra os dados/informação do elemento X do tipo DadosAVL

Criar um elemento do tipo DadosAVL

```
DadosAVL criarElementoAVL ()
```

- operação para criar um elemento do tipo DadosAVL, com dados vindos de fonte adequada

Comparar 2 elementos, X e Y, do tipo DadosAVL

```
int compararElementosAVL (DadosAVL X, DadosAVL Y)
```

- operação que compara dois elementos do tipo DadosAVL, segundo um dos seus campos
- devolve: **-1** ($X < Y$), **0** ($X = Y$) ou **1** ($X > Y$)

Operações sobre um nodo de uma árvore AVL

Criar um nodo de uma árvore AVL

Entrada: a informação X que se pretende guardar
Saída: ponteiro para um nodo com a informação X

PNodeAVL criarNodeAVL (DadosAVL X)

```
{  
    PNodeAVL P;  
    P = (PNodeAVL) malloc(sizeof(struct NodeAVL));  
    if (P == NULL)  
        return NULL;  
    P→Elemento = X;  
    P→Altura = 0;  
    P→Esquerda = NULL;  
    P→Direita = NULL;  
    return P;  
}
```

Libertar/destruir um nodo de uma árvore AVL

- Adaptar a partir da implementada para Árvores Binárias de Pesquisa

Entrada: um ponteiro para o nodo que se pretende destruir

Saída: o ponteiro a apontar para NULL

PNodeAVL libertarNodoAVL (PNodeAVL P)

```
{  
    P→Esquerda = NULL;  
    P→Direita = NULL;  
    free(P);  
    P = NULL;  
    return P;  
}
```

Operações sobre uma Árvore AVL

Criar uma Árvore AVL (vazia)

- Adaptar a partir da implementada para Árvores Binárias de Pesquisa

Entrada: --

Saída: devolve um AVL vazia (sem elementos)

PNodeAVL criarAVL ()

```
{  
    PNodeAVL T;  
    T = NULL;  
    return T;  
}
```

Libertar/destruir uma Árvore AVL

- Adaptar a partir da implementada para Árvores Binárias de Pesquisa

Entrada: uma AVL T (a que se pretende libertar/destruir)

Saída: a AVL T vazia (T a apontar para NULL)

PNodeAVL destruirAVL (PNodeAVL T)

```
{  
    if (T == NULL)  
        return T;  
    T→Esquerda = destruirAVL(T→Esquerda);  
    T→Direita = destruirAVL(T→Direita);  
    T = libertarNodoAVL(T);  
    return T;  
}
```

Verificar se uma Árvore AVL está vazia

- Adaptar a partir da implementada para Árvores Binárias de Pesquisa

Entrada: uma AVL T

Saída: 1 (se T está vazia) ou 0 (se T não está vazia)

```
int AVLVazia (PNodoAVL T)
```

```
{
```

```
    if (T == NULL)
```

```
        return 1;
```

```
    else
```

```
        return 0;
```

```
}
```

Mostrar os elementos numa Árvore AVL

- Adaptar a partir da implementada para Árvores Binárias de Pesquisa
- Operação para fazer a travessia em **em-ordem** (esquerda, raiz e direita)

Entrada: uma AVL T

Saída: -- (mostra os elementos da árvores pela ordem indicada)

```
void mostrarEmOrdemAVL (PNodoAVL T)
```

```
{  
    if (T != NULL) {  
        mostrarEmOrdemAVL(T→Esquerda);  
        mostrarElementoAVL(T→Elemento);  
        mostrarEmOrdemAVL(T→Direita);  
    }  
}
```

Mostrar os elementos numa Árvore AVL

- Adaptar a partir da implementada para Árvores Binárias de Pesquisa
- Operação para fazer a travessia em **pré-ordem** (raiz, esquerda e direita)

Entrada: uma AVL T

Saída: -- (mostra os elementos da árvores pela ordem indicada)

```
void mostrarPreOrdemAVL (PNodoAVL T)
```

```
{  
    if (T != NULL) {  
        mostrarElementoAVL(T→Elemento);  
        mostrarPreOrdemAVL(T→Esquerda);  
        mostrarPreOrdemAVL(T→Direita);  
    }  
}
```

Mostrar os elementos duma Árvore AVL

- Adaptar a partir da implementada para Árvores Binárias de Pesquisa
- Operação para fazer a travessia em **pós-ordem** (esquerda, direita e raiz)

Entrada: uma AVL T

Saída: -- (mostra os elementos da árvores pela ordem indicada)

```
void mostrarPosOrdemAVL (PNodoAVL T)
```

```
{  
  if (T != NULL) {  
    mostrarPosOrdemAVL(T→Esquerda);  
    mostrarPosOrdemAVL(T→Direita);  
    mostrarElementoAVL(T→Elemento);  
  }  
}
```

Pesquisar um elemento numa Árvore AVL

- Adaptar a partir da implementada para Árvores Binárias de Pesquisa

Entrada: o elemento X a pesquisar e uma AVL T

Saída: ponteiro para o nodo com o elemento a pesquisar (existe); NULL (não existe)

PNodeAVL pesquisarAVL (DadosAVL X, PNodeAVL T)

```
{  
    if (T == NULL)  
        return NULL;  
    if (compararElementosAVL(X, T→Elemento) == 0)    // X = T→Elemento  
        return T;  
    if (compararElementosAVL(X, T→Elemento) == -1)  // X < T→Elemento  
        return pesquisarAVL(X, T→Esquerda);  
    else  
        return pesquisarAVL(X, T→Direita);  
}
```

