AED III
Árvores Binárias Balanceadas — AVL

Ciência da Computação – IFSULDEMINAS

Primeiro Semestre de 2014

Roteiro

Estrutura para Árvore AVL

Nó

```
Nó:
    chave valor armazenado no nó;
    fb fator de balanceamento do no nó;
    direita ponteiro para o filho da direita;
    esquerda ponteiro para o filho da esquerda;
    pai ponteiro para o pai;
```

Inserção em Árvore AVL

Inserção

- Insira o novo nó.
- Iniciando com o **nó pai** do nó recém-inserido, teste se a propriedade AVL foi violada neste nó:
  - Caso a condição AVL tenha sido violada: execute as operações de rotação conforme for o caso (Rotação Simples ou Dupla).
  - Caso contrário, a árvore está correta.

Inserção em Árvore AVL

Inserção

- Insira o novo nó.
- Iniciando com o **nó pai** do nó recém-inserido, teste se a propriedade AVL foi violada neste nó:
  - Caso a condição AVL tenha sido violada: execute as operações de rotação conforme for o caso (Rotação Simples ou Dupla).
  - Caso contrário, a árvore está correta.

Inserção em Árvore AVL

Inserção

- Insira o novo nó.
- Iniciando com o **nó pai** do nó recém-inserido, teste se a propriedade AVL foi violada neste nó:
  - Caso a condição AVL tenha sido violada: execute as operações de rotação conforme for o caso (Rotação Simples ou Dupla).
  - Caso contrário, a árvore está correta.

Inserção em Árvore AVL

Inserção

- Insira o novo nó.
- Iniciando com o **nó pai** do nó recém-inserido, teste se a propriedade AVL foi violada neste nó:
  - Caso a condição AVL tenha sido violada: execute as operações de rotação conforme for o caso (Rotação Simples ou Dupla).
  - Caso contrário, a árvore está correta.

Inserção em Árvore AVL

Inserção

```
INSERIR(T, k) {
    if T == NIL
        Alocar um novo nó com chave k
        T.raiz = novo;
    if k < T.chave
        INSERIR(T.esquerda, k);
        // Ajustar ponteiro para nó pai
        CALCULAR_FB(T);
        BALANCEAR(T);
    else
        INSERIR(T.direita, k);
        // Ajustar ponteiro para nó pai
        CALCULAR_FB(T);
        BALANCEAR(T);
}
```

Inserção em Árvore AVL

- Parâmetros da função:
  - **T:** a raiz da árvore
  - **k:** valor a ser inserido
- Usa as funções:
  - **CALCULAR_FB(T):** calcula o fator de balanceamento apenas de T. O fator de balanceamento é dado pela diferença de alturas.
  - **BALANCEAR(T):** verifica se a condição de AVL foi violada. Caso tenha sido, verifica o tipo de rotação a ser aplicada e executa as rotações.
  - **Observação:** ao rotacionar uma sub-árvore ajuste os fatores de balanceamento.

Inserção em Árvore AVL

- Parâmetros da função:
  - **T:** a raiz da árvore
  - **k:** valor a ser inserido
- Usa as funções:
  - **CALCULAR_FB(T):** calcula o fator de balanceamento apenas de T. O fator de balanceamento é dado pela diferença de alturas.
  - **BALANCEAR(T):** verifica se a condição de AVL foi violada. Caso tenha sido, verifica o tipo de rotação a ser aplicada e executa as rotações.
  - **Observação:** ao rotacionar uma sub-árvore ajuste os fatores de balanceamento.

Inserção em Árvore AVL

- Parâmetros da função:
  - **T:** a raiz da árvore
  - **k:** valor a ser inserido
- Usa as funções:
  - **CALCULAR_FB(T):** calcula o fator de balanceamento apenas de T. O fator de balanceamento é dado pela diferença de alturas.
  - **BALANCEAR(T):** verifica se a condição de AVL foi violada. Caso tenha sido, verifica o tipo de rotação a ser aplicada e executa as rotações.
  - **Observação:** ao rotacionar uma sub-árvore ajuste os fatores de balanceamento.

