Introdução a AVL

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

Baseado nos slides do Prof. Rudinei Goularte

Conteúdo

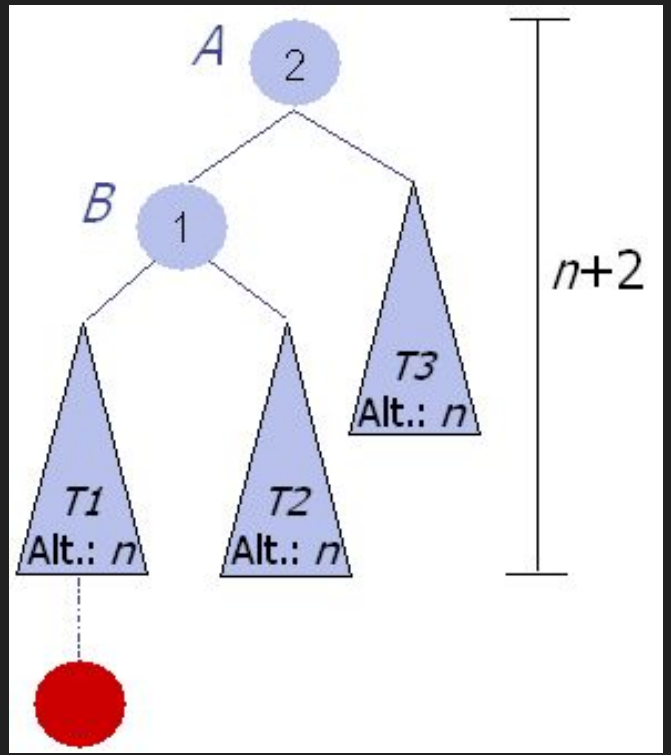
- ➔ Revisão de Rotação Direita e Esquerda
- ➔ Rotações Simples
- ➔ Rotações Duplas
- ➔ Qual Rotação Usar
- ➔ Implementação
- ➔ Inserção em Árvores AVL
- ➔ Remoção em Árvores AVL

Visualização de uma AVL

Árvore AVL - Rotação Direita

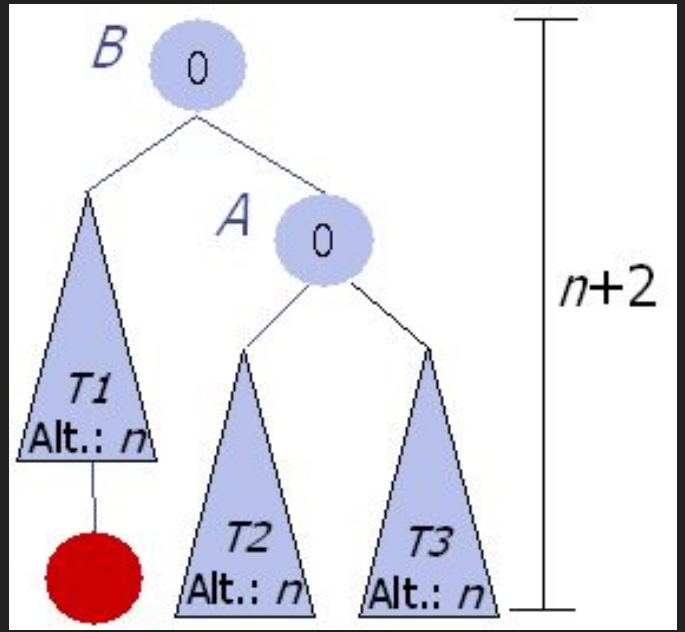
Árvores AVL - Rotação Direita

- A rotação direita tem formato geral ilustrado à direita
- T1, T2 e T3 podem ser sub-árvores de qualquer tamanho, inclusive 0
- A é o nó mais jovem a se tornar desbalanceado



Árvores AVL - Rotação Direita

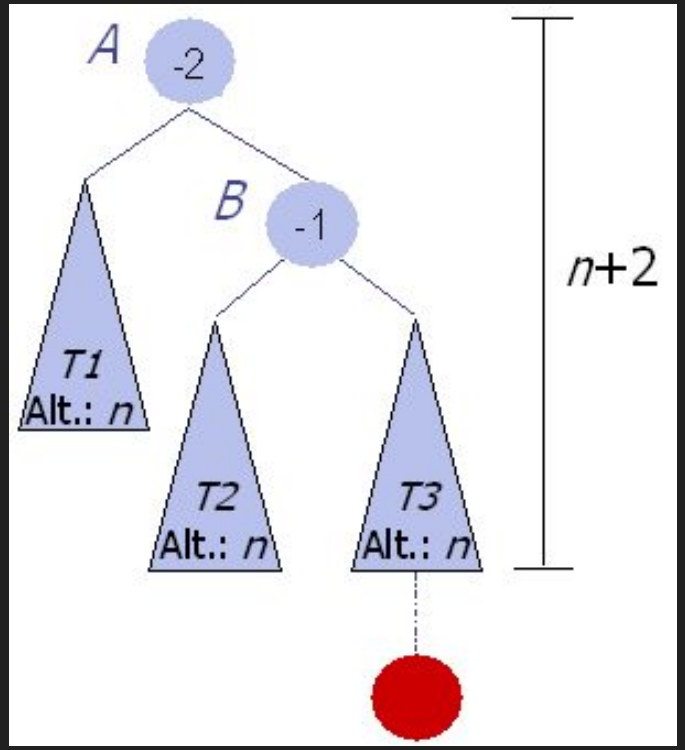
- A rotação direita tem formato geral ilustrado à direita
- T1, T2 e T3 podem ser sub-árvores de qualquer tamanho, inclusive 0
- A é o nó mais jovem a se tornar desbalanceado