Consultar um elemento numa Árvore AVL

- Adaptar a partir da implementada para Árvores Binárias de Pesquisa

Entrada: o elemento a consultar X (a chave) e uma AVL T

Saída: -- (mostra o elemento da árvore T com chave igual à do elemento X)

```
void consultarAVL (DadosAVL X, PNodeAVL T)
```

```
{
```

```
    PNodeAVL P;
```

```
    P = pesquisarAVL(X, T);
```

```
    if (P != NULL)
```

```
        mostrarElementoAVL(P→Elemento);
```

```
    else
```

```
        printf("Nao existe elemento na AB com a mesma chave de X!\n");
```

```
}
```

Atualizar um elemento numa Árvore AVL

- Adaptar a partir da implementada para Árvores Binárias de Pesquisa

Entrada: a informação atual (chave) X, a nova informação Y e uma AVL T

Saída: ponteiro para a raiz da árvore T (pode ser diferente do inicial)

PNodeAVL atualizarAVL (DadosAVL X, DadosAVL Y, PNodeAVL T)

```
{  
    PNodeAVL P;  
    P = pesquisarAVL(X, T);  
    if (P != NULL) {  
        T = removerAVL(X, T);  
        T = inserirAVL(Y, T);  
    }  
    return T;  
}
```

Inserir um elemento numa árvore AVL

- A inserção de um nodo numa árvore AVL, consiste em
 - usar o mesmo mecanismo de inserção de um nodo numa ABP (uma AVL é uma ABP), em que aquele nodo é inserido como folha da árvore (com altura 0)
 - após a inserção do nodo na árvore, é necessário realizar duas operações:
 - 1º: atualizar a altura dos nodos já existentes na árvore
 - 2º: verificar a necessidade de equilibrar a árvore
- Atualização da altura dos nodos da árvore
 - basta, no pior caso, atualizar a altura dos nodos que definem o caminho da raiz da árvore ao nodo (folha) inserido
 - quando o nodo é inserido na subárvore de um nodo de menor altura, apenas é necessário atualizar as alturas dos nodos desta subárvore

Inserir um elemento numa árvore AVL

- Equilibrar uma árvore, através de operações de **rotações**
 - rotação simples à esquerda
 - rotação simples à direita
 - rotação dupla à esquerda
 - rotação dupla à direita

Inserir um elemento numa árvore AVL

- Operação de inserção
 - A operação de inserção de um elemento numa árvore AVL
 - é semelhante à apresentada para o caso de uma árvore binária de pesquisa
 - à qual se acrescentou a operação de reequilíbrio
 - Na travessia da árvore é usada a pesquisa binária recursiva para
 - determinar se o elemento a inserir **não existe** na árvore
 - marcando no nodo exterior (pai) de inserção, caso não exista
 - Depois de inserir um nodo com o elemento na árvore, que é uma folha,
 - no caminho de retorno desta folha até à raiz, a operação de reequilíbrio (função **equilibrarAVL**) é realizada de forma
 - a reequilibrar a árvore e
 - a atualizar as alturas dos nodos envolvidos, caso seja necessário

Inserir um elemento numa árvore AVL

Entrada: ponteiro para a árvore AVL e elemento a inserir (que não existe na AVL)
Saída: ponteiro da raiz da árvore AVL atualizado

PNodeAVL inserirAVL (PNodeAVL T, DadosAVL X)

```
{  
    if (T == NULL) {  
        T = criarNodeAVL(X);  
        return T;  
    }  
    if (compararElementosAVL(T→Elemento, X) == 1)    // T→Elemento > X  
        T→Esquerda = inserirAVL(T→Esquerda, X);  
    else    // T→Elemento < X, pois X não está na árvore  
        T→Direita = inserirAVL(T→Direita, X);  
    T→Altura = atualizarAlturaNode(T);  
    T = equilibrarAVL(T);    // reequilibrar a árvore  
    return T;  
}
```

Inserir um elemento numa árvore AVL

- Descrição: atualizar o campo “Altura” do nodo raiz de uma árvore AVL

Operação: atualizar altura da raiz de uma árvore

Entrada: ponteiro para a raiz da árvore (ou subárvore) AVL a atualizar altura

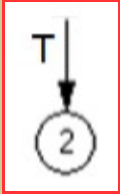
Saída: inteiro correspondente à altura atualizada da raiz da AVL

```
int atualizarAlturaNodo (PNodoAVL T)
```

```
{  
    int altEsq, altDir;  
    if (T == NULL)  
        return -1;  
    altEsq = T→Esquerda→Altura;  
    altDir = T→Direita→Altura;  
    if (altEsq > altDir)  
        return altEsq + 1;  
    else  
        return altDir + 1;  
}
```

Inserir um elemento numa árvore AVL

- Inserção de um **nodo** com o elemento **2** numa árvore AVL

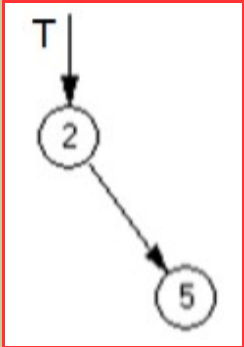


- atualização da altura dos nodos:
 - a **altura** do nodo com o **2** (**raiz**) é **0**

- a **árvore** está **equilibrada**

Inserir um elemento numa árvore AVL

- Inserção de um **nodo** com o elemento **5** na árvore AVL

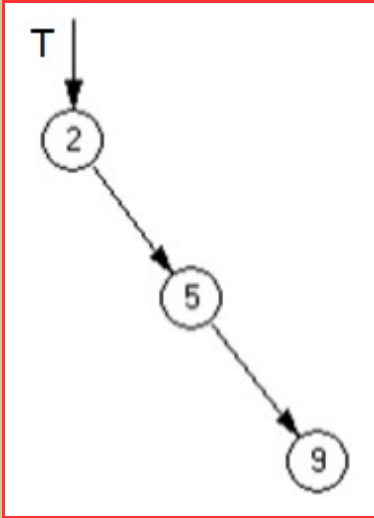


- atualização da altura dos nodos:
 - a **altura** do **nodo** com o **2** (**raiz**) passou de 0 para **1**
 - a **altura** do **nodo** com o **5** é **0**

