Algoritmos de inserción y eliminación de un nodo en un árbol AVL

Andrés Gómez de Silva Garza

Las figuras en este documento fueron tomadas del siguiente libro (del cual también se adaptaron algunas de las descripciones del algoritmo):

Heileman, Gregory L., Data Structures, Algorithms, and Object-Oriented Programming, McGraw-Hill, New York, 1996. En la biblioteca del ITAM: 005.73 H466D.

Inserción:

1) Realizar la inserción usando el algoritmo de inserción en un árbol binario de búsqueda "normal".

2) Recalcular los factores de equilibrio (FE) de todos los nodos que hay entre el nuevo nodo y la raíz, inclusive.

3) Si se llega a la raíz y ningún nodo tiene un FE diferente de –1, 0 o +1 (los que se permiten en un árbol AVL), entonces el árbol resultante sigue siendo un árbol AVL y la inserción no provocó un desequilibrio.

4) En caso contrario, el primer nodo "problemático" NP (que tiene un FE de –2 o +2) encontrado al recalcular los FE indica en qué zona del árbol hay que hacer ajustes para re-equilibrarlo. Los ajustes implican hacer rotaciones, a veces una pero otras veces dos. Las rotaciones a veces son hacia la izquierda y a veces hacia la derecha, lo cual se ilustra gráficamente en la siguiente figura (fig. 11.1, pg. 295):

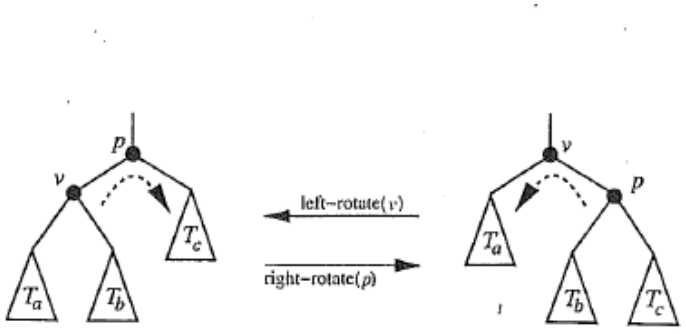

FIGURE 11.1
The two basic rotation operations used to implement all on the rotation patterns discussed in this chapter. The right-rotate(p) operation starts with the subtree on the left and produces the subtree on the right, while the left-rotate(v) operation starts with the subtree on the right and produces the one on the left.

5) Para determinar si se requiere una sola rotación o dos, primero hay que ver si el signo del FE de NP es – o +. Si el signo es negativo, hay que examinar el hijo izquierdo de NP; si el signo es positivo, hay que examinar el hijo derecho de NP. Sea H el nombre del hijo que se va a examinar. Si H tiene un FE cuyo signo es el mismo que el del FE de NP, entonces sólo habrá que hacer una rotación (paso 6); en caso contrario, se necesitarán dos rotaciones (paso 7). Como resultado de las rotaciones puede ser que cambie el nodo raíz (el

nodo que era la raíz del árbol antes de las rotaciones puede terminar ya no siéndolo, y algún otro nodo que no era raíz antes de las rotaciones lo reemplazaría).

6) En caso de que se requiera una sola rotación, la rotación va a ser hacia la derecha si el signo del FE de NP es negativo, o hacia la izquierda si el signo del FE de NP es positivo. En los dos casos el pivote de la rotación va a ser el nodo NP. En la terminología de Heileman (fig. 11.1, pg. 295), en caso de que se vaya a hacer una rotación hacia la derecha, NP va a ser *p* (y el hijo izquierdo de NP va a ser *v*), mientras que en caso de que se vaya a hacer una rotación hacia la izquierda, NP va a ser *v* (y el hijo derecho de NP va a ser *p*). Un ejemplo ilustrativo esquemático se muestra en la siguiente figura (fig. 11.5, pg. 299):