Árvore AVL - Rotação Esquerda

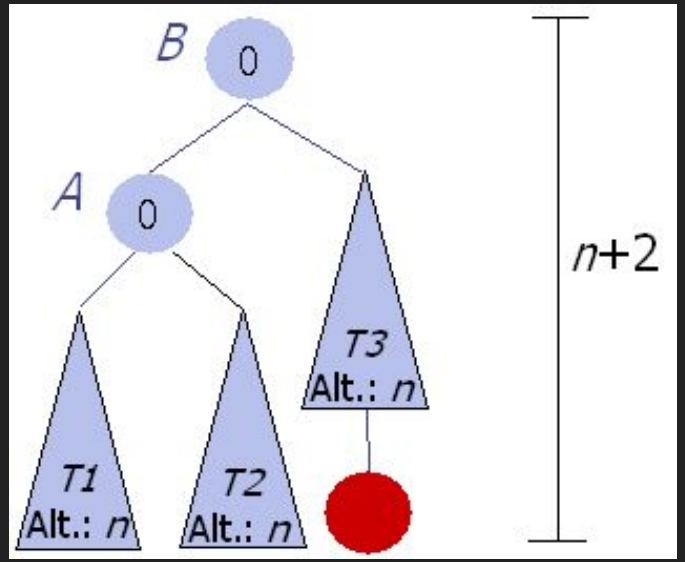
Árvores AVL - Rotação Esquerda

- A rotação esquerda tem formato geral ilustrado à direita
- T1, T2 e T3 podem ser sub-árvores de qualquer tamanho, inclusive 0
- A é o nó mais jovem a se tornar desbalanceado



Árvores AVL - Rotação Esquerda

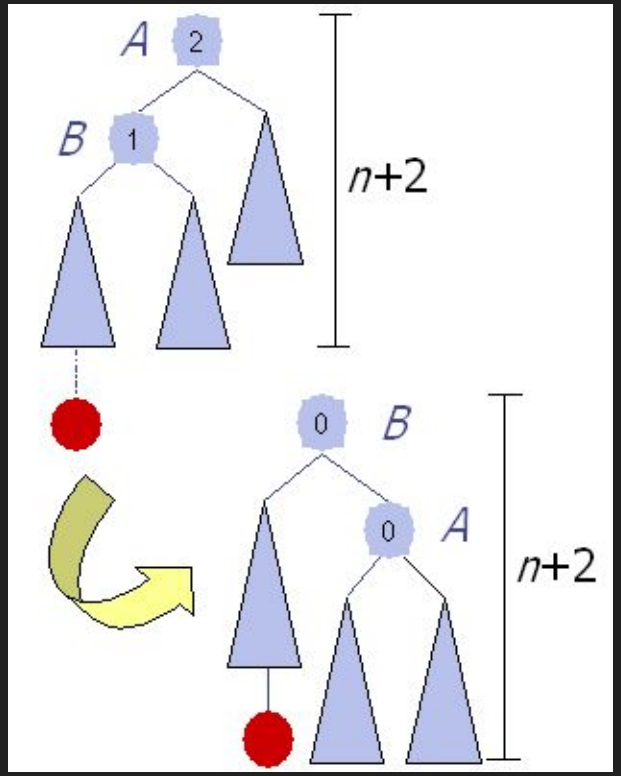
- A rotação esquerda tem formato geral ilustrado à direita
- T1, T2 e T3 podem ser sub-árvores de qualquer tamanho, inclusive 0
- A é o nó mais jovem a se tornar desbalanceado



Árvore AVL - Rotações Simples

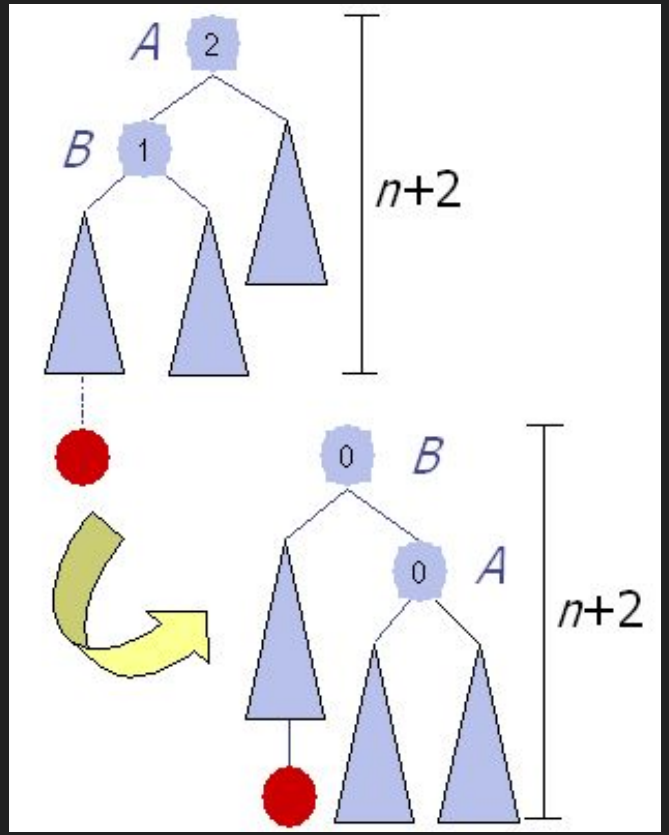
Rotações Simples

- Tanto para a rotação direita quanto para a rotação esquerda, a sub-árvore resultante tem como altura a mesma altura a sub-árvore original
- Isso significa que o fator de balanceamento de nenhum nó acima de A é afetado



