

Arboles Binarios: revisión previa de definiciones

Arbol binario de búsqueda:

Es una estructura de datos formada por nodos, cada uno de los cuales tiene a lo sumo dos hijos, con un nodo distinguido llamado raíz, y las claves almacenadas sin repeticiones y totalmente ordenadas.

La definición recursiva de un árbol binario de búsqueda (ABB o BST, de 'binary search tree') especifica que o bien es nulo o bien consta de un nodo que contiene una clave y dos subárboles hijos, izquierdo y derecho, que son árboles binarios de búsqueda.

Para cada nodo del ABB se verifica que las claves del subárbol izquierdo son menores que la clave del nodo padre, y esta clave es menor que las de su subárbol derecho.

Un árbol binario de búsqueda **balanceado por su altura** con diferencia 1 es aquel para el cual para todo nodo se verifica que la diferencia de número de niveles entre su subárbol izquierdo y su subárbol derecho es menor o igual que 1.

Arboles AVL:

Los arboles AVL (Adelson-Velskii y Landis desarrollaron los algoritmos) son ABB balanceados por su altura que usan ciertas rotaciones para rebalancear el ABB cuando es necesario.

Estas estructuras agregan en los nodos una variable FB, el factor de balanceo, que permite determinar si es necesario realizar una rotación luego de una alta o una baja.

Alta en un arbol AVL:

La primera parte se realiza como en la forma habitual en un ABB.

Luego se procede a recalcular los valores de los FB.

El nodo más próximo a aquel que con su alta produjo el desbalanceo, se llama pivote. Las rotaciones a realizar son reasignaciones en función de las características del esquema en el cual ha quedado el pivote y el nuevo nodo.

Las rotaciones posibles son:

Rotaciones simples:

Rotación Simple a derecha (DD o RR)

Rotación Simple a izquierda (II o LL)

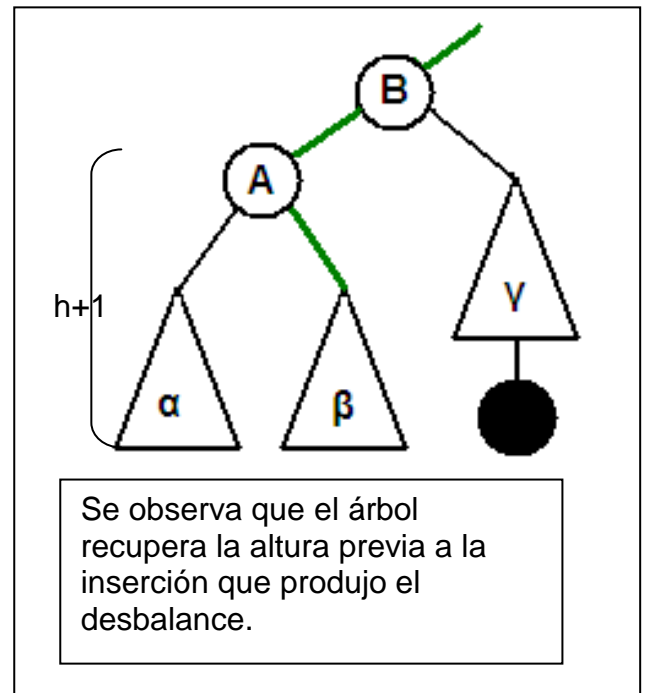
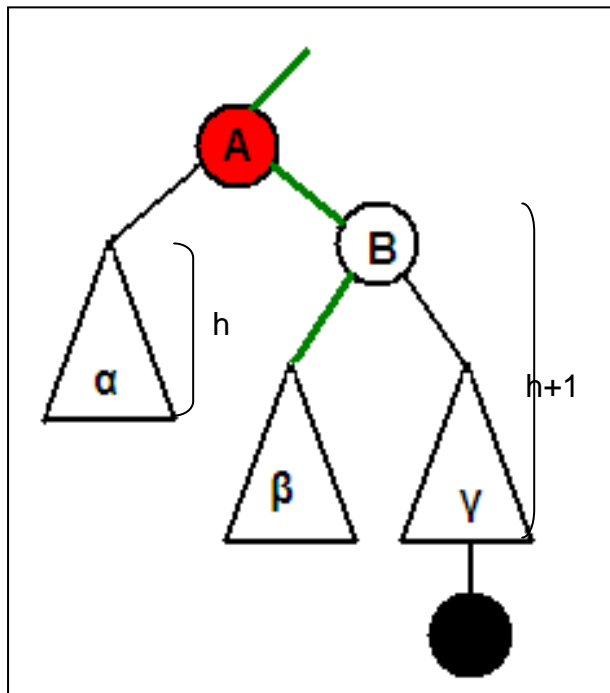
Rotaciones dobles

