Algoritmos e Estruturas de Dados II
Algoritmos para Árvores AVL

Bacharelado em Análise de Sistemas, DCT–UFMS, 22/6/2005

1 Inserção em árvores AVL

1.1 Idéia da Inserção

Suponha que um nó q foi incluído em uma árvore AVL T. Note que a inclusão deve garantir que a árvore resultante é uma árvore binária de busca e, além disso, balanceada, conforme o critério de balanceamento de árvores AVL. Se após a inserção desse nó q todos os nós mantêm-se regulados, então T é uma árvore AVL e não há nada a fazer. Se, ao contrário, existe um ou mais nós desregulados em T, então devemos retificar T de modo que se transforme em uma árvore AVL.

Seja p o nó mais próximo às folhas de T que se tornou desregulado. Sejam $h_E(p)$ e $h_D(p)$ as alturas das subárvores esquerda de direita de p, respectivamente. Note que ou $h_E(p) - h_D(p) = 2$ ou $h_D(p) - h_E(p) = 2$. Veja a figura 1 para um exemplo.

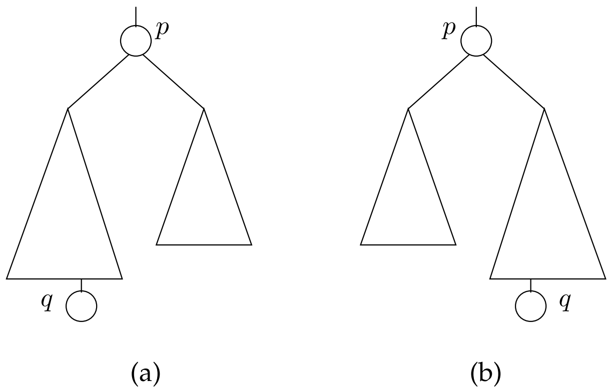


Figura 1: Nó desregulado p em uma árvore T devido à inclusão de um nó q. O nó p torna-se desregulado devido ao aumento da altura de sua subárvore esquerda (a) ou direita (b).

Então, a partir da figura 1, identificamos dois casos:

Caso 1. $h_E(p) > h_D(p)$

Como pode ser observado na figura 1(a), q pertence à subárvore esquerda de p. Além disso, p tem um filho esquerdo u com a propriedade que $h_E(u) \neq h_D(u)$, conforme mostra a figura 2.

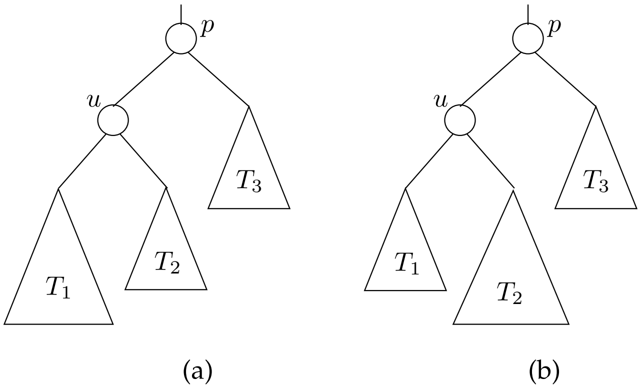


Figura 2: Caso 1, onde $h_E(p) > h_D(p)$. Temos dois subcasos: (a) ou q foi inserido na subárvore esquerda de u ou (b) q foi inserido na subárvore direita de u.

Temos então dois subcasos:

Caso 1.1. $h_E(u) > h_D(u)$

Este caso corresponde à figura 2(a). Note que q pertence à árvore T_1, $h(T_1) = h(T_2) + 1$ e $h(T_2) = h(T_3)$. Neste caso, usamos um processo para reestabelecer a regulagem do nó p denomidado **rotação direita**. Veja a figura 3 para um exemplo.

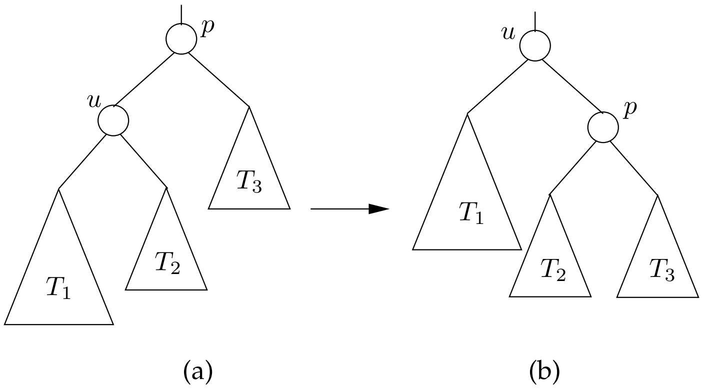


Figura 3: Caso 1.1, onde $h_E(p) > h_D(p)$ e $h_E(u) > h_D(u)$ e uma rotação direita é realizada.