- a **árvore** continua **equilibrada**

Inserir um elemento numa árvore AVL

- Inserção de um **nodo** com o elemento **9** na árvore AVL



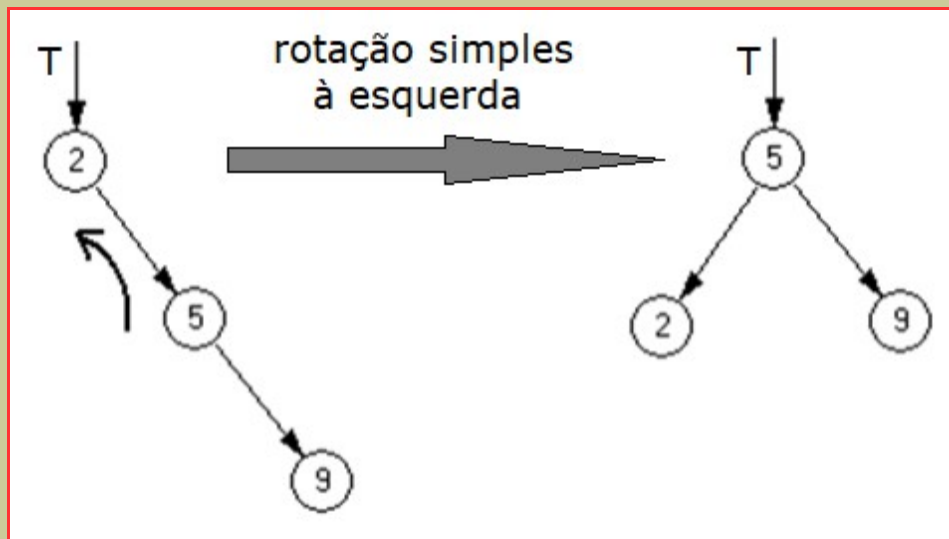
- atualização da altura dos nodos:

- a **altura** do nodo com o **2** (**raiz**) passou de 1 para **2**
- a **altura do nodo com o 5** passou de 0 para **1**
- a **altura** do nodo com o **9** é **0**

- a **árvore** fica **desequilibrada** no **nodo** com o **2** (**raiz**), pois a diferença de alturas entre as suas subárvores é 2:
 - a altura da sua subárvore esquerda (vazia) é -1
 - a altura da sua subárvore direita (com raiz em 5) é 1

Inserir um elemento numa árvore AVL

- Reequilibrar a árvore através de uma rotação simples à esquerda
 - efetuar uma **rotação simples à esquerda** do **nodo** com o **2**, usando o seu **filho direito** (o **nodo** com o **5**)

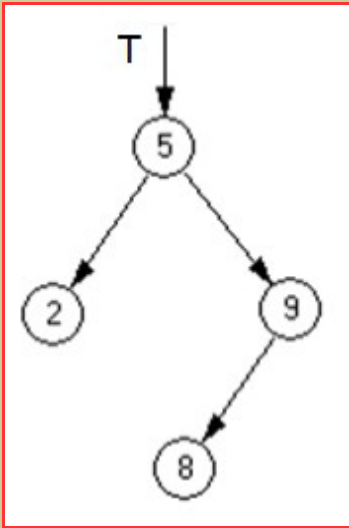


- atualização da altura dos nodos:
 - a **altura** do **nodo** com o **5** (agora **raiz**) mantém-se em **1**
 - a **altura** do **nodo** com o **2** passa de 2 para **0**
 - a **altura** do **nodo** com o **9** mantém o valor **0**

- o **nodo** com o **5** é agora a **raiz** da árvore (era o filho direito do nodo com o 2)
- a **árvore** ficou **equilibrada**

Inserir um elemento numa árvore AVL

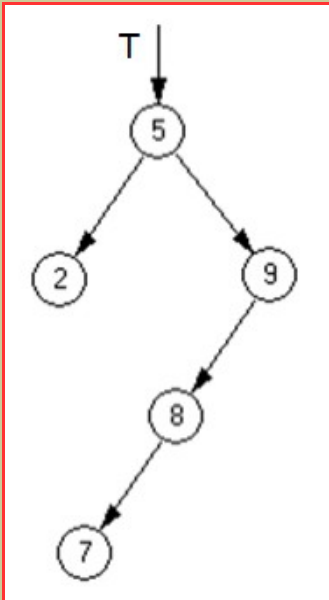
- Inserção de um nodo com o elemento 8 (com altura 0) na árvore AVL



- atualização da altura dos nodos:
 - a **altura** do **nodo** com o **5** (**raiz**) passa de 1 para **2**
 - a **altura** do **nodo** com o **9** passa de 0 para **1**
 - a **altura** do **nodo** com o **2** mantém-se com **0**
- a **árvore** mantém-se **equilibrada**

