UNILASALLE
Curso de Bacharelado em Ciência da Computação

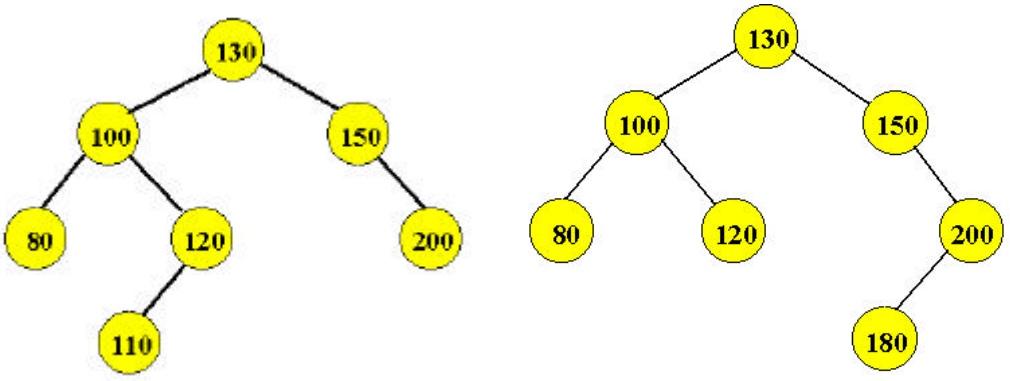
Estrutura de Dados II
Prof. Magalí T. Longhi

Árvores AVL*

* Material preparado para a disciplina de Estrutura de Dados II do Curso de Bacharelado em Ciência da Computação do UNILASALLE por Magalí T. Longhi e baseado nos livros: (a) Estrutura de Dados e seus Algoritmos de Jayme L. Szwarcfiter e Lilian Markenzon e (b) Tabelas: organização e Pesquisa de Clésio S. dos Santos e Paulo A. de Azeredo

1. Definição

Uma árvore binária ***T*** é denominada AVL quando, para todo nodo raiz da árvore ***T***, as alturas (**H**) de suas subárvores esquerda e direita (**SE** e **SD**) não excede a uma unidade. Se um nodo raiz não satisfizer a condição de altura, então ele é considerado desbalanceado, e uma árvore que contenha um nodo nessas condições é também chamada desbalanceada. A árvore da figura 1(a) é uma árvore AVL pois na avaliação de todos os nodos raízes a diferença das alturas de suas subárvore não excede a uma unidade. Entretanto, a árvore da figura 1(b) não é AVL pois a diferença das alturas das subárvores do nodo 150 é maior que uma unidade.



(a)
H(100) = | H (SD) – H (SE) |
H(100) = | 2 – 1 | = 1

(b)
H(150) = | H (SD) – H (SE) |
H(150) = | 2 – 0 | = 2

Fig. 1

2. Balanceamento

Processo que busca a distribuição equilibrada dos nodos, de modo a otimizar a operação de consulta, isto é, minimizar o número médio de comparações necessárias para localizar uma chave. É desejável que as chaves mais solicitadas

estejam próximas à raiz. Uma árvore é completamente balanceada se a distância média dos nodos até a raiz for mínima. A figura 2 ilustra uma árvore que deixa de ser balanceada na inserção de novo nodo.