Caso 1.2. $h_D(u) > h_E(u)$

Este caso corresponde à figura 2(b). Note que u tem um filho direito v, conforme ilustra a figura 4(a). Note também que $|h(T_2) - h(T_3)| \leqslant 1$ e $\max\{h(T_2), h(T_3)\} = h(T_1) = h(T_4)$. Então, usamos um processo para reestabelecer a regulagem do nó p denomidado **dupla rotação direita**. Veja a figura 4 para um exemplo.

Caso 2. $h_D(p) > h_E(p)$

Como pode ser observado na figura 1(b), q pertence à subárvore direita de p. Além disso,

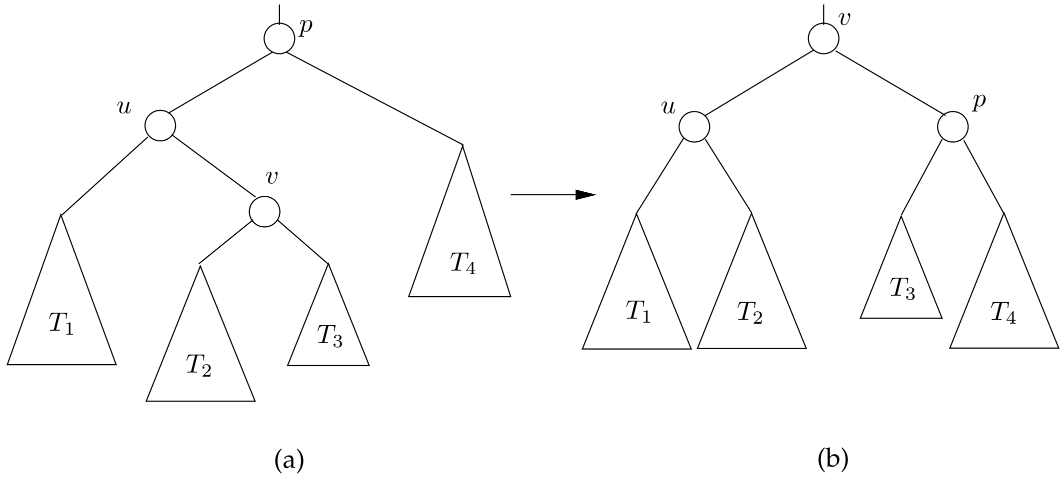


Figura 4: Caso 1.2, onde $h_E(p) > h_D(p)$ e $h_D(u) > h_E(u)$ e uma dupla rotação direita é realizada. Nesta figura estamos supondo que o nó q foi inserido na árvore T_2 e então $h(T_2) - h(T_3) \leqslant 1$, $h(T_2) = h(T_1) = h(T_4)$.

p tem um filho direito z com a propriedade que $h_E(z) \neq h_D(z)$, conforme mostra a figura 5. Note ainda que o Caso 2 é um espelho do Caso 1.

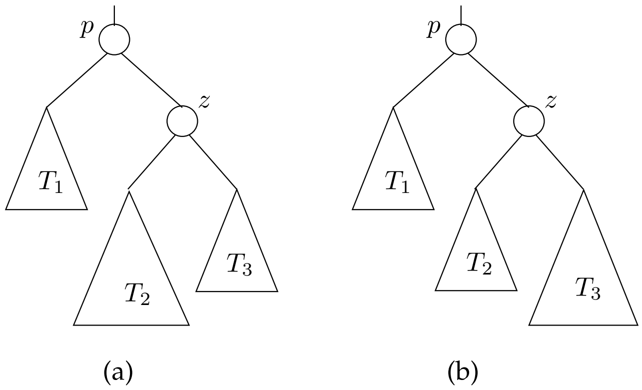


Figura 5: Caso 2, onde $h_D(p) > h_E(p)$. Temos dois subcasos: (a) ou q foi inserido na subárvore esquerda de z ou (b) q foi inserido na subárvore direita de z.