Rotações Simples

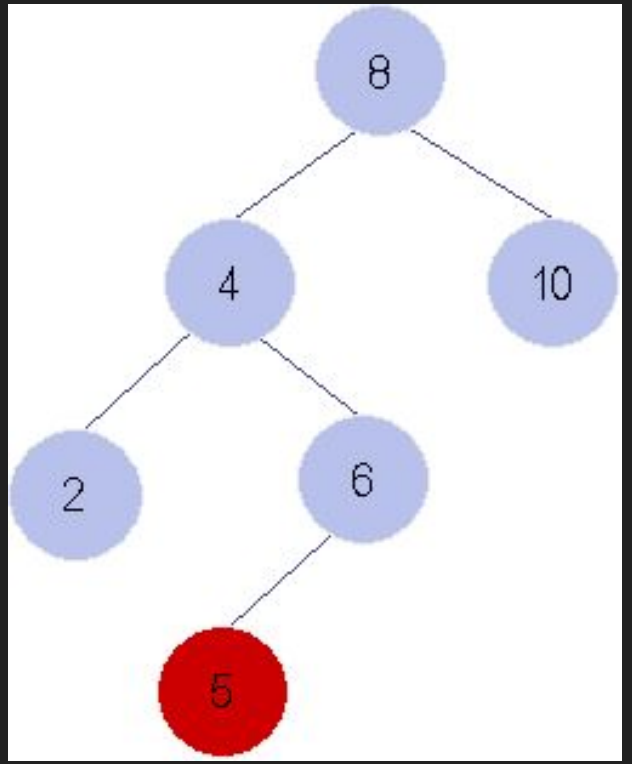
- Quando se deve utilizar a rotação direita ou esquerda?
- Quando o fator de balanceamento do nó A é positivo, a rotação é direita. Se for negativo a rotação é esquerda



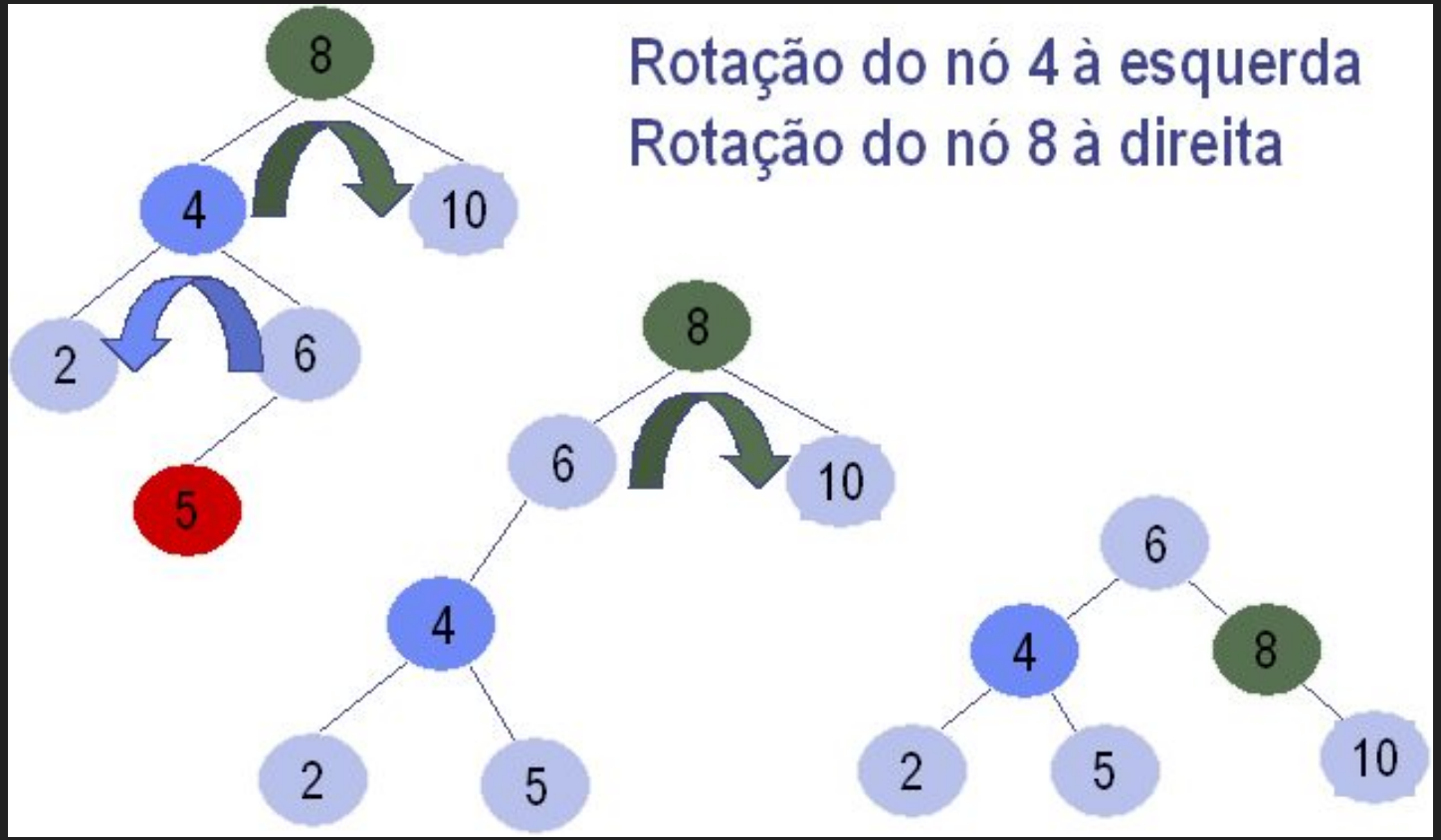
Árvore AVL - Rotação Duplas

Rotações Duplas

- Será que as rotações simples solucionam todos os tipos de desbalanceamento?
- Infelizmente, não
- Existem situações nas quais é necessário uma rotação dupla