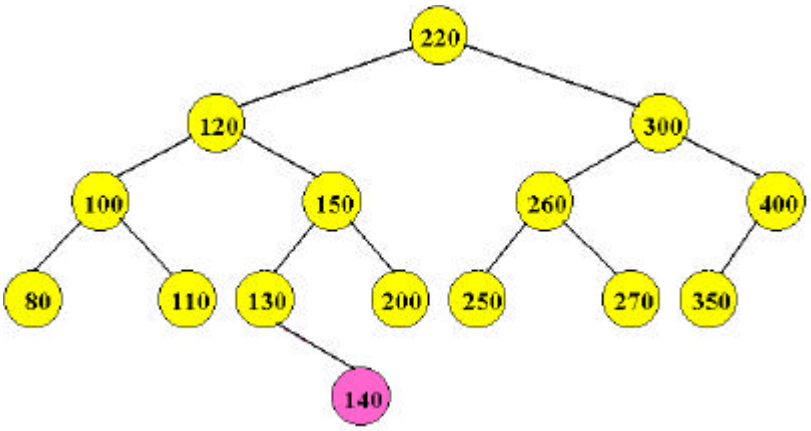


Fig. 2

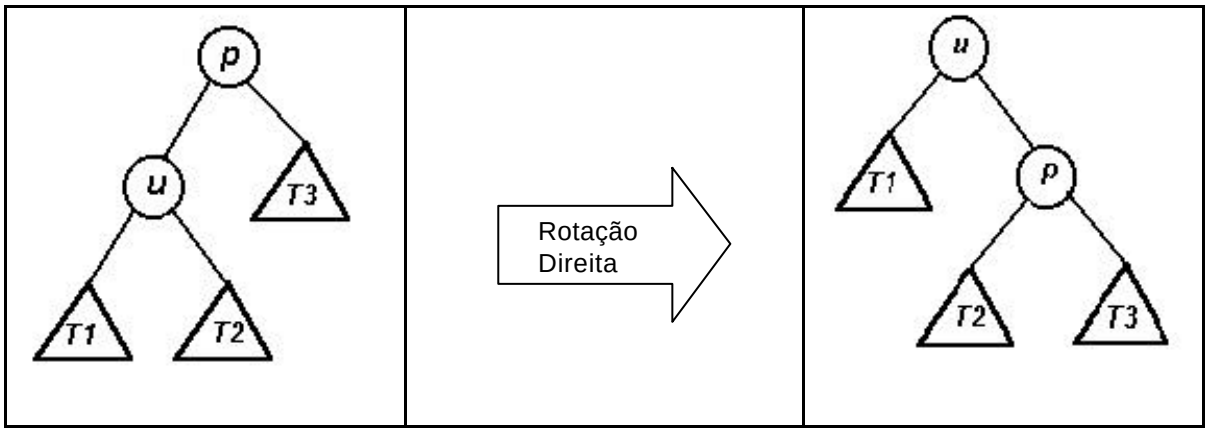
3. Inserção em árvores AVL

Seja **T** uma árvore AVL onde serão efetuadas inclusões de nodos. Para **T** continuar sendo uma árvore AVL (uma árvore balanceada) é preciso restabelecer o balanceamento se necessário.

A idéia consiste em verificar, se algum nodo raiz ***p*** está desbalanceado, isto é, se a diferença de altura entre as duas subárvores de ***p*** ficou maior que 1. Neste caso, e necessário aplicar as transformações apropriadas como *rotação simples direita, rotação simples esquerda, rotação dupla direita* e *rotação dupla esquerda.*

3.1 Transformações em árvores AVL

As transformações estão indicadas na figura 3. A subárvores T_1, T_2, T_3 e T_4 apresentadas na figura podem ser vazias ou não.