Temos então dois subcasos:

Caso 2.1. $h_D(z) > h_E(z)$

Este caso corresponde à figura 5(b). Note que q pertence à árvore T_3, $h(T_3) = h(T_2) + 1$ e $h(T_2) = h(T_1)$. Neste caso, usamos um processo para reestabelecer a regulagem do nó p denomidado **rotação esquerda**. Veja a figura 6 para um exemplo. Perceba também a simetria com a rotação direita do Caso 1.1.

Caso 2.2. $h_E(z) > h_D(z)$

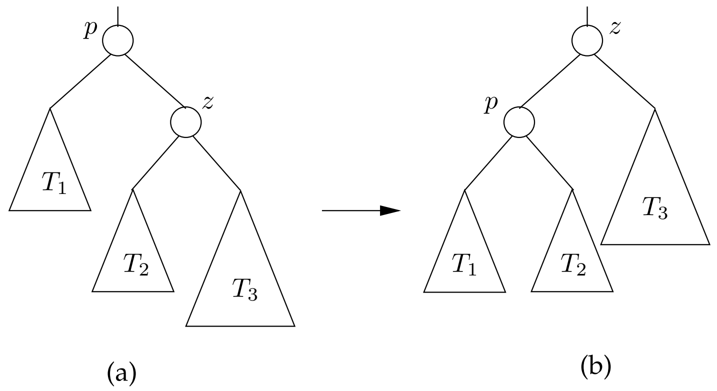


Figura 6: Caso 2.1, onde $h_D(p) > h_E(p)$ e $h_D(z) > h_E(z)$ e uma rotação esquerda é realizada.

Este caso corresponde à figura 5(a). Note que z tem um filho esquerdo y, conforme ilustra a figura 7(a). Note também que $|h(T_2) - h(T_3)| \leqslant 1$ e $\max\{h(T_2), h(T_3)\} = h(T_1) = h(T_4)$. Então, usamos um processo para reestabelecer a regulagem do nó p denomidado **dupla rotação esquerda**. Veja a figura 7 para um exemplo.

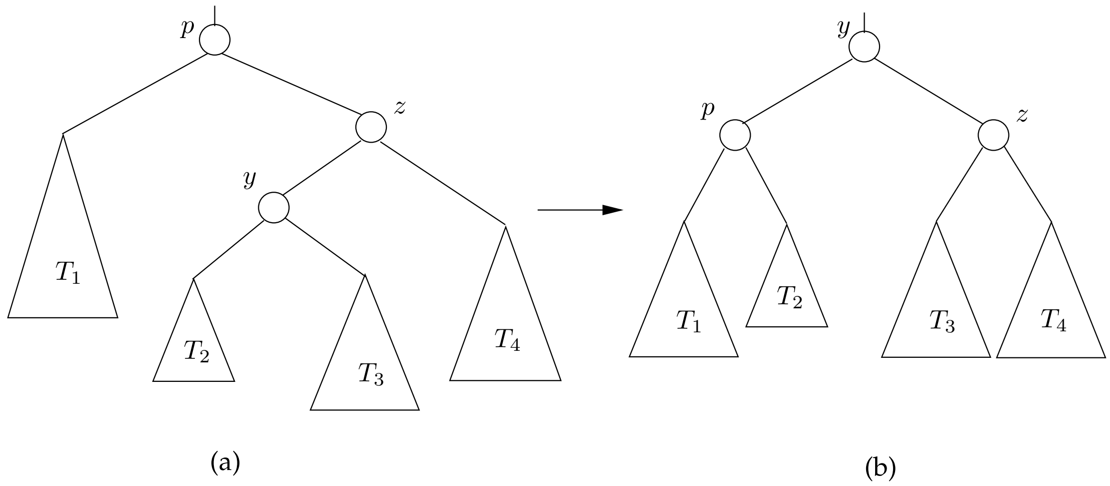


Figura 7: Caso 2.2, onde $h_D(p) > h_E(p)$ e $h_E(z) > h_D(z)$ e uma dupla rotação esquerda é realizada. Nesta figura estamos supondo que o nó q foi inserido na árvore T_3 e então $h(T_3) - h(T_2) \leqslant 1$, $h(T_3) = h(T_1) = h(T_4)$.