Rotación Doble derecha izquierda (DI o RL o RDI)

Rotación Doble izquierdaderecha (ID o LR o RID)

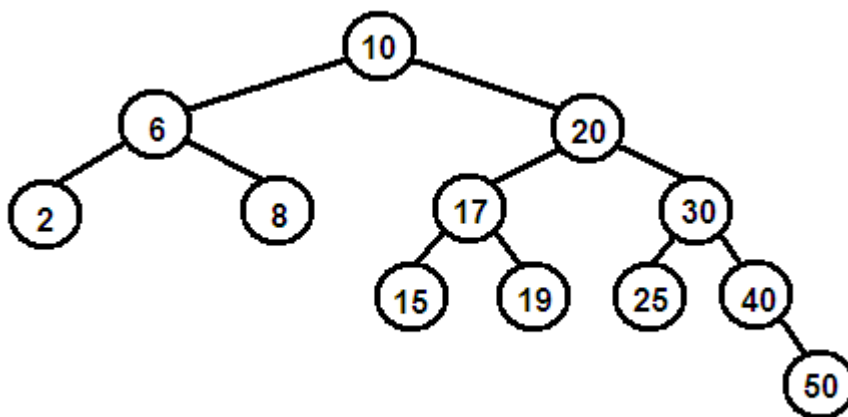
Rotacion simple a derecha (DD o RR)

En los siguientes gráficos se presenta la situación de desbalanceo y su resolución. En rojo se ha coloreado el nodo pivote. En negro, el nodo que produjo el desbalance. En verde, los punteros involucrados en el rebalanceo.

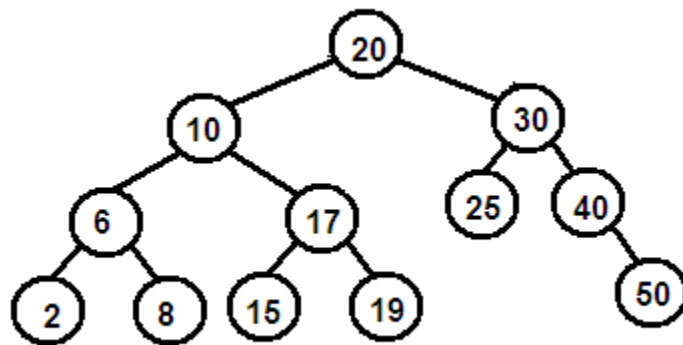


Ejemplo:

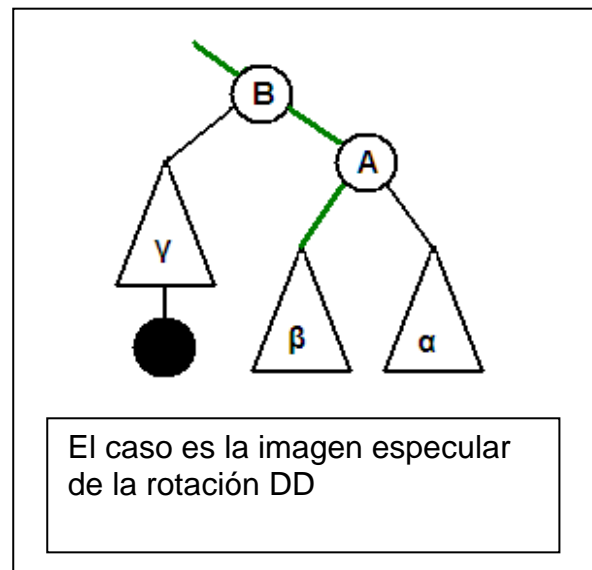
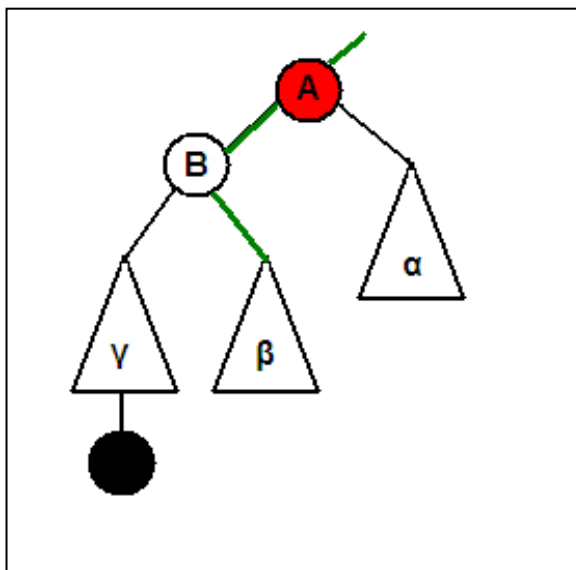
En el siguiente árbol, la inserción del 50 produce un desbalanceo.



El rebalanceo con DD da como resultado este árbol:



Caso II o LL o RSI:



En las rotaciones simples la altura del árbol vuelve a ser la que tenía antes de la inserción.

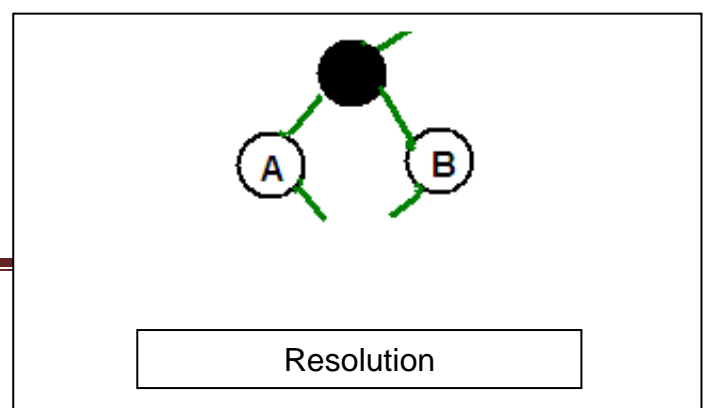
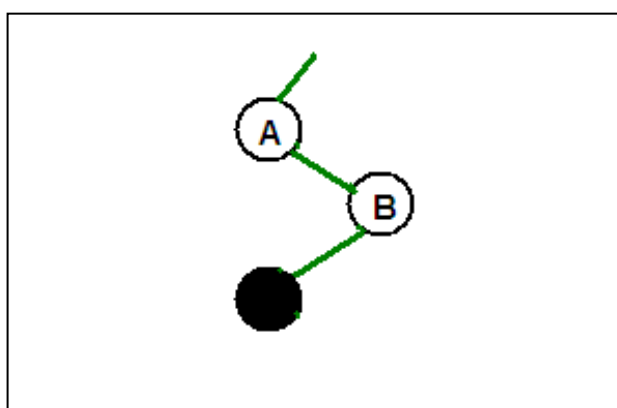
La búsqueda del pivote se hace en el subárbol que se desequilibra siguiendo el camino a la raíz.