Inserção em Árvore AVL

- Parâmetros da função:
  - **T:** a raiz da árvore
  - **k:** valor a ser inserido
- Usa as funções:
  - **CALCULAR_FB(T):** calcula o fator de balanceamento apenas de T. O fator de balanceamento é dado pela diferença de alturas.
  - **BALANCEAR(T):** verifica se a condição de AVL foi violada. Caso tenha sido, verifica o tipo de rotação a ser aplicada e executa as rotações.
  - **Observação:** ao rotacionar uma sub-árvore ajuste os fatores de balanceamento.

Inserção em Árvore AVL

- Parâmetros da função:
  - **T:** a raiz da árvore
  - **k:** valor a ser inserido
- Usa as funções:
  - **CALCULAR_FB(T):** calcula o fator de balanceamento apenas de T. O fator de balanceamento é dado pela diferença de alturas.
  - **BALANCEAR(T):** verifica se a condição de AVL foi violada. Caso tenha sido, verifica o tipo de rotação a ser aplicada e executa as rotações.
  - **Observação:** ao rotacionar uma sub-árvore ajuste os fatores de balanceamento.

Inserção em Árvore AVL

- Parâmetros da função:
  - **T:** a raiz da árvore
  - **k:** valor a ser inserido
- Usa as funções:
  - **CALCULAR_FB(T):** calcula o fator de balanceamento apenas de T. O fator de balanceamento é dado pela diferença de alturas.
  - **BALANCEAR(T):** verifica se a condição de AVL foi violada. Caso tenha sido, verifica o tipo de rotação a ser aplicada e executa as rotações.
  - **Observação:** ao rotacionar uma sub-árvore ajuste os fatores de balanceamento.

Remoção em Árvore AVL

Remoção

- Remova o nó.
- Iniciando com o **nó pai** do nó removido, teste se a propriedade AVL foi violada neste nó:
  - Caso a condição AVL tenha sido violada: execute as operações de rotação conforme for o caso (Rotação Simples ou Dupla).
  - Caso contrário, a árvore está correta.

Remoção em Árvore AVL

Remoção

- Remova o nó.
- Iniciando com o **nó pai** do nó removido, teste se a propriedade AVL foi violada neste nó:
  - Caso a condição AVL tenha sido violada: execute as operações de rotação conforme for o caso (Rotação Simples ou Dupla).
  - Caso contrário, a árvore está correta.

Remoção em Árvore AVL

Remoção

- Remova o nó.
- Iniciando com o **nó pai** do nó removido, teste se a propriedade AVL foi violada neste nó:
  - Caso a condição AVL tenha sido violada: execute as operações de rotação conforme for o caso (Rotação Simples ou Dupla).
  - Caso contrário, a árvore está correta.

Remoção em Árvore AVL

Remoção

- Remova o nó.
- Iniciando com o **nó pai** do nó removido, teste se a propriedade AVL foi violada neste nó:
  - Caso a condição AVL tenha sido violada: execute as operações de rotação conforme for o caso (Rotação Simples ou Dupla).
  - Caso contrário, a árvore está correta.

Remoção em Árvore AVL

Remoção

```
REMOVER(T, k) {
    if T == NIL
        termine;
    else if T.chave == k
        Casos da remoção
    else if k < T.chave
        REMOVER(T.esquerda, k);
        CALCULAR_FB(T);
        BALANCEAR(T);
    else
        REMOVER(T.direita, k);
        CALCULAR_FB(T);
        BALANCEAR(T);
}
```

- Parâmetros da função:
  - T: a raiz da árvore
  - k: valor a ser removido

Remoção em Árvore AVL

Remoção

```
REMOVER(T, k) {
    if T == NIL
        termine;
    else if T.chave == k
        Casos da remoção
    else if k < T.chave
        REMOVER(T.esquerda, k);
        CALCULAR_FB(T);
        BALANCEAR(T);
    else
        REMOVER(T.direita, k);
        CALCULAR_FB(T);
        BALANCEAR(T);
}
```

- Parâmetros da função:
  - **T:** a raiz da árvore
  - **k:** valor a ser removido

Outra Proposta de Inserção em Árvore AVL

Inserção

```
int insereAVL(no **T, int x) {
  // variável booleana que indica se a altura da árvore cresceu
  int cresceu;
  if (*T == NULL) {
    *T = (no *) malloc(sizeof(no));
    (*T)->chave = x;
    (*T)->dir = (*T)->esq = NULL;
    (*T)->bal = 0;
    cresceu = 1;   // Esta sub arvore cresceu
  } else if ((*T)->chave > x) {
     // chama inserção para esquerda
  } else if ((*T)->chave < x) {
     // chama inserção para direita
  } else cresceu = 0;
  return cresceu;
}
```