Note que não há outras possibilidades para restabelecimento da regulagem do nó p, ou seja, os casos apresentados cobrem todas as possibilidades e um desses casos ocorrerá quando um nó p em uma árvore AVL estiver desregulado. Note também que, após a execução de uma das rotações descritas nos casos acima, a regulagem do nó p é reestabelecida. Além disso, todos os nós ancestrais ao nó p também têm suas regulagens reestabalecidas, o que indica que uma única transformação como essa é suficiente para que a árvore T torne-se novamente uma árvore AVL.

1.2 Procedimento de Inserção

Seja T uma árvore AVL e x a chave a ser incluída em T. Um procedimento geral para inserção na árvore AVL T é apresentado a seguir.

INSERÇÃO–GERAL–AVL

buscar x na árvore T
se $x \in T$ **então**
  chave já pertence a T
senão
  a busca indica o local correto do novo nó
  inserção do novo nó em T
  se não existe nó desregulado em T **então**
    a árvore T continua uma árvore AVL
  senão
    regular nó

O problema neste procedimento geral de inserção em uma árvore AVL é verificar se algum nó v de T se tornou desregulado após a inserção ter sido realizada. Uma solução ingênua para verificar se um nó tornou-se desregulado é determinar as alturas de suas subárvores esquerda e direita ($h_E(v)$ e $h_D(v)$). Esta operação consome tempo $O(n)$, um tempo excessivo para uma árvore AVL. A solução empregada no procedimento de inserção é definir um atributo *bal* para cada nó v da árvore T da seguinte forma:

$$\mathit{bal}(v) := h_D(v) - h_E(v).$$

O nó v de T está regulado se e somente se $-1 \leqslant \mathit{bal}(v) \leqslant +1$.

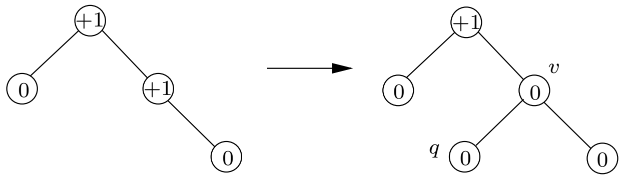
Após a inclusão de um nó q em uma árvore AVL T, a atualização desse atributo $\mathit{bal}(v)$ para um nó v de T de forma eficiente é uma questão fundamental. O procedimento de inserção que veremos atualiza esse atributo da seguinte forma. Se q pertencer à subárvore esquerda de v e essa inclusão ocasionar um aumento na altura dessa subárvore, então fazemos $\mathit{bal}(v) = \mathit{bal}(v) - 1$. Analogamente, se q pertencer à subárvore direita de v e provocar aumento de sua altura, fazemos $\mathit{bal}(v) = \mathit{bal}(v) + 1$.

A questão mais importante, isto é, em quais situações a inserção de um nó q em uma árvore AVL T provoca um acréscimo na altura $h(v)$ da subárvore de raiz v, é respondida com a seguinte observação: após a inserção do nó q ocorre obrigatoriamente a alteração da altura da subárvore esquerda ou direita do nó pai w de v. Então, a situação do atributo *bal* permite verificar se a alteração na altura da subárvore esquerda ou direita de q pode, ou não, se propagar aos outros nós no caminho de w até a raiz de T.

Suponha que o nó q é inserido na subárvore esquerda de v. Iniciamos uma análise com $v = w$ e prosseguimos com seus nós ancestrais de forma recursiva. O processo se encerra quando da constatação de que a altura da subárvore de raiz v não foi modificada ou de que v se tornou regulado. Temos de analisar então três casos:

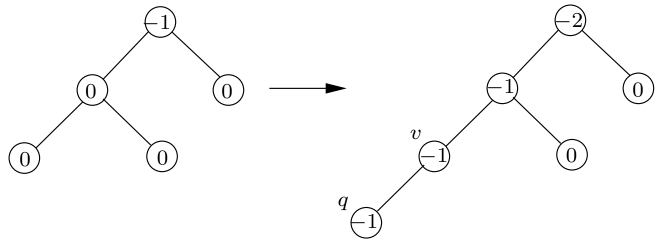
Caso 1. $bal(v) = +1$ antes da inserção de q.

Neste caso, $bal(v)$ torna-se 0, a altura da subárvore de raiz v não é modificada e os nós do caminho de v até a raiz não se alteram.