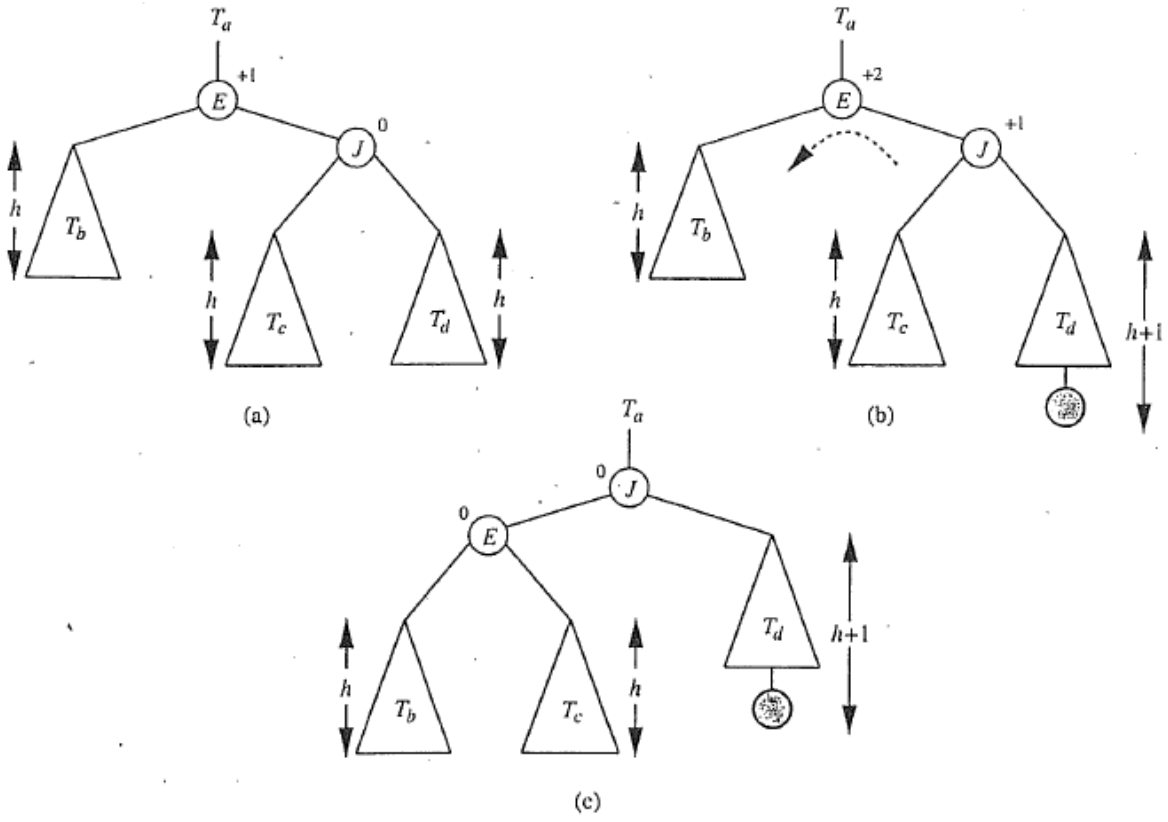

FIGURE 11.5
A single left rotation pattern applied at subtree T_a in an AVL tree. (a) Subtree T_a prior to insertion. The height of subtrees T_b through T_d are h, and the height of T_a as a whole is $h+2$. The balance of vertices E and J are $+1$ and 0, respectively. (b) Insertion of a vertex that results in the height of subtree T_d changing from h to $h+1$. This causes the balance of vertex E to become $+2$, necessitating a rebalancing of T_a. Subtree T_a is rebalanced by performing a left-rotate(E) operation. (c) Subtree T_a after rebalancing. Vertex E now has left subtree T_b, right subtree T_c, and appears to the left of vertex J. The balance of vertices E and J are now 0, and the height of T_a as a whole is $h+2$.