Rotaciones dobles:

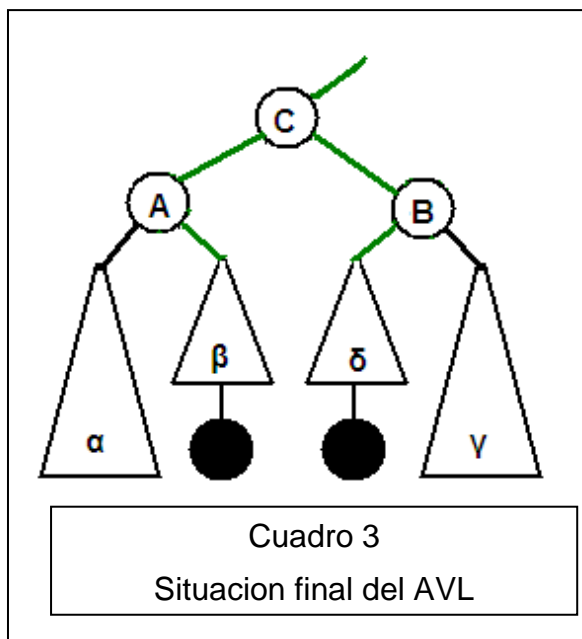
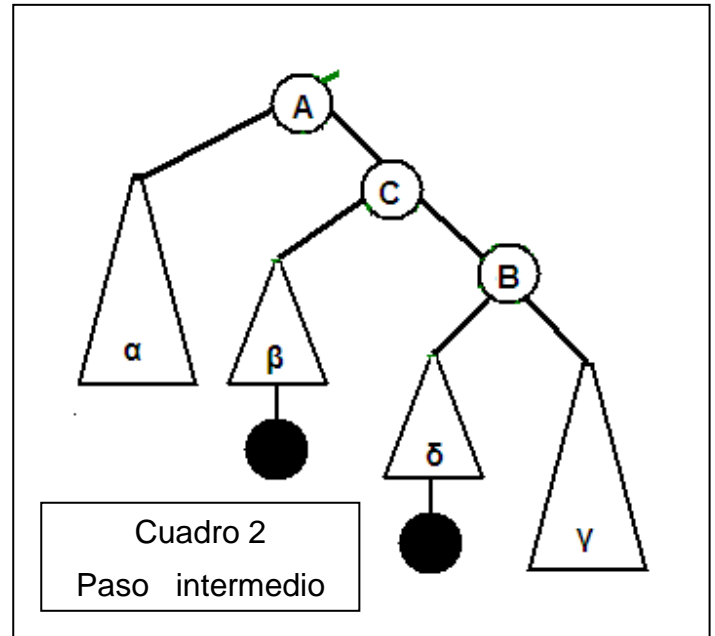
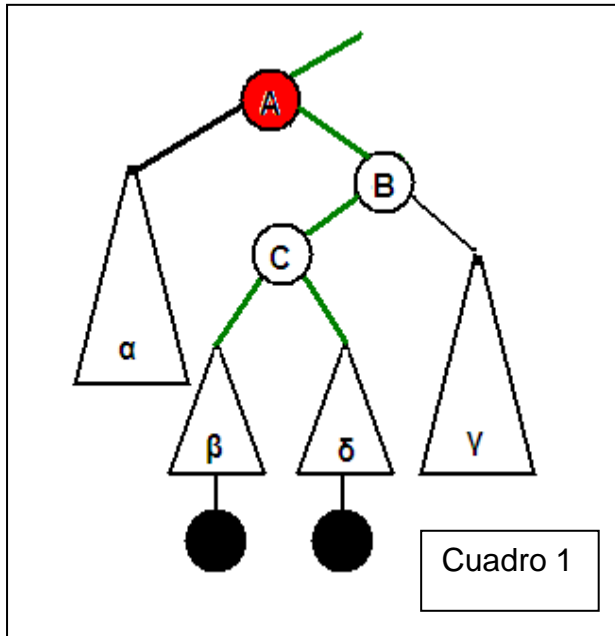
Caso DI o RL o RDI:

Las rotaciones dobles tienen dos variantes: la trivial y la no trivial, y la primera forma no puede reducirse a la segunda.