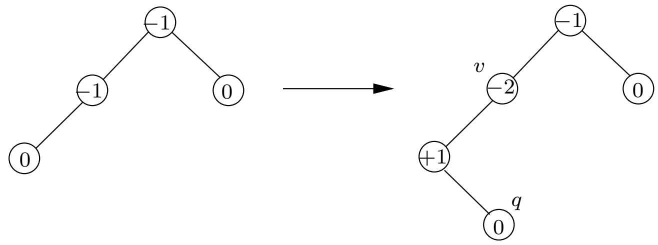
Caso 2. $bal(v) = 0$ antes da inserção de q.

Neste caso, *bal* torna-se -1 e a altura da subárvore de raiz v é modificada. Os nós restantes do caminho de v até a raiz também podem ter suas alturas modificadas e devem ser analisados. Se v é a raiz de T, o processo se encerra. Caso contrário, repetimos esse processo com v substituído pelo seu pai.



Caso 3. $bal(v) = -1$ antes da inserção de q.

Observe que *bal* vale -2 e o nó v está desregulado. Uma rotação apropridada deve ser empregada. Qualquer rotação implica que a subárvore resultante tem a mesma altura da subárvore anterior e as alturas dos nós ancestrais de v não precisam de avaliação.



Observe, finalmente, que se uma inserção na subárvore direita de v é realizada, então casos simétricos aos anteriores devem ser considerados.

A complexidade de tempo do procedimento de inserção, apresentado a seguir, depende da operação de busca na árvore. Como essa busca tem tempo de execução proporcional à altura da árvore, e uma árvore AVL tem altura logarítmica no número de chaves que possui, o tempo de execução da inserção é $O(\log n)$, onde n é o número de chaves na árvore.

INSERE-AVL(x, *apt*, h): recebe uma chave x a ser inserida na árvore AVL, um apontador *apt* para um nó da árvore e um indicador lógico h; a variável x é passada por cópia e as variáveis *apt* e h são passadas por referência.

```
se apt = λ então
  INICIALIZA-NÓ(x, apt)
  h ← verdadeiro
senão
  se x = apt↑.chave então
    h ← falso
  se x < apt↑.chave então
    INSERE-AVL(x, apt↑.esq, h)
    se h então
      caso apt↑.bal seja
        +1  :  apt↑.bal ← 0
               h ← falso
         0  :  apt↑.bal ← −1
        −1  :  CASO1(apt, h)          ▷ Rebalanceamento
  senão
    INSERE-AVL(x, apt↑.dir, h)
    se h então
      caso apt↑.bal seja
        −1  :  apt↑.bal ← 0
               h ← falso
         0  :  apt↑.bal ← +1
        +1  :  CASO2(apt, h)          ▷ Rebalanceamento
```

INICIALIZA-NÓ(x, *apt*): recebe uma chave x e um apontador *apt* para um nó da árvore; a variável x é passada por cópia e a variável *apt* por referência.

```
ALOCAR(apt)
apt↑.chave ← x
apt↑.esq ← λ
apt↑.dir ← λ
apt↑.bal ← 0
```

CASO1(apt, h): recebe um apontador *apt* para um nó de uma árvore AVL e um indicador lógico h, passados por referência.

$aux1 \leftarrow apt\uparrow.esq$
se $aux1\uparrow.bal = -1$ **então**
 ▷ CASO 1.1 – rotação direita
 $apt\uparrow.esq \leftarrow aux1\uparrow.dir$
 $aux1\uparrow.dir \leftarrow apt$
 $apt\uparrow.bal \leftarrow 0$
 $apt \leftarrow aux1$
senão
 ▷ CASO 1.2 – dupla rotação direita
 $aux2 \leftarrow aux1\uparrow.dir$
 $aux1\uparrow.dir \leftarrow aux2\uparrow.esq$
 $aux2\uparrow.esq \leftarrow aux1$
 $apt\uparrow.esq \leftarrow aux2\uparrow.dir$
 $aux2\uparrow.dir \leftarrow apt$
 se $aux2\uparrow.bal = -1$ **então**
  $apt\uparrow.bal \leftarrow +1$
 senão
  $apt\uparrow.bal \leftarrow 0$
 se $aux2\uparrow.bal = +1$ **então**
  $aux1\uparrow.bal \leftarrow -1$
 senão
  $aux1\uparrow.bal \leftarrow 0$
 $apt \leftarrow aux2$
$apt\uparrow.bal \leftarrow 0$
$h \leftarrow$ **falso**

CASO2(apt, h): recebe um apontador *apt* para um nó de uma árvore AVL e um indicador lógico h, passados por referência.