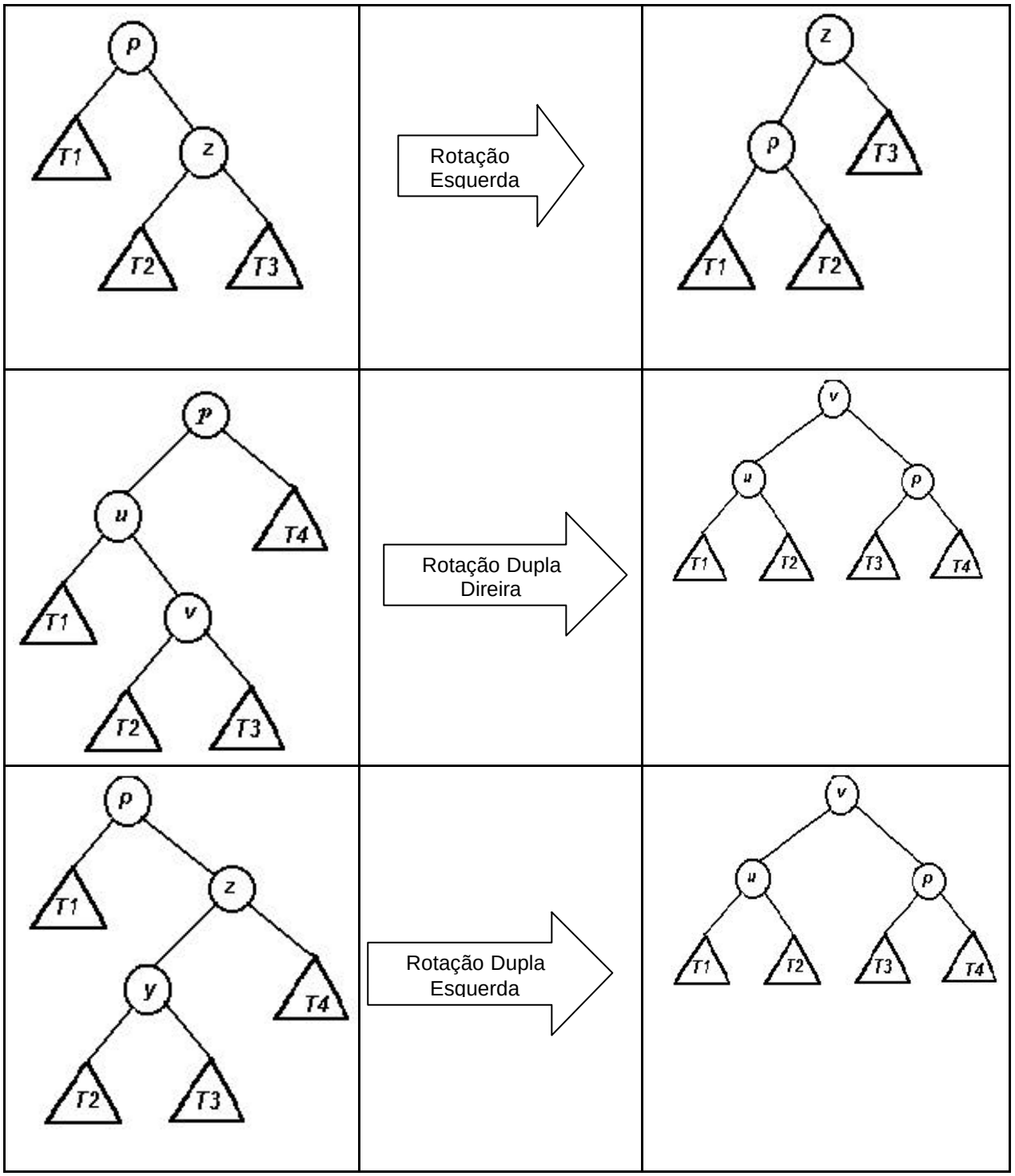


Fig. 3

Supondo-se que o nodo ***q*** foi incluído em ***T***. Se após a inclusão todos os nodos mantiveram-se balanceados, então a árvore manteve-se AVL e não há porque

aplicar as transformações. Caso contrário, seja ***p*** o nodo mais próximo às folhas de T que se tornou desbalanceada. Conclui-se que se T era uma árvore AVL, a inclusão de um nodo não pode aumentar em mais de uma unidade a altura de qualquer subávore, então:

$$H(\boldsymbol{p}) = | H(SE) - H(SD) | = 2$$

Pode-se identificar os seguintes casos:

- **Caso 1:** $H(SE_p) > H(SD_p)$

  O nodo inserido ***q*** pertence à subárvore esquerda de ***p***. Além disso, ***p*** possui o filho esquerdo ***u***, logo $H(SE_u) \neq H(SD_u)$. Então

  - **Caso 1.1:** $H(SE_u) > H(SD_u) \Rightarrow$ aplicar rotação simples direita da raiz ***p***.
  - **Caso 1.2:** $H(SE_u) < H(SD_u) \Rightarrow$ aplicar rotação dupla direita da raiz ***p***.