Caso DI trivial:



Caso DI no trivial:



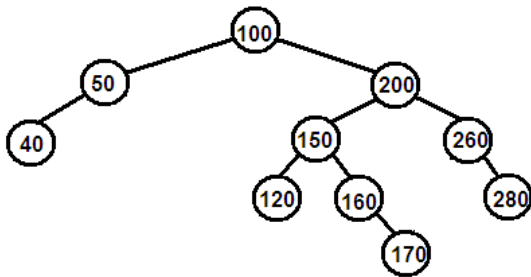
En el cuadro 1 se observa la situación que produce el desbalanceo. La misma se da por la inserción de un nodo en cualquiera de las posiciones indicadas en negro.

El cuadro 2 muestra el paso intermedio, el cuadro 3 la situación final.

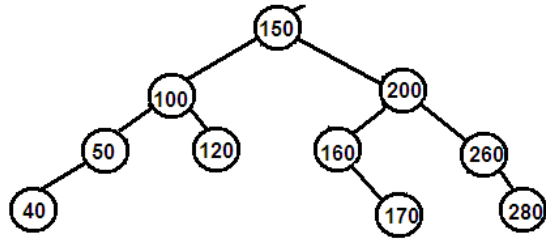
El pasaje del cuadro 1 al cuadro 2 y del cuadro 2 al cuadro 3 implica rotaciones simples.

La primera es una II y la segunda es una DD.

Ejemplo:



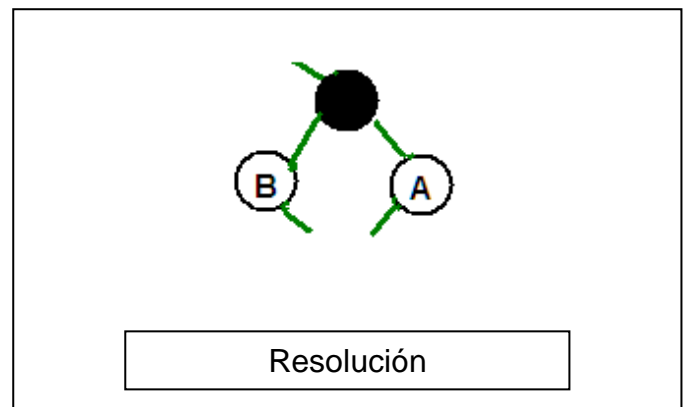
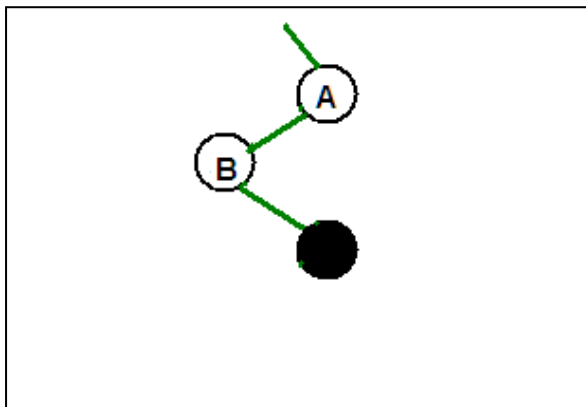
La inserción del 170 produce un desbalanceo. El pivote es 100