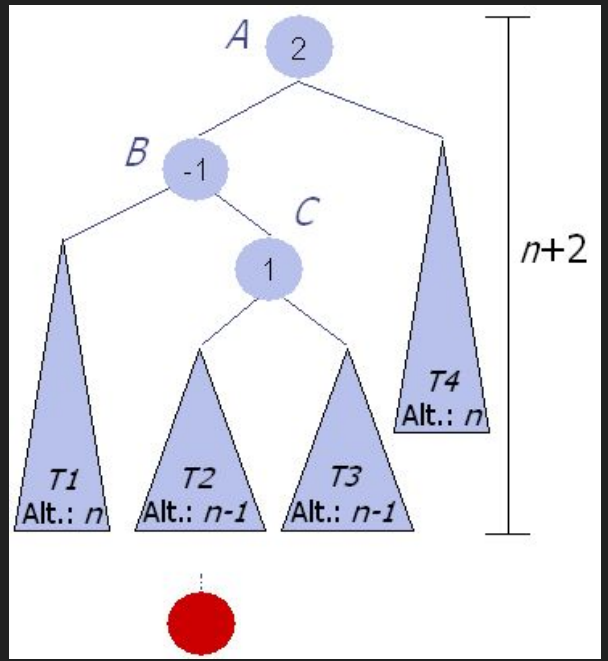
Rotações Duplas



Árvore AVL - Rotação Esquerda/Direita

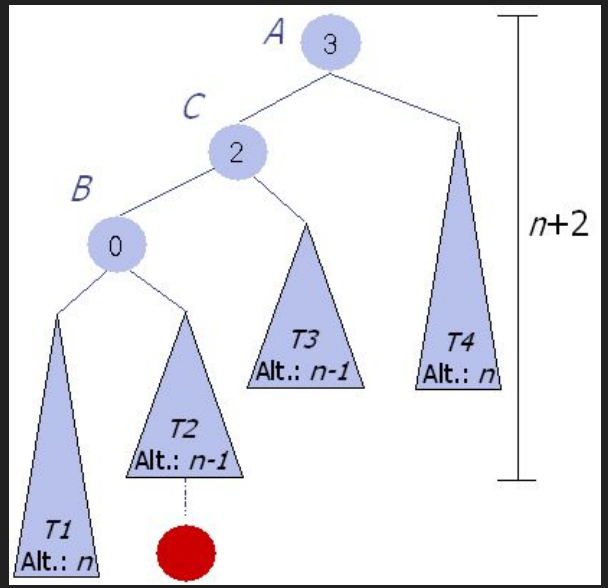
Rotação Esq./Dir.

- A rotação dupla esquerda/direita tem formato geral ilustrado à direita
- T1, T2, T3 e T4 podem ser sub-árvores de qualquer tamanho, inclusive 0
- A é o nó mais jovem a se tornar desbalanceado



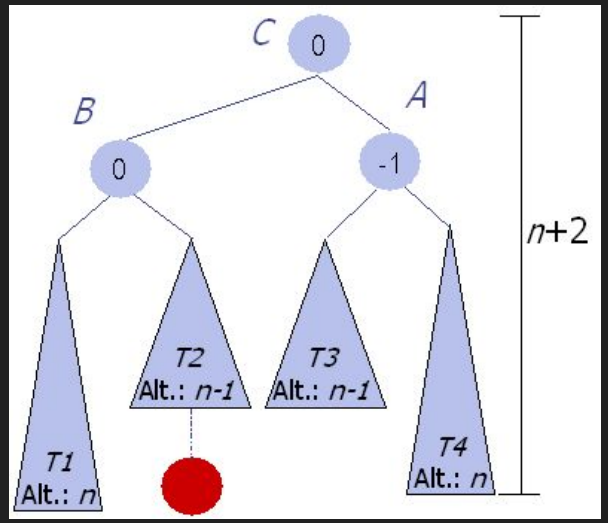
Rotação Esq./Dir.

- Passo 1: rotação esquerda em B
- A princípio a rotação esquerda parece deixar a árvore ainda mais desbalanceada
- Entretanto...



Rotação Esq./Dir.

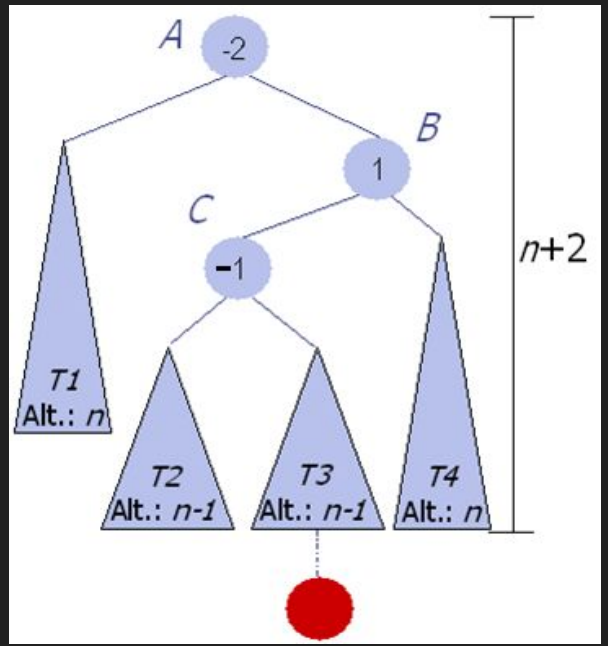
- Passo 2: rotação direita em A
- Repare que a altura final da sub-árvore é n + 2
- Funciona também se o novo nó tivesse sido inserido em T3



Árvore AVL - Rotação Direita/Esquerda

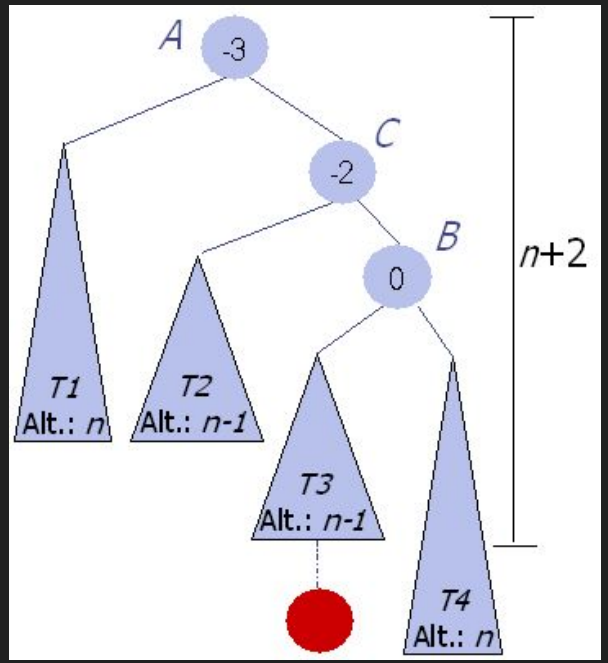
Rotação Dir./Esq.