- **Caso 2:** $H(SE_p) < H(SD_p)$

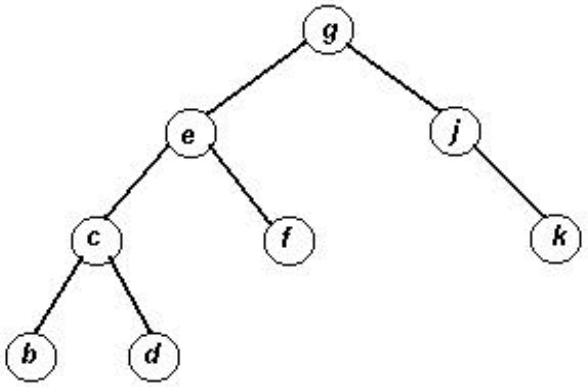
  O nodo inserido ***q*** pertence à subárvore direita de ***p***. Além disso, ***p*** possui o filho direito ***z***, logo $H(SE_z) \neq H(SD_z)$. Então

  - **Caso 2.1:** $H(SE_z) < H(SD_z) \Rightarrow$ aplicar rotação simples esquerda da raiz ***p***.
  - **Caso 2.2:** $H(SE_u) > H(SD_u) \Rightarrow$ aplicar rotação dupla esquerda da raiz ***p***.

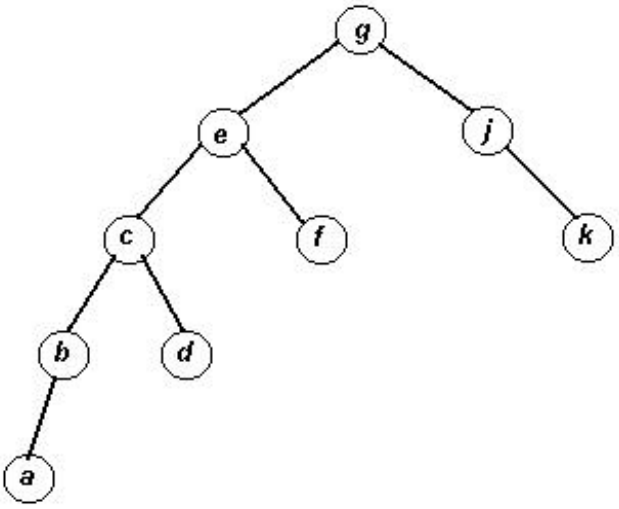
Conclui-se que o nodo pai de **p**, após a transformação, encontra-se também balanceado, pois a transformação diminui de uma unidade a altura da subárvore da raiz p. Isto assegura o balanceamento de toda a árvore e uma única operação é suficiente para balancear **T**.

Considerando a árvore AVL da figura 4 (a) e a inclusão do nodo ***a***. Após a inclusão do nodo ***a***, a árvore se desbalanceou. A subárvore da esquerda do nodo ***e*** excedeu em 2 unidades a da direita, além de ser o nodo mais próximo às folhas da árvore (Fig. 4(b)).

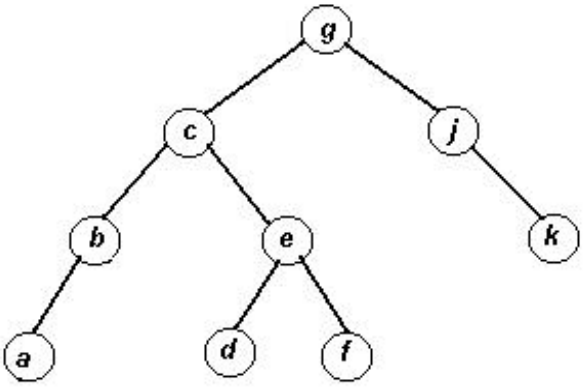
O novo nodo a pertence à subárvore esquerda do filho **c** de **e**. Logo, a situação corresponde ao caso 1 e a rotação simples direita sobre a raiz e torna esse nodo balanceado. Observa-se que o nodo **g** também está desbalanceado e a rotação utilizada também o balanceou Fig. 4(c).



(a)



(b)



(c)

Fig. 4

3.2 Implementação da inserção de nodos em árvores AVL

Seja ***T*** uma árvore AVL e ***c*** a chave a ser incluída em um novo nodo ***q***. O processo de inclusão é dividido em:

1) Verificar se ***c*** já se encontra em ***T***.
   a) Se ***c*** já estiver em ***T***, então processo é encerrado, pois trata-se de chave dupla.
   b) Caso contrário, encontrar o local correto de inserção do novo nodo e inserir dados no novo nodo.