$aux1 \leftarrow apt\uparrow.dir$
se $aux1\uparrow.bal = +1$ **então**
  ▷ CASO 2.1 – rotação direita
  $apt\uparrow.dir \leftarrow aux1\uparrow.esq$
  $aux1\uparrow.esq \leftarrow apt$
  $apt\uparrow.bal \leftarrow 0$
  $apt \leftarrow aux1$
senão
  ▷ CASO 2.2 – dupla rotação esquerda
  $aux2 \leftarrow aux1\uparrow.esq$
  $aux1\uparrow.esq \leftarrow aux2\uparrow.dir$
  $aux2\uparrow.dir \leftarrow aux1$
  $apt\uparrow.dir \leftarrow aux2\uparrow.esq$
  $aux2\uparrow.esq \leftarrow apt$
  se $aux2\uparrow.bal = +1$ **então**
    $apt\uparrow.bal \leftarrow -1$
  senão
    $apt\uparrow.bal \leftarrow 0$
  se $aux2\uparrow.bal = -1$ **então**
    $aux1\uparrow.bal \leftarrow +1$
  senão
    $aux1\uparrow.bal \leftarrow 0$
  $apt \leftarrow aux2$
$apt\uparrow.bal \leftarrow 0$
$h \leftarrow$ **falso**

2 Remoção em árvores AVL

A remoção em uma árvore AVL tem dois passos principais. No primeiro, o nó contendo a chave a ser removida é buscado na estrutura. O nó encontrado pode ter 0, 1 ou 2 filhos e essas opções encaminham o procedimento para um dos casos de remoção equivalentes ao caso da remoção em árvores binárias de busca. Com 0 filhos, o nó é removido, só alterando o apontador esquerdo ou direito de seu pai. Com 1 filho, o nó também é removido, não antes de modificar corretamente o apontador esquerda ou direita de seu pai. Com 2 filhos, o sucessor da chave do nó contendo a chave a ser removida é buscado na árvore; sua chave é copiada para esse ponto da árvore e, em seguida, o nó contendo o sucessor original é removido (esse nó tem 0 ou 1 filho).

Depois disso, o segundo passo encarrega-se de regular algum nó que tornou-se desregulado na estrutura após essa remoção. Essa regulagem é feita usando uma das rotações descritas no procedimento de inserção (rotação esquerda, dupla rotação esquerda, rotação direita ou dupla rotação direita). Observe ainda que a regulagem pode se estender desse nó desregulado mais próximo das folhas da árvore até a sua raiz.

O procedimento de remoção a seguir leva em conta esses passos, mas, diferentemente de como vimos na remoção em árvores binárias de busca, de forma recursiva.

3 Tempos de execução

O tempo de execução de pior caso dos procedimentos de inserção e remoção em uma árvore AVL são dominados pelo tempo de execução do procedimento de busca de uma chave em uma árvore AVL. As operações restantes são todas operações de troca de apontadores, comparações, atribuições, etc.

O tempo de execução de pior caso do procedimento de busca em uma árvore AVL é proporcional à altura da árvore. Como vimos, uma árvore AVL tem altura proporcional a $O(\log n)$, onde n é a quantidade de nós que a árvore possui. Isso significa que os procedimentos de busca, inserção e remoção em uma árvore AVL têm complexidade de tempo $O(\log n)$, cada um deles, o que significa que essa estrutura é bastante eficiente para armazenamento e gerenciamento de informações.

REMOVE–AVL(x, *apt*, h): recebe uma cópia da chave x a ser removida da árvore, um apontador *apt* para a raiz da árvore e um indicador lógico h, passados por referência.

```
se apt = λ então
  escreva "x não pertence à árvore."
senão
  se x < apt↑.chave então
    REMOVE–AVL(x, apt↑.esq, h)
    se h então
      caso apt↑.bal seja
        −1  :  apt↑.bal ← 0
         0  :  apt↑.bal ← +1
               h ← falso
        +1  :  REGULA–ESQ(apt, h)          ▷ Rebalanceamento
  senão
    se x > apt↑.chave então
      REMOVE–AVL(x, apt↑.dir, h)
      se h então
        caso apt↑.bal seja
          +1  :  apt↑.bal ← 0
           0  :  apt↑.bal ← −1
                 h ← falso
          −1  :  REGULA–DIR(apt, h)        ▷ Rebalanceamento
    senão
      aux ← apt                            ▷ nó apontado por aux será removido
      se aux↑.dir = λ então
        apt ← aux↑.esq
        h ← verdadeiro
      senão
        se aux↑.esq = λ então
          apt ← aux↑.dir
          h ← verdadeiro
        senão
          REMOVE(apt↑.esq, aux, h)
          se h então
            REGULA–ESQ(apt, h)
      LIBERAR(aux)
```