- A rotação dupla direita/esquerda tem formato geral ilustrado à direita
- T1, T2, T3 e T4 podem ser sub-árvores de qualquer tamanho, inclusive 0
- A é o nó mais jovem a se tornar desbalanceado



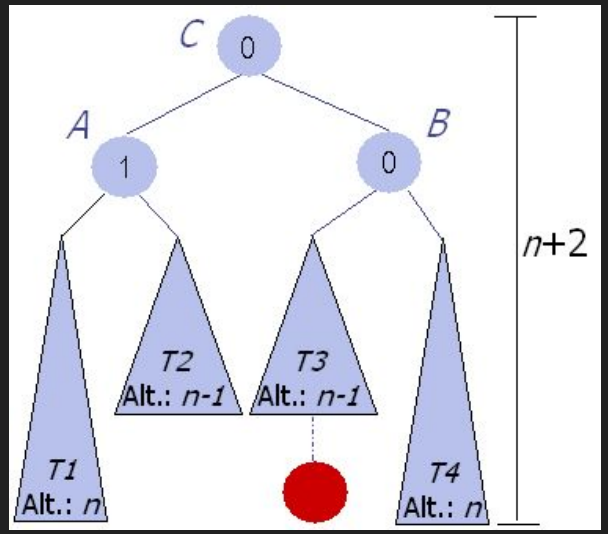
Rotação Dir./Esq.

- Passo 1: rotação direita em B
- A princípio a rotação direita parece deixar a árvore ainda mais desbalanceada
- Entretanto...



Rotação Dir./Esq.

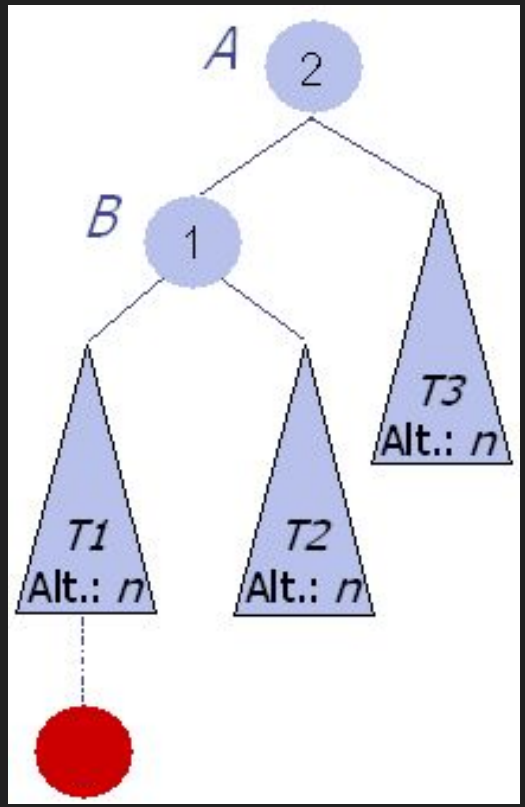
- Passo 2: rotação esquerda em A
- Repare que a altura final da sub-árvore é n + 2
- Funciona também se o novo nó tivesse sido inserido em T2



Qual Rotação Usar?

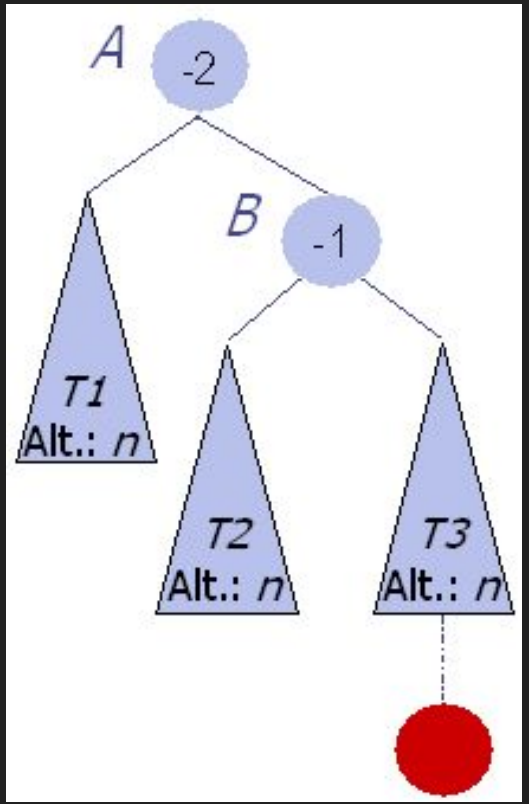
Qual Rotação Usar?