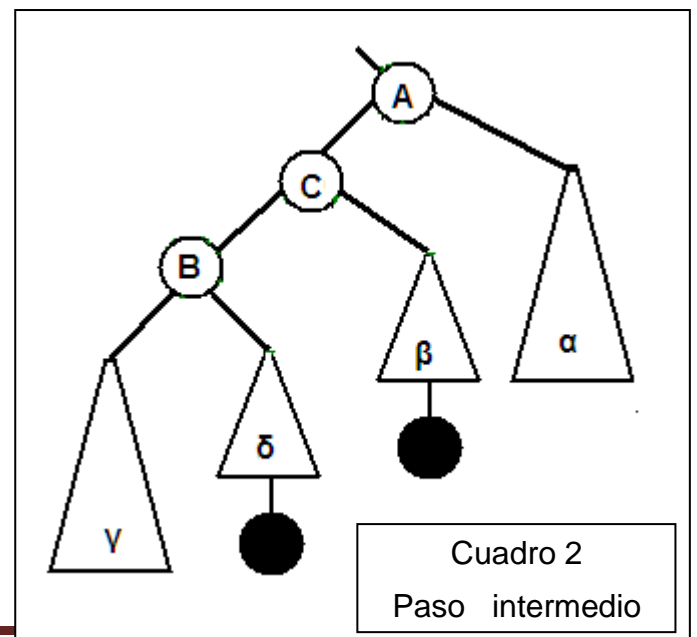
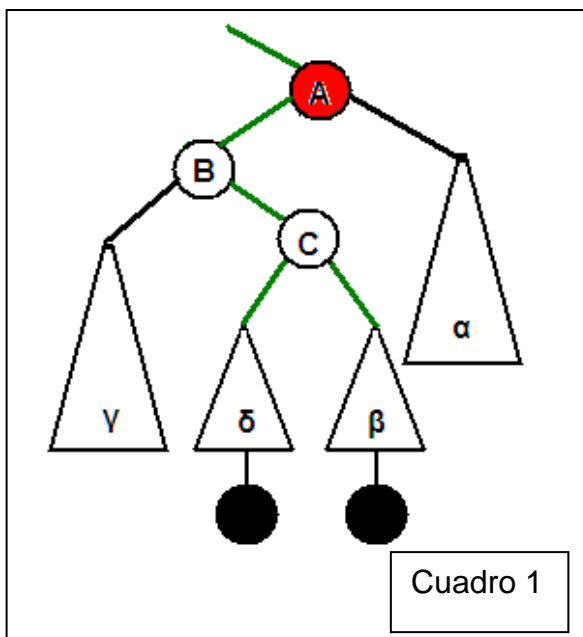
Resolución. El árbol ahora verifica la condición de balanceo.

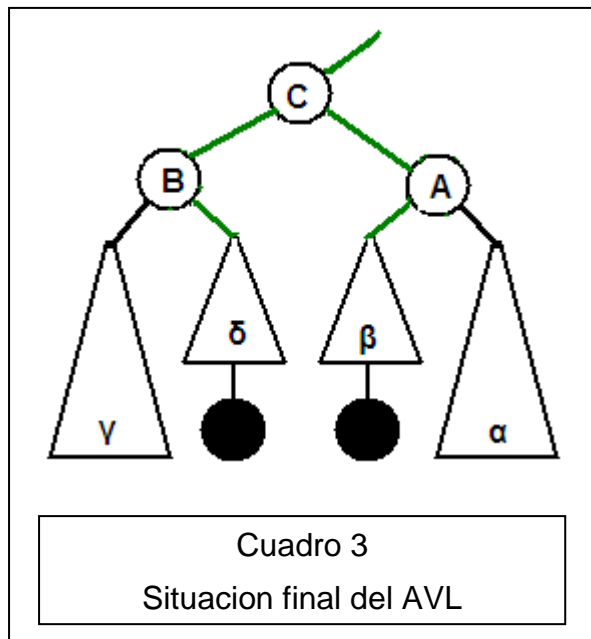
Caso rotacion doble ID

Caso ID trivial:



Caso ID no trivial





En el cuadro 1 se observa la situación que produce el desbalanceo. La misma se da por la inserción de un nodo en cualquiera de las posiciones indicadas en negro.