7) En caso de que se requieran dos rotaciones, las dos van a ser en sentidos opuestos (usando distintos pivotes). La primera rotación va a utilizar a H como pivote. Si el signo del FE de H es negativo, la primera rotación tiene que hacerse hacia la derecha; si el signo es positivo, la primera rotación tiene que hacerse hacia la izquierda. En la terminología de Heileman (fig. 11.1, pg. 295), en caso de que la rotación vaya a ser hacia la derecha, H va a

ser *p* (y el hijo izquierdo de H va a ser *v*), mientras que en caso de que la rotación vaya a ser hacia la izquierda, H va a ser *v* (y el hijo derecho de H va a ser *p*). La segunda rotación va a utilizar a NP como pivote y va a ser en sentido contrario a la primera rotación. En la terminología de Heileman (fig. 11.1, pg. 295), en caso de que esta segunda rotación vaya a ser hacia la derecha, NP va a ser *p* (y el nodo que haya terminado siendo el hijo izquierdo de NP después de la primera rotación va a ser *v*), mientras que en caso de que la rotación vaya a ser hacia la izquierda, NP va a ser *v* (y el nodo que haya terminado siendo el hijo derecho de NP después de la primera rotación va a ser *p*). Un ejemplo ilustrativo esquemático se muestra en la siguiente figura (fig. 11.6, pg. 300):