- Se o **sinal** do nó A e do nó B forem **iguais** então a **rotação** é **simples**
- Se o **fator de balanceamento nó A** (nó mais jovem a se tornar desbalanceado) for **positivo**, então a **rotação** é **direita**



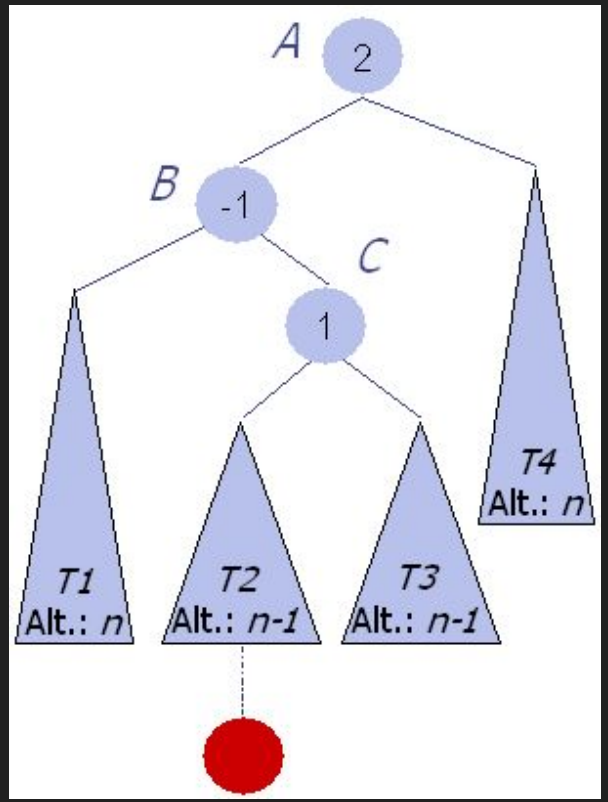
Qual Rotação Usar?

- Se o **sinal** do nó A e do nó B forem **iguais** então a **rotação** é **simples**
- Se o **fator de balanceamento nó A** (nó mais jovem a se tornar desbalanceado) for **negativo**, então a **rotação** é **esquerda**



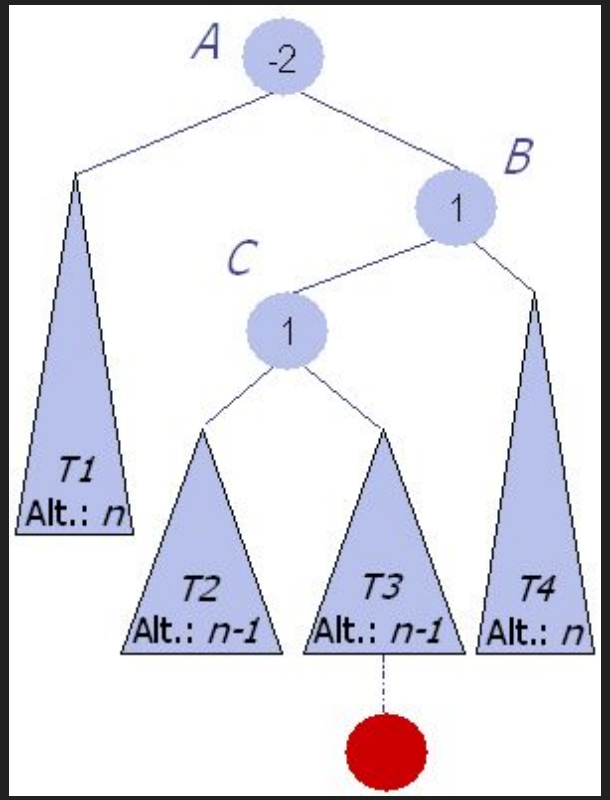
Qual Rotação Usar?

- Se o **sinal** do nó A e do nó B forem **diferentes** então a **rotação** é **dupla**
- Se o **fator de balanceamento nó A** (nó mais jovem a se tornar desbalanceado) for **positivo**, então a **rotação** é **esquerda/direita**



Qual Rotação Usar?

- Se o **sinal** do nó A e do nó B forem **diferentes** então a **rotação** é **dupla**
- Se o **fator de balanceamento nó A** (nó mais jovem a se tornar desbalanceado) for **negativo**, então a **rotação** é **direita/esquerda**



Implementação

Definição de Tipos

```
(.h)
 #define max(a, b) ((a > b) ? a : b)
 typedef struct avl AVL;
```

Definição de Tipos

```
(.c)
#include “avl.h”
typedef struct no NO;
struct no {
    ITEM *item;
    struct NO *fesq;
    struct NO *fdir;
    int altura;
};
struct avl {
    NO *raiz;
    int profundidade; ...
};
```

Criar AVL

```
AVL *avl_criar(void) {
     AVL *arvore = (AVL *) malloc(sizeof (AVL));
     if (arvore != NULL) {
          arvore->raiz = NULL; arvore->profundidade = -1;
     }
     return arvore;
}
```

Apagar AVL

```
void avl_apagar_aux(NO **raiz) {
     if (*raiz != NULL) {
          apagar_avl_aux(&((*raiz)->fesq));
          apagar_avl_aux(&((*raiz)->fdir));
          apagar_item(&(*raiz)->item);
     free(raiz);
     }
}

void avl_apagar(AVL **arvore) {
     avl_apagar_aux(&(*arvore)->raiz);
     free(*arvore);
     *arvore = NULL;
}
```

Altura do nó

```
int avl_altura_no(NO* raiz) {
     if (raiz == NULL) {
          return -1;
     } else {
          return raiz->altura;
     }
}
```

Cria nó

```
NO *avl_cria_no(ITEM *item) {
     NO *no = (NO *) malloc(sizeof (NO));
     if (no != NULL) {
          no->altura = 0;
          no->fdir = NULL;
          no->fesq = NULL;
          no->item = item;
     }
     return no;
}
```

Rotação Direita

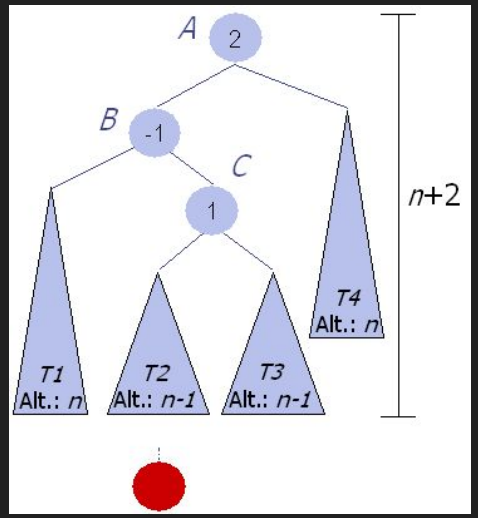
```
NO *rodar_direita(NO *a) {
    NO *b = a->fesq;
    a->fesq = b->fdir;
    b->fdir = a;

    a->altura = max(avl_altura_no(a->fesq), avl_altura_no(a->fdir)) + 1;
    b->altura = max(avl_altura_no(b->fesq), a->altura) + 1;
    return b;
}
```