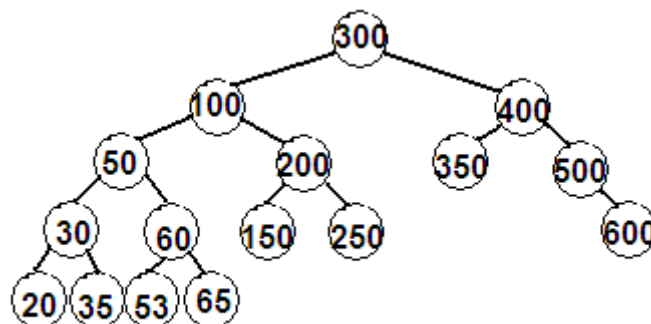
El cuadro 2 muestra el paso intermedio, el cuadro 3 la situación final.

El pasaje del cuadro 1 al cuadro 2 y del cuadro 2 al cuadro 3 implican rotaciones simples.

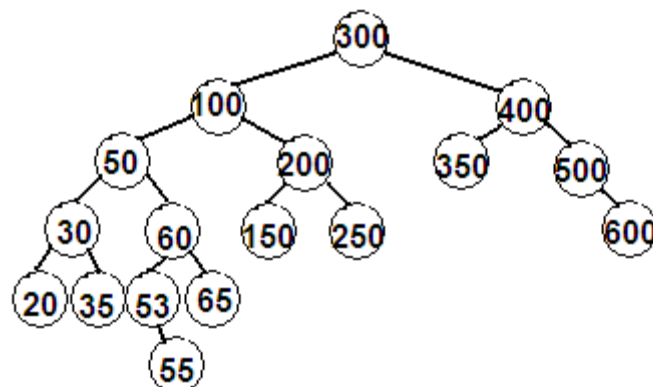
La situación es una imagen especular de las vistas en la DI

Ejemplo:

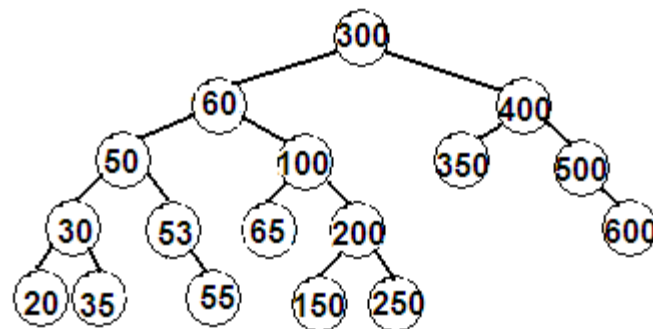
En el siguiente árbol



Se agrega un nodo con clave 55 y se produce un desbalanceo, con pivote en 100



El rebalanceo se lleva a cabo con una rotación doble ID; el árbol queda así:



Borrados en un AVL

Cuando se lleva a cabo un borrado, el desequilibrio se produce cuando la eliminación de un nodo trae como consecuencia el no cumplimiento de la propiedad del AVL.

En principio, el borrado en un AVL se lleva a cabo como en cualquier ABB; y luego se considera lo siguiente:

Si la supresión es en el subárbol izquierdo del pivote, entonces debe analizarse la situación del subárbol derecho, ya que los algoritmos de rotación que deban realizarse dependen de la relación entre las alturas de los subárboles del subárbol derecho (las cuales no pueden ser 0).

Análogo razonamiento se hace para un borrado en el subárbol derecho del pivote, pero considerando el subárbol izquierdo.

Este es el análisis y las rotaciones a realizar en cada caso:

derecho

Desequilibrio por supresión en el subárbol izquierdo del pivote:

izquierdo

Los algoritmos dependen de la relación entre las alturas del subárbol **derecho**

izquierdo

del pivote (el subárbol **derecho** no puede ser 0).

izquierdo

Relación posible entre las alturas del subárbol derecho:

izquierdo

Alturas iguales entre los subárboles del subárbol derecho: realizar

//

rotación **DD**.