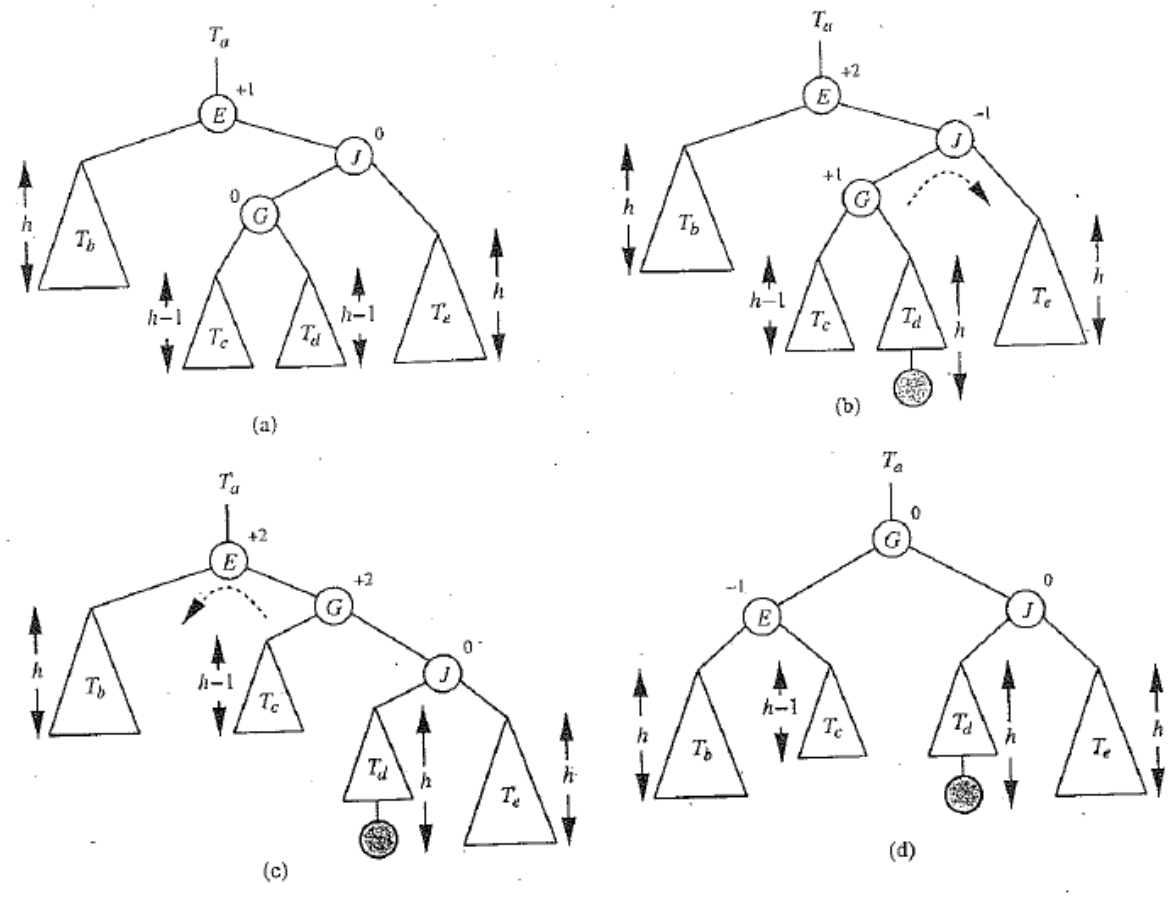

FIGURE 11.6
A double RL rotation performed after the insertion of an element in subtree T_a of an AVL tree. This rotation pattern consists of a right-rotate operation followed by a left-rotate operation.

Cada rotación implica modificar tres apuntadores.

- Para implementar una rotación hacia la derecha: sean X el nodo padre de *p* y Y el hijo derecho de *v*; si antes de la rotación *p* era el hijo izquierdo de X, después de la rotación *v* debe terminar siendo el hijo izquierdo de X; si antes de la rotación *p* era el hijo derecho de X, después de la rotación *v* debe terminar siendo el hijo derecho de X; después de la rotación Y debe terminar siendo el hijo izquierdo de *p*; después de la rotación *p* debe terminar siendo el hijo derecho de *v*.

- Para implementar una rotación hacia la izquierda: sean X el nodo padre de *v* y Y el hijo izquierdo de *p*; si antes de la rotación *v* era el hijo izquierdo de X, después de la rotación *p* debe terminar siendo el hijo izquierdo de X; si antes de la rotación *v* era el hijo derecho de X, después de la rotación *p* debe terminar siendo el hijo derecho de X; después de la rotación Y debe terminar siendo el hijo derecho de *v*; después de la rotación *v* debe terminar siendo el hijo izquierdo de *p*.

Eliminación:

1) Realizar la eliminación del nodo X siguiendo el algoritmo de eliminación de un árbol binario de búsqueda "normal" (que termina "promoviendo" el valor más pequeño Y almacenado en el subárbol derecho cuya raíz es X, reemplazando a X por dicho valor Y).

2) Empezar a recalcular los FE de todos los nodos que hay entre el nodo padre P de este nodo promovido Y (previo a la promoción) y la raíz, inclusive. Nota: si P resulta ser el nodo eliminado X, entonces hay que empezar a recalcular los FE de todos los nodos que hay entre el nodo que termina reemplazando a P/X (después de la promoción) y la raíz, inclusive.

3) Seguir con el paso 3 (y subsecuentes) del algoritmo de inserción (arriba). Al hacer esto, en el paso 5 al comparar los FE de dos nodos se considera que el signo de 0 es tanto negativo como positivo (es decir, si el FE de un nodo es 0, no importa cuál es el signo del otro nodo, se considera que es del mismo signo que el FE del nodo cuyo FE es 0).

4) En caso de que se haya necesitado realizar una o dos rotaciones (en el paso 6 o 7, respectivamente, del algoritmo de inserción, arriba) empezar a recalcular los FE de todos los nodos que hay entre el nodo que "ascendió de nivel" (visualmente, lo cual equivale a descender numéricamente) y la raíz, inclusive. (El nodo que "ascendió de nivel" en la terminología de Heileman será el nodo *v* en caso de que la última rotación haya sido hacia la derecha y *p* en caso de que la última rotación haya sido hacia la derecha). Regresar al paso 3.

5) Nota: como parte del proceso de rebalanceo (mediante rotaciones, dentro de los pasos 3 y/o 4) del árbol posterior a una eliminación, puede ser que tenga que cambiar la raíz del árbol (aún si el nodo que se eliminó no fue el nodo raíz; por otra parte, si el nodo que se eliminó fue la raíz a fuerzas va a tener que cambiar la raíz del árbol desde el paso 1).