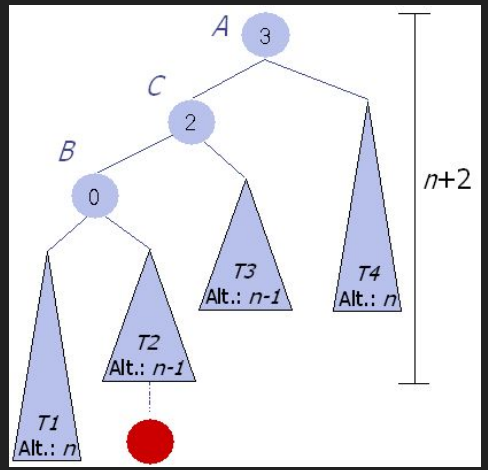
Rotação Esquerda

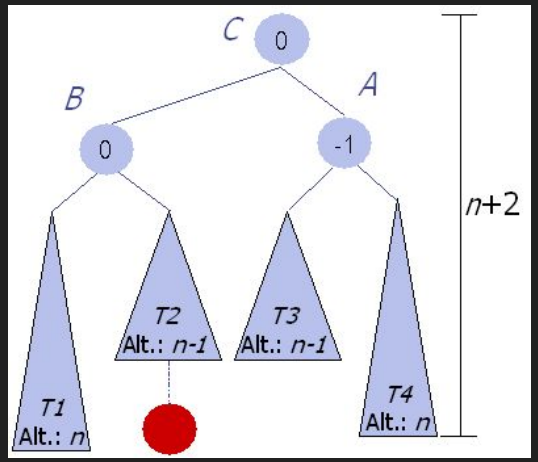
```
NO *rodar_esquerda(NO *a) {
    NO *b = a->fdir;
    a->fdir = b->fesq;
    b->fesq = a;

    a->altura = max(avl_altura_no(a->fesq), avl_altura_no(a->fdir)) + 1;
    b->altura = max(avl_altura_no(b->fdir), a->altura) + 1;
    return b;
}
```

Rotação Esquerda/Direita

```
NO *rodar_esquerda_direita(NO *a)   {
    a->fesq = rodar_esquerda(a->fesq);
    return rodar_direita(a);
}
```

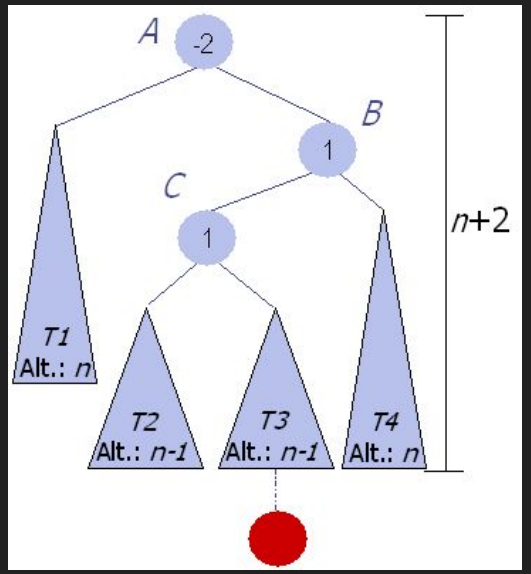


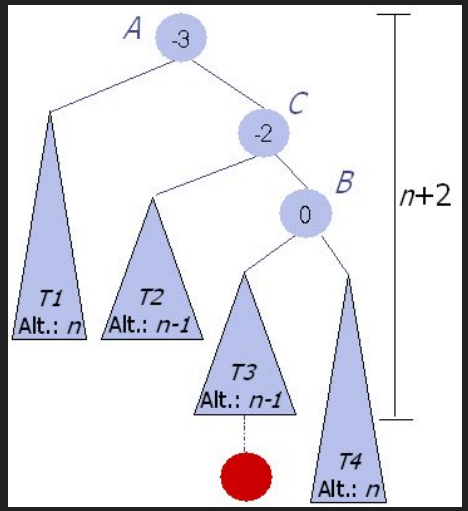


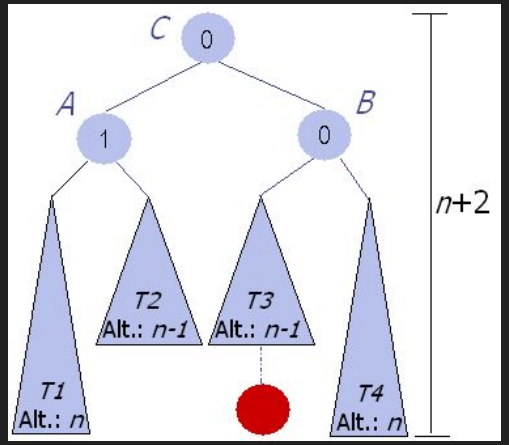


Rotação Esquerda/Direita

```
NO *rodar_direita_esquerda(NO *a)   {
    a->dir = rodar_direita(a->dir);
    return rodar_esquerda(a);
}
```







Algoritmo de Inserção

Algoritmo de Inserção

- Utilizando as rotinas de rotação pode-se definir um algoritmo de inserção em árvores AVL
- A maioria das implementações guardam o fator de balanceamento, porém guardar a altura dos nós facilita
- A inserção é feita em dois passos
  - O primeiro é uma inserção em ABBs; e
  - O segundo é o rebalanceamento, se necessário
- A 1ª etapa é definir uma inserção em ABB e atualizar as alturas dos nós