2) Verificar se a operação de inserção desbalanceou a árvore.
   a) Em caso negativo, o processo termina pois ***T*** continuou sendo uma árvore AVL.
   b) Caso contrário, o balanceamento de ***T*** deverá ser efetuado e para tanto deve-se determinar qual operação de transformação deverá ser realizada para que ***T*** volte a condição de árvore AVL.

Para verificar se algum nodo ***v*** de ***T*** se tornou desbalanceado, basta determinar as alturas de suas subárvores e subtrair uma da outra. Para tanto, define-se o balanço de ***v***, para cada nodo ***v*** de ***T***:

$$Balanço\ (v) = H(SE_v) - H(SD_v)$$

Conclui-se que ***v*** estará balanceado se $-1 \leq Balanço\ (v) \leq 1$.

Para determinar qual a operação de transformação a ser efetuada deve-se verificar qual o acréscimo na altura de ***v***. A inserção do nodo ***q*** obrigatoriamente acarreta uma alteração na altura da subárvore esquerda ou direita de nodo ***w*** que se tornou seu pai (raiz). A utilização do campo *Balanço(v)* permite avaliar se a alteração pode, ou não, se propagar aos outros nodos ***v*** do caminho de ***w*** até a raiz da árvore.

Por exemplo, se ***q*** for inserido na subárvore esquerda de ***v***, a análise se inicia com ***v = w*** e prossegue em seus nodos ancestrais, de forma recursiva. O processo termina quando for constatado que a altura T_v não foi modificada, ou de que ***v*** se tornou balanceado. Podem ocorrer 3 casos distintos:

1) *Balanço(v)* = 1 antes da inclusão

   Balanço(v) = 0 e a altura da subárvore de raiz ***v*** não foi alterada. Consequentemente, as alturas dos nodos ancestrais de ***v*** até a raiz não se alteram.

2) *Balanço(v)* = 0 antes da inclusão

   Balanço(v) = -1 e a altura da subárvore de raiz ***v*** foi alterada, mas não provoca desbalanceamento em ***v***. Consequentemente, as alturas dos nodos ancestrais de ***v*** até a raiz podem ter sido alteradas e devem ser analisadas.

3) *Balanço(v)* = -1 antes da inclusão

Balanço(v) = -2 e ***v*** se desbalanceou. Consequentemente, é necessário aplicar a operação de transformação adequada e encerrar pois a altura volta a ser igual antes da inclusão e as alturas dos nodos ancestrais de ***v*** não necessitam de avaliação.

Para uma inserção na subárvore direita de ***v***, os mesmos casos devem ser verificados apenas considerando a simetria dos casos.