Inserir um elemento numa árvore AVL

- Inserção de um nodo com o elemento 7 (com altura 0) na árvore AVL



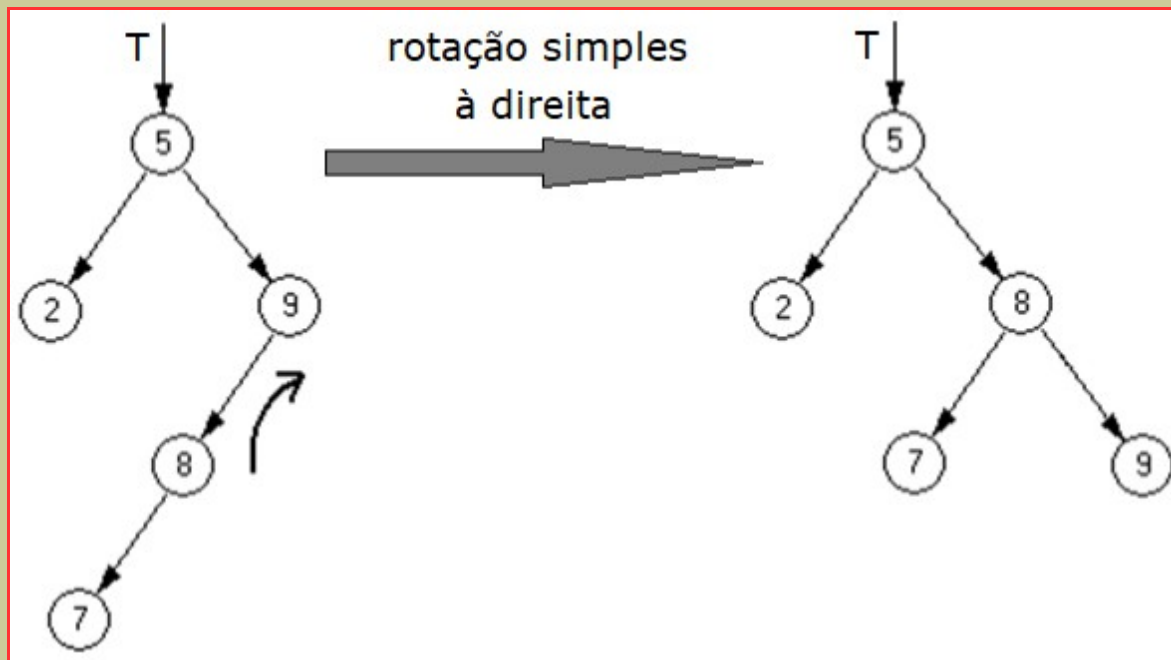
- atualização da altura dos nodos:

- a **altura** do **nodo** com o **5** (**raiz**) é atualizada de 2 para **3**
- a **altura** do **nodo** com o **9** é atualizada de 1 para **2**
- a **altura** do **nodo** com o **8** é atualizada de 0 para **1**
- a **altura** do **nodo** com o **2** mantém-se com **0**

- a **árvore** fica **desequilibrada** no **nodo** com o **9** (com **altura 2**)
 - a **altura** da sua **subárvore esquerda** (raiz no **nodo** com o **8**) é **1**
 - a altura da sua subárvore direita (vazia) é -1 (por definição)
- a **árvore** fica **desequilibrada** no **nodo** com o **5** (com **altura 3**)
 - a **altura** da sua **subárvore esquerda** (raiz no **nodo** com o **2**) é **0**
 - a **altura** da sua **subárvore direita** (raiz no **nodo** com o **9**) é **2**

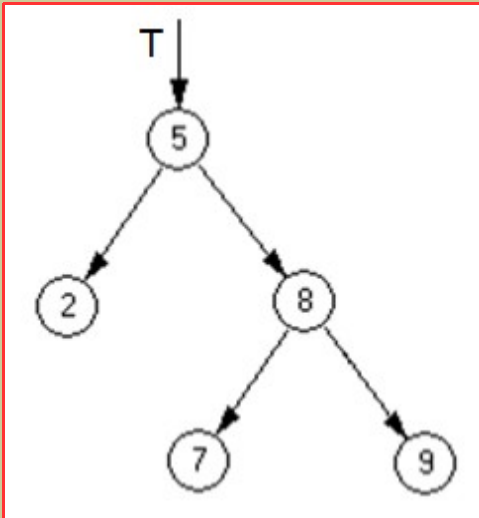
Inserir um elemento numa árvore AVL

- Reequilibrar a árvore através de uma rotação simples à direita
 - efetuar uma **rotação simples à direita** do **nodo** com o **9**, usando o seu **filho esquerdo** (o **nodo** com o **8**)
 - o **nodo** com o **8** passa a ser a **raiz** desta subárvore (com 7, 8 e 9)



Inserir um elemento numa árvore AVL

- Após esta rotação



- atualização da altura dos nodos:

- a **altura** do **nodo** com o **5** passa de 3 para **2**
- a **altura** do **nodo** com o **8** mantém-se em **1**
- a **altura** do **nodo** com o **9** passa de 2 para **0**
- a **altura** do **nodo** com o **7** mantém-se com **0**
- a **altura** do **nodo** com o **2** mantém-se com **0**

- a **árvore** fica **equilibrada** no **nodo** com o **8** (com **altura 1**)

- a **altura** da **subárvore direita** (raiz no **nodo** com o **9**) é **0**

- a **altura** da **subárvore esquerda** (raiz no **nodo** com o **7**) é **0**

- a **árvore** torna-se automaticamente **equilibrada** no **nodo** com o **5** (com **altura 2**)

- a **altura** da sua **subárvore direita** (com raiz no **nodo** com o **8**) é **1**

- a **altura** da **subárvore esquerda** (raiz no **nodo** com o **2**) é **0**

Inserir um elemento numa árvore AVL

- Há necessidade de aplicar uma operação de **rotação simples**, quando se está perante uma **inserção externa**
 - acontece quando o nodo é inserido
 - à esquerda do filho esquerdo, ou
 - à direita do filho direito
 - a árvore facilmente se torna equilibrada (o desequilíbrio acontece após esta inserção), aplicando a um único nodo uma única rotação, respetivamente
 - à direita, ou
 - à esquerda

Inserir um elemento numa árvore AVL

Operação: rotação simples à esquerda

Entrada: ponteiro para o nodo da árvore AVL onde se deteta o desequilíbrio

Saída: ponteiro anterior atualizado

PNodeAVL rotacaoSimplesEsquerda (PNodeAVL P) {

int altEsq, altDir;

PNodeAVL nodoAux = P→Direita;

P→Direita = nodoAux→Esquerda;

nodoAux→Esquerda = P;

// atualizar a altura do nodo P (um dos envolvidos na rotação)

altEsq = P→Esquerda→Altura;

altDir = P→Direita→Altura;

if (altEsq > altDir)

 P→Altura = altEsq + 1;

else

 P→Altura = altDir + 1;

...