Analisando Inserção à Esquerda

Inserção

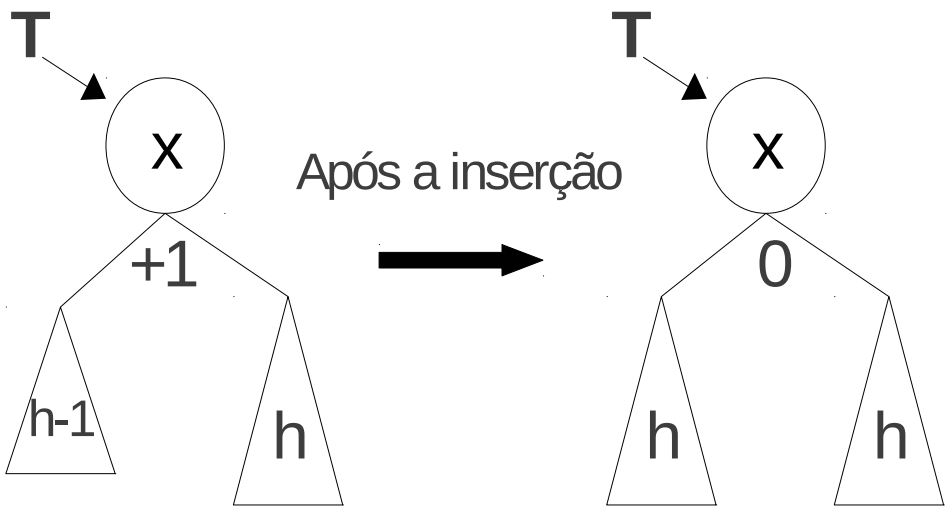
```
// Tenta inserir à esquerda e vê se a sub-árvore cresceu
cresceu = insereAVL(&(*T)->esq, x);
if (cresceu) {
   // Verifica o estado atual de balanceamento
   switch((*T)->bal) {
      case 1:
         ...
      case 0:
         ...
      case -1:
         ...
   }
}
```