O algoritmo a seguir busca e insere, se possível, uma nova chave **c** numa árvore AVL. A chamada externa é *insereAVL (c, ptr, h)* onde **ptr** é o ponteiro para o nodo raiz da árvore e **h** é um parâmetro lógico que informa se houve, ou não, alteração na altura da árvore fazendo com que *Balanço(v)* seja atualizado.

```
Programa Arvore_AVL;
tipo
    ptr = ↑arvore
    arvore = registro
                esq : ptr;
                chave : inteiro;
                bal : inteiro
                dir : ptr;
             fim;

var
    c : inteiro;
    h : booleano;
    ptrU : ptr;
    ptrV : ptr;
    ptrRaiz : ptr;

procedimento novoNodo (ptr, c); // monta um novo nodo para inserção na árvore AVL
início
    aloca(ptr);  // aloca um nodo com endereço ptr
    ptr↑.esq := nulo;
    ptr↑.esq := nulo;
    ptr↑.chave := c;
    ptr↑.bal := 0;
fim

procedimento rotaçãoDireita (ptr, h); // faz a inserção quando se aplica rotação direita
início
    ptrU := ptr↑.esq;  // variável tipo ponteiro auxiliar
    Se ptrU↑.bal = -1
    Então // rotação simples direita
        ptr↑.esq := ptrU↑.dir;
        ptrU↑.dir := ptr;
        ptrU↑.bal := 0;
```

```
            ptr := ptrU;
        Senão // rotação dupla direita
            ptrV := ptrU↑.dir;  // variável tipo ponteiro auxiliar
            ptrU↑.dir := ptrV↑.esq;
            ptrV↑.esq := ptrU;
            ptr↑.esq := ptrV↑.dir;
            ptrV↑.dir := ptr;
            Se ptrV↑.bal = -1 Então ptr↑.bal := 1 Senão ptr↑.bal := 0;
            ptr := ptrV;
            // termina senão
        ptr↑.bal := 0;
        h := falso;
fim

procedimento rotaçãoEsquerda (ptr, h); // faz a inserção quando se aplica rotação
                                       // esquerda
início
        ptrU := ptr↑.dir;  // variável tipo ponteiro auxiliar
        Se ptrU↑.bal = -1
        Então // rotação simples esquerda
            ptr↑.dir := ptrU↑.esq;
            ptrU↑.esq := ptr;
            ptrU↑.bal := 0;
            ptr := ptrU;
        Senão // rotação dupla esquerda
            ptrV := ptrU↑.esq;  // variável tipo ponteiro auxiliar
            ptrU↑.esq := ptrV↑.dir;
            ptrV↑.dir := ptrU;
            ptr↑.dir:= ptrV↑.esq;
            ptrV↑.esq := ptr;
            Se ptrV↑.bal = 1 Então ptr↑.bal := -1 Senão ptr↑.bal := 0;
            ptr := ptrV;
            // termina senão
        ptr↑.bal := 0;
        h := falso;
fim

procedimento insereAVL (c, ptr, h);
início
        Se ptr = nulo
        Então  // árvore vazia
            novoNodo (ptr, c);
            h := verdadeiro;
        Senão
            Se c = ptr↑.chave então Msg "chave já existe na árvore"; Sair;
            Se c < ptr↑.chave
            Então  // inserção na subárvore esquerda
                InsereAVL (c, ptr↑.esq, h)
                Se h então
```

```
                    Caso ptr↑.bal seja
                    -1:   rotacaoDireita (ptr, h); // rebalanceamento
                     0:   ptr↑.bal := -1;
                     1:   ptr↑.bal := 0;
                          h := falso;
                    outro: Erro;
          Senão // inserção na subárvore direita
               InsereAVL (c, ptr↑.dir, h)
               Se h então
                    Caso ptr↑.bal seja
                    -1:   ptr↑.bal := 0;
                          h := falso;
                     0:   ptr↑.bal := 1;
                     1:   rotacaoEsquerda (ptr, h); // rebalanceamento
                    outro: Erro;
fim
// PROGRAMA PRINCIPAL
início
     FIM = falso;
     ptr = nulo;
     Enquanto não FIM
          Ler item
          Se item = <valor indicativo fim de itens> então FIM = verdadeiro
          Senão
               insereAVL (c, ptr, falso);
fim
```

4. Remoção em árvores AVL

Basicamente, após a exclusão do nodo desejado, a idéia consiste em verificar se a árvore ficou desbalanceada. Como no caso da inclusão, o processo pode se restringir ao exame dos nodos no caminho da raiz até a folha. Contudo, não se consegue balancear a árvore com apenas uma rotação. É necessária uma rotação adicional e o processo se encerra.

5. Exercícios

1) Provar ou dar contra-exemplo: toda árvore estritamente binária é balanceada.

2) Mostrar que a rotação dupla esquerda (ou direita) pode ser obtida por uma rotação direita (esquerda) seguida por uma rotação esquerda (ou direita);

3) Detalhar o algoritmo de exclusão dos nodos em árvores AVL;

4) Determinar a probabilidade de que uma inclusão efetuada em uma árvore AVL provoque o seu desbalanceamento, supondo que a inclusão pode ocorrer, em idêntica probabilidade, em qualquer das subárvores vazias.

5) Determinar a probabilidade de que uma exclusão efetuada em uma árvore AVL provoque o seu desbalanceamento, supondo que todos os nodos possuem a mesma probabilidade de serem excluídos.

6) Desenhar a árvore AVL com todos os seus atributos (endereço de alocação do nodo, conteúdos dos endereços direto e esquerdo do nodo, sua chave e outras informações) obtida pela seqüência de inserções das chaves abaixo. Os números entre parênteses definem o “endereço” do nodo alocado.

[50 (1), 90 (15), 70 (30), 40 (8), 30 (23), 70 (30), 20 (10), 100 (9), 60 (7), 80 (12), 10 (27)]