Inserir um elemento numa árvore AVL

Operação: rotação simples à esquerda (cont.)

...

// atualizar a altura do nodo nodoAux (um dos envolvidos na rotação)

altEsq = nodoAux→Esquerda→Altura; // nodoAux→Esquerda = P

altDir = nodoAux→Direita→Altura;

if (altEsq > altDir)

 nodoAux→Altura = altEsq + 1;

else

 nodoAux→Altura = altDir + 1;

return nodoAux;

}

Inserir um elemento numa árvore AVL

Operação: rotação simples à direita

Entrada: ponteiro para o nodo da árvore AVL onde se deteta o desequilíbrio

Saída: ponteiro anterior atualizado

PNodeAVL rotacaoSimplesDireita (PNodeAVL P) {

int altEsq, altDir;

PNodeAVL nodoAux = P→Esquerda;

P→Esquerda = nodoAux→Direita;

nodoAux→Direita = P;

// atualizar a altura do nodo P (um dos envolvidos na rotação)

altEsq = P→Esquerda→Altura;

altDir = P→Direita→Altura;

if (altEsq > altDir)

 P→Altura = altEsq + 1;

else

 P→Altura = altDir + 1;

...

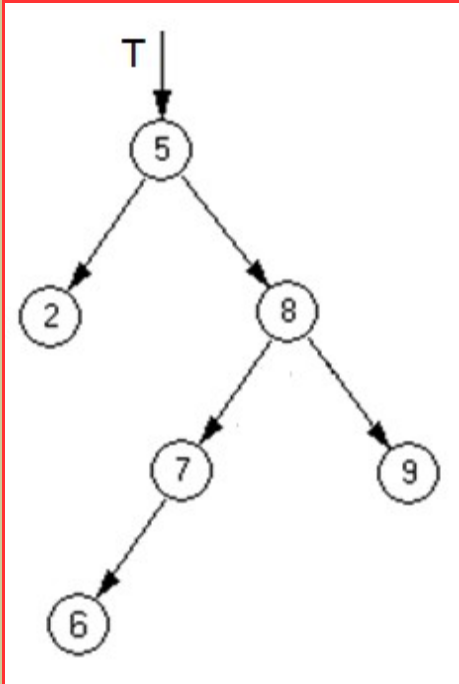
Inserir um elemento numa árvore AVL

Operação: rotação simples à direita (cont.)

```
...  
// atualizar a altura do nodo nodoAux (um dos envolvidos na rotação)  
altEsq = nodoAux→Esquerda→Altura;  
altDir = nodoAux→Direita→Altura; // nodoAux→Direita = P  
if (altEsq > altDir)  
    nodoAux→Altura = altEsq + 1;  
else  
    nodoAux→Altura = altDir + 1;  
return nodoAux;  
}
```

Inserir um elemento numa árvore AVL

- Inserir um nodo com o elemento 6 numa árvore AVL



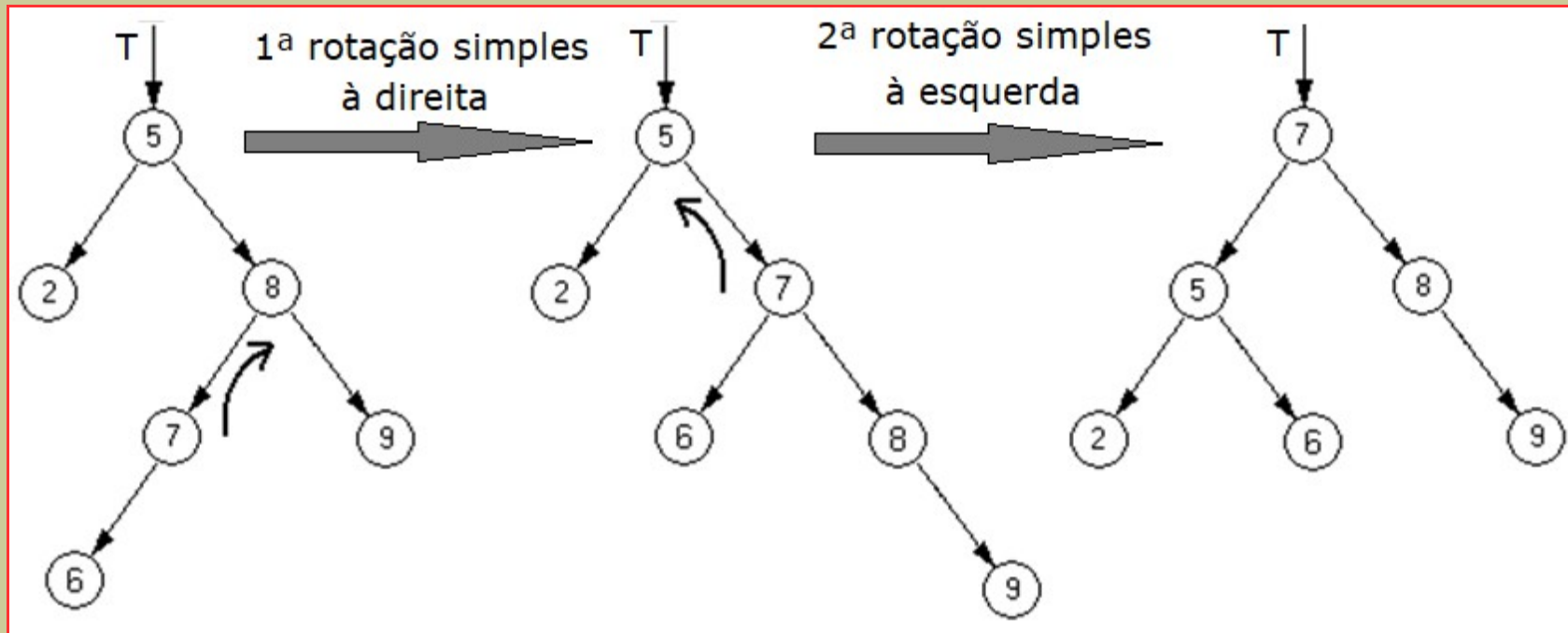
- atualização da altura dos nodos:

- a **altura** do **nodo** com o **5** (**raiz**) é atualizada de 2 para **3**
- a **altura** do **nodo** com o **8** é atualizada de 1 para **2**
- a **altura** do **nodo** com o **7** é atualizada de 0 para **1**
- a **altura** do **nodo** com o **9** mantém-se com **0**
- a **altura** do **nodo** com o **2** mantém-se com **0**

- a **árvore** fica **desequilibrada** no **nodo** com o **5** (com **altura 3**)
 - a **altura** da sua **subárvore esquerda** (raiz no **nodo** com o **2**) é **0**
 - a **altura** da sua **subárvore direita** (raiz no **nodo** com o **8**) é **2**

Inserir um elemento numa árvore AVL

- Reequilibrar a árvore através de uma **rotação dupla à esquerda**



- uma **rotação dupla à esquerda** consiste em efetuar duas rotações simples seguidas usando o mesmo nodo (o nodo 7)
 - rotação simples à **direita** do nodo 8 (filho direito de 5) usando o nodo 7, seguida
 - rotação simples à **esquerda** do nodo 5 usando o nodo 7

Inserir um elemento numa árvore AVL

Operação: rotação dupla à esquerda

Entrada: ponteiro para o nodo da árvore AVL onde se deteta o desequilíbrio

Saída: ponteiro anterior atualizado

PNodeAVL rotacaoDuplaEsquerda (PNodeAVL P)

{

 P→Direita = **rotacaoSimplesDireita**(P→Direita);

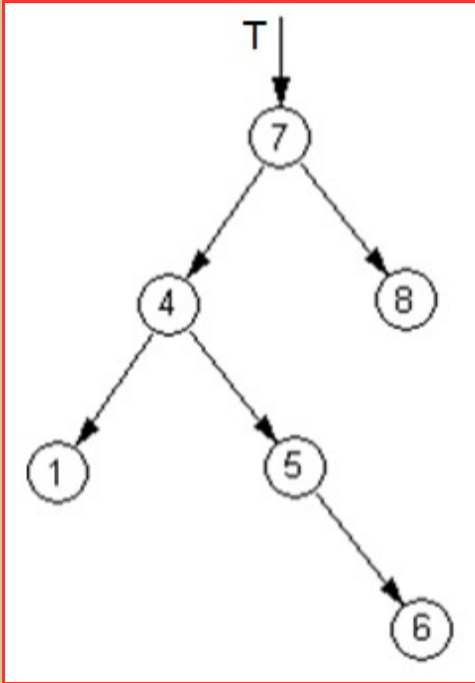
 P = **rotacaoSimplesEsquerda**(P);

return P;

}

Inserir um elemento numa árvore AVL

- Inserir um nodo com o elemento 6 numa árvore AVL



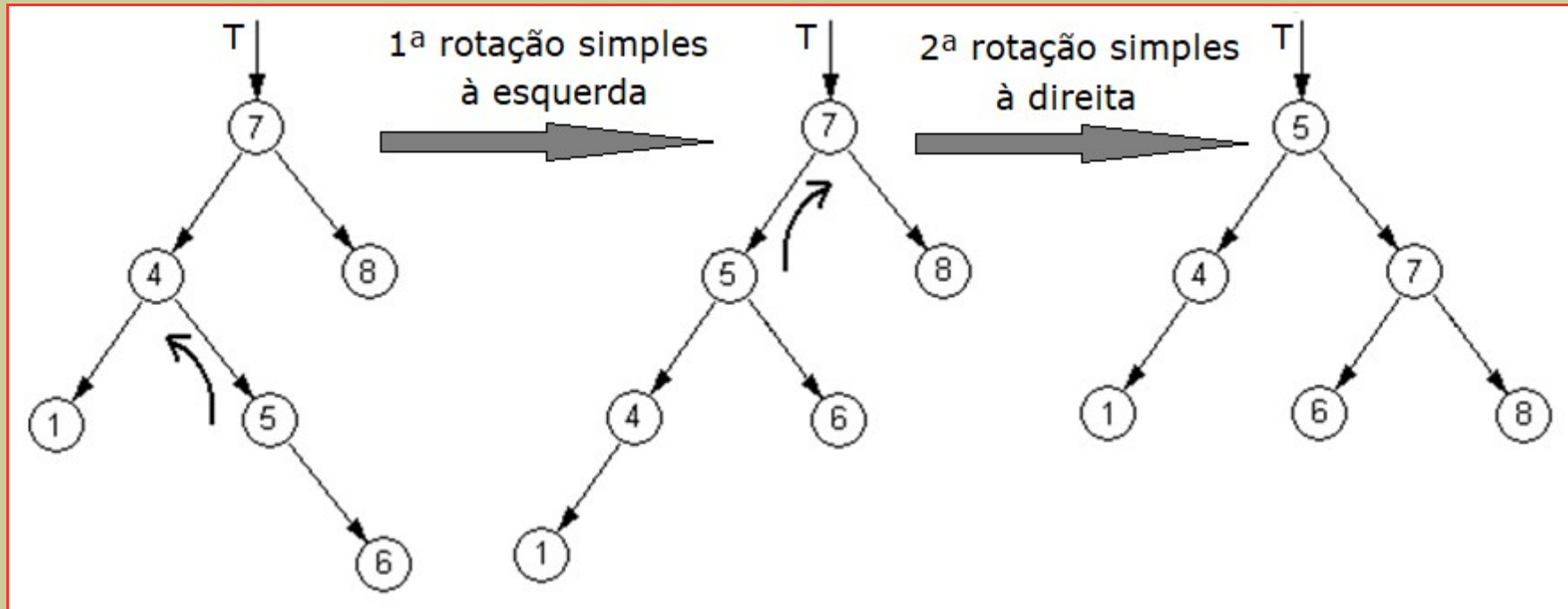
- atualização da altura dos nodos:

- a **altura** do **nodo** com o **7** (raiz) é atualizada de 2 para **3**
- a **altura** do **nodo** com o **4** é atualizada de 1 para **2**
- a **altura** do **nodo** com o **5** é atualizada de 0 para **1**
- a **altura** do **nodo** com o **6** mantém-se com **0**
- a **altura** do **nodo** com o **8** mantém-se com **0**

- a **árvore** fica **desequilibrada** no **nodo** com o **7** (com **altura 3**)
 - a **altura** da sua **subárvore esquerda** (raiz no **nodo** com o **8**) é **0**
 - a **altura** da sua **subárvore direita** (raiz no **nodo** com o **4**) é **2**

Inserir um elemento numa árvore AVL

- Reequilibrar a árvore através de uma **rotação dupla à direita**



- uma **rotação dupla à direita** consiste em efetuar duas rotações simples seguidas usando o mesmo nodo (o nodo 5)
 - rotação simples à esquerda do nodo 4 (filho esquerdo de 7) usando o nodo 5,
 - rotação simples à direita do nodo 7 usando o nodo 5

Inserir um elemento numa árvore AVL

Operação: rotação dupla à direita

Entrada: ponteiro para o nodo da árvore AVL onde se deteta o desequilíbrio

Saída: ponteiro anterior atualizado

PNodeAVL rotacaoDuplaDireita (PNodeAVL P)

{

 P→Esquerda = **rotacaoSimplesEsquerda**(P→Esquerda);

 P = **rotacaoSimplesDireita**(P);

return P;

}

Inserir um elemento numa árvore AVL

- Há necessidade de aplicar uma operação de **rotação dupla**, quando se está perante uma **inserção interna**
 - acontece quando o nodo é inserido
 - à esquerda do filho direito, ou
 - à direita do filho esquerdo
 - a árvore facilmente se torna equilibrada (o desequilíbrio acontece após esta inserção), aplicando a dois nodos uma rotação simples a cada um deles seguidas, respetivamente
 - à direita seguida de à esquerda, ou
 - à esquerda seguida de à direita

Inserir um elemento numa árvore AVL

Operação: equilibrar ABP do tipo AVL

Entrada: ponteiro para o nodo da árvore onde se deteta desequilíbrio (subárvore)

Saída: ponteiro anterior atualizado (se realizou reequilíbrio)

PNodeAVL equilibrarAVL (PNodeAVL T) {

int altEsq, altDir;

if (T == NULL) **return** T;

altEsq = T→Esquerda→Altura;

altDir = T→Direita→Altura;

if (altEsq - altDir == 2) { // subárvore esquerda desequilibrada

altEsq = T→Esquerda→Esquerda→Altura;

altDir = T→Esquerda→Direita→Altura;

if (altEsq >= altDir)

T = **rotacaoSimplesDireita**(T);

else

T = **rotacaoDuplaDireita**(T);

}

...

Inserir um elemento numa árvore AVL

Operação: equilibrar ABP do tipo AVL (cont.)

...

else

if (altDir - altEsq == 2) { // subárvore direita desequilibrada

altEsq = T→Direita→Esquerda→Altura;

altDir = T→Direita→Direita→Altura;

if (altDir >= altEsq)

T = **rotacaoSimplesEsquerda**(T);

else

T = **rotacaoDuplaEsquerda**(T);

}

return T;

}

Remover um elemento de uma árvore AVL

- A remoção de elementos numa árvore AVL provoca o efeito simétrico da inserção
- A **remoção interna** (remoção à direita do filho esquerdo ou à esquerda do filho direito) de um nodo
 - desequilibra a árvore da mesma forma que a inserção externa
 - é reequilibrada com uma rotação simples
- A **remoção externa** (remoção à esquerda do filho esquerdo ou à direita do filho direito) de um nodo
 - desequilibra a árvore da mesma forma que a inserção interna
 - é reequilibrada com uma rotação dupla