REMOVE(p, q, h): recebe referências a apontadores p e q e de um indicador lógico h.

se $p\uparrow.dir \neq \lambda$ **então**
  REMOVE($p\uparrow.dir, q, h$)
  REGULA-DIR(p, h)
senão
  $q\uparrow.chave \leftarrow p\uparrow.chave$
  $aux \leftarrow p$
  $p \leftarrow p\uparrow.esq$
  LIBERAR(aux)
  $h \leftarrow$ **verdadeiro**

REGULA–ESQ(apt, h): recebe referências ao apontador apt e à variável lógica h.

$aux1 \leftarrow apt\uparrow.dir$
se $aux1\uparrow.bal \geqslant 0$ **então**
  $apt\uparrow.dir \leftarrow aux1\uparrow.esq$
  $aux1\uparrow.esq \leftarrow apt$
  se $aux1\uparrow.bal = 0$ **então**
    $apt\uparrow.bal \leftarrow +1$
    $aux1\uparrow.bal \leftarrow -1$
    $h \leftarrow$ **falso**
  senão
    $apt\uparrow.bal \leftarrow 0$
    $aux1\uparrow.bal \leftarrow 0$
  $apt \leftarrow aux1$
senão
  $aux2 \leftarrow aux1\uparrow.esq$
  $aux1\uparrow.esq \leftarrow aux2\uparrow.dir$
  $aux2\uparrow.dir \leftarrow aux1$
  $apt\uparrow.dir \leftarrow aux2\uparrow.esq$
  $aaux2\uparrow.esq \leftarrow apt$
  se $aux2\uparrow.bal = +1$ **então**
    $apt\uparrow.bal \leftarrow -1$
  senão
    $apt\uparrow.bal \leftarrow 0$
  se $aux2\uparrow.bal = -1$ **então**
    $aux1\uparrow.bal \leftarrow +1$
  senão
    $aux1\uparrow.bal \leftarrow 0$
  $apt \leftarrow aux2$
  $apt\uparrow.bal \leftarrow 0$

REGULA–DIR(*apt*, h): recebe referências ao apontador apt e à variável lógica h.

aux1 $\leftarrow$ *apt*$\uparrow$.*esq*
se *aux1*$\uparrow$.*bal* $\leqslant 0$ **então**
  apt$\uparrow$.*esq* $\leftarrow$ *aux1*$\uparrow$.*dir*
  aux1$\uparrow$.*dir* $\leftarrow$ *apt*
  se *apt*$\uparrow$.*bal* $= 0$ **então**
    apt$\uparrow$.*bal* $\leftarrow -1$
    aux1$\uparrow$.*bal* $\leftarrow +1$
    $h \leftarrow$ **falso**
  senão
    apt$\uparrow$.*bal* $\leftarrow 0$
    aux1$\uparrow$.*bal* $\leftarrow 0$
  apt $\leftarrow$ *aux1*
senão
  aux2 $\leftarrow$ *aux1*$\uparrow$.*dir*
  aux1$\uparrow$.*dir* $\leftarrow$ *aux2*$\uparrow$.*esq*
  aux2$\uparrow$.*esq* $\leftarrow$ *aux1*
  apt$\uparrow$.*esq* $\leftarrow$ *aux2*$\uparrow$.*dir*
  aux2$\uparrow$.*dir* $\leftarrow$ *apt*
  se *aux2*$\uparrow$.*bal* $= -1$ **então**
    apt$\uparrow$.*bal* $\leftarrow +1$
  senão
    apt$\uparrow$.*bal* $\leftarrow 0$
  se *aux2*$\uparrow$.*bal* $= +1$ **então**
    aux1$\uparrow$.*bal* $\leftarrow -1$
  senão
    aux1$\uparrow$.*bal* $\leftarrow 0$
  apt $\leftarrow$ *aux2*
  apt$\uparrow$.*bal* $\leftarrow 0$