Analisando Inserção à Esquerda

A sub-árvore da direita era maior, não há crescimento

Inserção

```
case 1:
    (*T)->bal = 0;
    cresceu = 0;
    break;
```

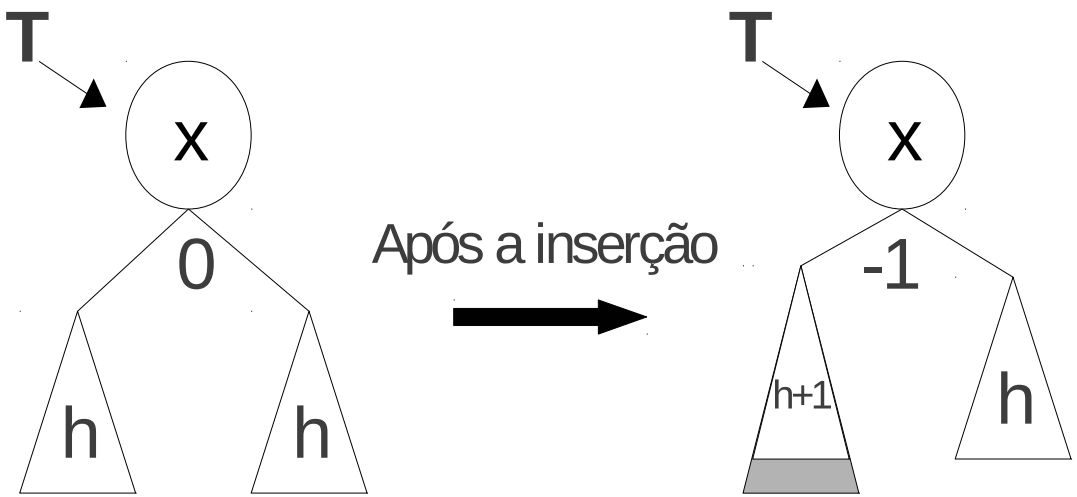


Analisando Inserção à Esquerda

A sub-árvore da direita tinha tamanho igual, houve crescimento

Inserção

```
case 0:
    (*T)->bal = -1;
    cresceu = 1;
    break;
```



Analisando Inserção à Esquerda

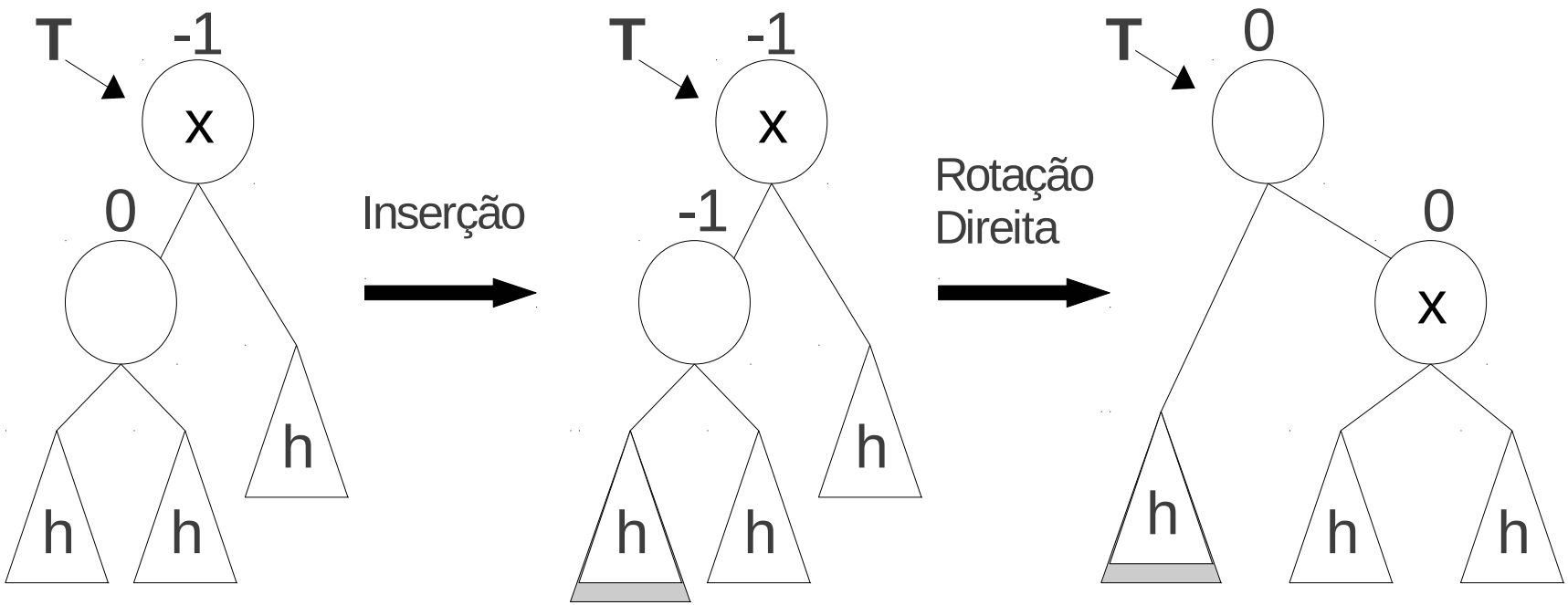
A sub-árvore da direita era menor, houve crescimento

Inserção

```
case -1:
   if ((*T)->esq->bal == -1) {   // FB filho esquerdo = -1
      rot_dir(T);
      (*T)->bal = (*T)->dir->bal = 0;
   } else {   // FB filho esquerdo = 0 ou 1
      rot_esq(&(*T)->esq);
      rot_dir(T);
      if ((*T)->bal == -1) {
         (*T)->esq->bal = 0;
         (*T)->dir->bal = 1;
      } else {
         (*T)->dir->bal = 0;
         (*T)->esq->bal = -(*T)->bal;
      }
      (*T)->bal = 0;
    }
```

Analisando Inserção à Esquerda

O fator de balanceamento do nó X é -1 e do seu filho à esquerda é -1.



Caso 0

Analisando Inserção à Esquerda

O fator de balanceamento do nó X é -1 e do seu filho à esquerda é 1.