Remover um elemento de uma árvore AVL

- Este processo implica, para além da remoção do nodo,
 - a atualização das alturas dos nodos pertencentes ao caminho entre o nodo removido e a raiz da árvore, e
 - aplicar as rotações adequadas para reequilibrar a árvore (se necessário)
- Com exceção do reequilíbrio da árvore, este mecanismo é semelhante ao usado para a remoção de um elemento de uma ABP
 - na remoção do nodo, e
 - no ajuste das ligações
- O mecanismo de remoção de um nodo de uma AVL combina os mecanismos
 - de remoção de um elemento de uma árvore binária de pesquisa e
 - de equilíbrio de uma árvore AVL

Remover um elemento de uma árvore AVL

Entrada: uma árvore AVL e o elemento (existe na árvore) a remover
Saída: árvore AVL atualizada

```
PNodeAVL removerAVL (PNodeAVL T, DadosAVL X) {  
  if (compararElementosAVL(T→Elemento, X) == 0) { // T→Elemento = X  
    T = removerNodoAVL(T);  
    return T;  
  }  
  if (compararElementosAVL(T→Elemento, X) == 1) // T→Elemento > X  
    T→Esquerda = removerAVL(T→Esquerda, X);  
  else  
    T→Direita = removerAVL(T→Direita, X);  
  if (T != NULL) {  
    T→Altura = atualizarAlturaNodo(T);  
    T = equilibrarAVL(T); // reequilibrar a árvore  
  }  
  return T;  
}
```

Remover um elemento de uma árvore AVL

- Mecanismo muito semelhante ao usado em ABP
- Se o nodo a remover tem dois filhos, então é necessário substituir o seu elemento pelo menor da sua subárvore direita (**substituirNodoMinAVL**)

Remover um elemento de uma árvore AVL

Operação: remover um nodo de uma árvore AVL

Entrada: ponteiro para o pai do nodo da árvore AVL a remover

Saída: ponteiro para o pai do nodo removido atualizado

PNodeAVL removerNodoAVL (PNodeAVL T)

```
{  
  PNodeAVL nodoAux = T;  
  DadosAVL X;  
  if (T→Esquerda == NULL && T→Direita == NULL) // remover uma folha  
    T = libertarNodoAVL(T);  
  else  
    if (T→Esquerda == NULL) {      // só subárvore direita  
      T = T→Direita;  
      nodoAux = libertarNodoAVL(nodoAux);  
    }  
    else  
      ...  
}
```

Remover um elemento de uma árvore AVL

Operação: remover um nodo de uma árvore AVL (cont.)

```
...  
    if (T→Direita == NULL) { // só subárvore esquerda  
        T = T→Esquerda;  
        nodoAux = destruirNodoAVL(nodoAux);  
    }  
    else {  
        // duas subárvores não vazias: substituir pelo menor elemento da direita  
        T→Direita = substituirNodoMinAVL(T→Direita, &X);  
        T→Elemento = X;  
    }  
  
    if (T != NULL) {  
        T→Altura = atualizarAlturaNodo(T);  
        T = equilibrarAVL(T); // reequilibrar a árvore  
    }  
  
    return T;  
}
```

Remover um elemento de uma árvore AVL

- O mecanismo de substituir um elemento pelo menor elemento da subárvore direita do nodo com aquele elemento, consiste em
 - pesquisar recursivamente o nodo mais à esquerda das subárvores direitas desde aquele nodo
 - copiar o elemento do nodo substituto para o elemento do nodo a remover
 - ligar a subárvore direita do nodo substituto à subárvore esquerda do nodo anterior, e
 - destruir o nodo substituto
- Como esta operação pode desequilibrar a árvore, é usada no fim a operação de reequilíbrio de uma árvore AVL

Remover um elemento de uma árvore AVL

Entrada: ponteiro para o raiz da árvore AVL

Saída: ponteiro para o nodo pai do nodo substituto e o menor elemento (substituto)

PNodeAVL substituirNodoMinAVL (PNodeAVL T, DadosAVL *X) {

PNodeAVL nodoAux = T;

if (T→Esquerda == NULL) {

 *X = T→Elemento;

 T = T→Direita;

 nodoAux = **libertarNodoAVL**(nodoAux);

}

else

 T→Esquerda = **substituirNodoMinAVL**(T→Esquerda, X);

if (T != NULL) {

 T→Altura = **atualizarAlturaNodo**(T);

 T = **equilibrarAVL**(T);

}

return T;

}

Operação de reequilíbrio

- Consiste nos seguintes passos
 - calcular as alturas das subárvores do nodo onde poderá existir desequilíbrio
 - determinar se existe um desequilíbrio na árvore naquele nodo
 - se a árvore está desequilibrada, realizar a rotação adequada, a qual também atualiza a altura dos nodos envolvidos na rotação efetuada
 - se a árvore continua equilibrada, então é necessário atualizar a altura do nodo inspecionado
- Existem dois tipos de desequilíbrios num nodo numa árvore AVL
 - **externo:**
 - acontece após uma inserção externa ou uma remoção interna
 - sendo corrigido com uma rotação simples
 - **interno:**
 - acontece após uma inserção interna ou uma remoção externa
 - sendo corrigido com uma rotação dupla

Operação de reequilíbrio

- Se existe desequilíbrio na sua subárvore esquerda de um nodo, então
 - é necessário realizar uma rotação à direita
 - se a altura da subárvore esquerda é maior ou igual à altura da subárvore direita, do filho esquerdo deste nodo
 - então existe um desequilíbrio externo
 - senão existe um desequilíbrio interno
- Se o desequilíbrio ocorrer na subárvore direita daquele nodo, então
 - é necessário fazer uma rotação à esquerda
 - se a altura da subárvore direita é maior ou igual à altura da subárvore esquerda, do filho direito do nodo
 - então existe um desequilíbrio externo
 - senão existe um desequilíbrio interno
- Se não existe qualquer desequilíbrio no nodo, então
 - apenas será atualizado a sua altura