Algoritmo de Inserção (sem balanceamento)

```
NO *avl_inserir_no(NO *raiz, ITEM *item) {
     if (raiz == NULL) {
          raiz = avl_cria_no(item);
     } else if (item_chave(item) > item_chave(raiz->item)) {
          raiz->fdir = avl_inserir_no(raiz->fdir, item);
     } else if (item_chave(item->chave) < item_chave(raiz->item)) {
          raiz->fesq = avl_inserir_no(raiz->fesq, item);
     }
     raiz->altura = max(avl_altura_no(raiz->fesq),avl_altura_no(raiz->fdir)) + 1;
     return raiz;
}

boolean avl_inserir(AVL *arvore, ITEM *item) {
     return ((arvore->raiz = avl_inserir_no(arvore->raiz, item)) != NULL);
}
```

Algoritmo de Inserção

- ➔ Na volta da inserção o balanceamento é verificado, se a árvore estiver desbalanceada, aplicar as rotações necessárias
- ➔ O desbalanceamento é verificado com base na altura dos nós, o fator de balanceamento não precisa ser armazenado

Algoritmo de Inserção

- Se FB = -2 as rotações podem ser
  - Esquerda
  - Direita/Esquerda
- Se chave(novo) > chave(fdir)
  - Rotação esquerda
- Caso contrário
  - Rotação Direita/Esquerda
- FB = avl_altura_no(no->fesq) – avl_altura_no(no->fdir)

Algoritmo de Inserção

- Se FB = 2 as rotações podem ser
  - Direita
  - Esquerda/Direita
- Se chave(novo) < chave(fesq)
  - Rotação direita
- Caso contrário
  - Rotação Esquerda/Direita
- FB = avl_altura_no(no->fesq) – avl_altura_no(no->fdir)

Algoritmo de Inserção

- Se FB = 2 as rotações podem ser
  - Direita
  - Esquerda/Direita
- Se chave(novo) < chave(fesq)
  - Rotação direita
- Caso contrário
  - Rotação Esquerda/Direita
- FB = avl_altura_no(no->fesq) – avl_altura_no(no->fdir)

Algoritmo de Inserção (com balanceamento)

```
NO *avl_inserir_no(NO *raiz, ITEM *item) {
	if (raiz == NULL) {
		raiz = avl_cria_no(item);
	} else if (item_chave(item) > item_chave(raiz->item)) {
		raiz->fdir = avl_inserir_no(raiz->fdir, item);
	} else if (item_chave(item->chave) < item_chave(raiz->item)) {
		raiz->fesq = avl_inserir_no(raiz->fesq, item);
	}
	raiz->altura = max(avl_altura_no(raiz->fesq),avl_altura_no(raiz->fdir)) + 1;
	if (avl_altura_no(raiz->fesq) - avl_altura_no(raiz->fdir) == -2)
		if (item_chave(item) > item_chave(raiz->fdir->item))
			raiz = rodar_esquerda(raiz);
		else
			raiz = rodar_direita_esquerda(raiz);
	if (avl_altura_no(raiz->fesq) - avl_altura_no(raiz->fdir) == 2)
		if (item_chave(item->chave) < item_chave(raiz->fesq->item))
			raiz = rodar_direita(raiz);
		else
			raiz = rodar_esquerda_direita(raiz);
	return raiz;
}
```

Remoção em AVL

Algoritmo de Remoção

- Utilizando as rotinas de rotação pode-se definir um algoritmo de remoção em árvores AVL
- A remoção é feita em dois passos
  - O primeiro é uma remoção em ABBs
    - Existem 3 casos possíveis: o nó a ser removido possui grau 0, 1 ou 2.
  - O segundo é o rebalanceamento, se necessário
- O processo é semelhante à inserção

Complexidade das AVLs

Complexidade das AVLs

- A altura máxima de uma ABB AVL é 1,44 log2 n
  - Dessa forma, uma pesquisa nunca exige mais do que 44% mais comparações que uma ABB totalmente balanceada.
- Na prática, para n grande, os tempos de busca são por volta de $\log_2 n + 0{,}25$
- Na média, é necessária uma rotação em 46,5% das inserções

Exercícios

Exercícios

- Simule a inserção da seguinte seqüência de valores em uma árvore AVL: 10, 7, 20, 15, 17, 25, 30, 5, 1
- Em cada opção abaixo, insira as chaves na ordem mostrada de forma a construir uma arvore AVL. Se houver rebalanceamento de nós, mostre qual o procedimento a fazer
  - a, z, b, y, c, x
  - a, z, b, y, c, x, d, w, e, v, f
  - a, v, l, t, r, e, i, o, k
  - m, t, e, a, z, g, p

Exercícios

- ➔ Escreva uma função que retorna a altura da árvore AVL.
- ➔ Qual é a complexidade da operação implementada? Ela é mais eficiente que a implementação para ABBs?
- ➔ Implemente o TAD AVL com as operações de inserção e busca e demais operações auxiliares

Exercícios

- Mostre passo-a-passo a árvore AVL gerada pelas inserções das seguintes chaves na ordem fornecida
  - 10, 5, 20, 1, 3, 4, 8, 30, 40, 35, 50, 45, 55, 51, 100

- Para a AVL gerada, mostre passo-a-passo a remoção das chaves 51, 3, e 40

Referências

- Material baseado no originais produzidos pelo professor Rudinei Gularte
- SZWARCFITER, J. L.; MARKENZON, L. Estruturas de Dados e seus Algoritmos, Livros Técnicos e Científicos, 1994.
- TENEMBAUM, A.M., e outros Data Structures Using C, Prentice-Hall, 1990.
- ZIVIANI, N. Projeto de Algoritmos, Thomson, 2a. Edição, 2004.
- https://www.cs.usfca.edu/~galles/visualization/AVLtree.html