En este caso, el árbol resultante la misma altura que antes de la rotación.

Alturas distintas:

izquierdo

Analizar relación entre las alturas de los subárboles del subárbol **derecho**.

derecho

izquierdo

Altura del subarbol izquierdo menor que altura del subárbol derecho:

//

realizar rotación **DD**.

El árbol resultante tiene una altura inferior en 1 a la altura previa al borrado; se vuelve necesario examinar los subárboles englobados para detectar y corregir eventuales desbalances.

derecho

izquierdo

Altura del subárbol izquierdo mayor que altura del subárbol derecho:

ID

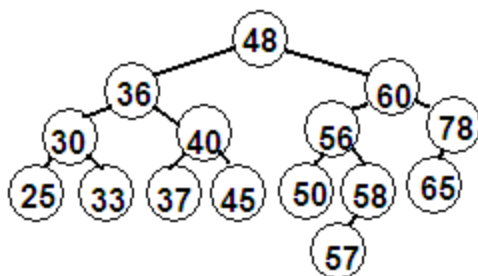
realizar rotación **DI**.

El árbol resultante tiene una altura inferior en 1 a la altura previa al borrado; se vuelve necesario examinar los subárboles englobados para detectar y corregir eventuales desbalances.

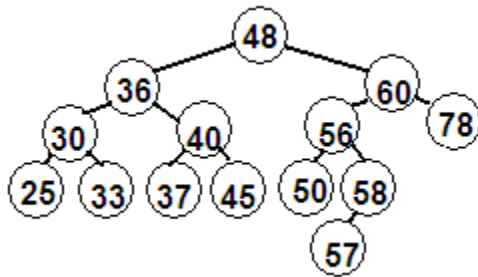
Los estudios empíricos indican que, en promedio se realiza una rotación cada 2 altas y una rotación cada 5 supresiones, y que ambas rotaciones (simples y dobles) son equiprobables.

Ejemplo:

En este árbol

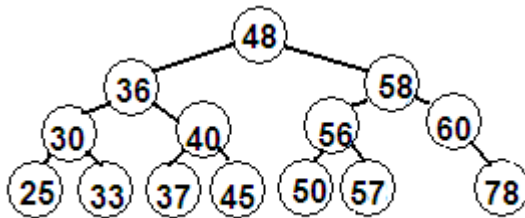


se realiza el borrado del nodo que contiene 65; queda



Se observa un desbalanceo; el pivote es 60.

El borrado se ha hecho en el subárbol derecho del pivote; el subárbol izquierdo del mismo contiene a su vez, dos subárboles, siendo el izquierdo de menor altura que el derecho; se debe aplicar una rotación doble I D, quedando el árbol con la siguiente forma:



Revisados los subárboles englobados, se determina que están equilibrados.

Es eficiente un árbol AVL?

La intención de las estructuras de árbol balanceadas por la altura, como los árboles AVL, es disminuir la complejidad temporal de las operaciones de alta, baja y búsqueda en un ABB.







La complejidad de esas tres operaciones depende de la altura del árbol; si el árbol crece desbalanceado, en el peor caso se trata de $O(N)$, esto significa que se darán hasta N pasos, o que se cumplirán N etapas, cada una de las cuales corresponde a lo que se debe realizar en un nivel del ABB. En cambio, si el árbol está completamente balanceado, será de $O(\log N)$, (se puede pensar que el número de pasos será directamente proporcional a $\log N$).

Cuanto pueden alejarse los árboles AVL de la situación ideal de altura $\log N$?

Para responder esta pregunta, consideremos árboles AVL lo más ralos que sea posible, es decir, que, para una altura determinada tengan la menor

cantidad de nodos que sea posible. A continuación, una tabla que nos muestra que ocurre:

(los dibujos son orientativos, la forma del árbol AVL puede variar un poco, pero el número de nodos, no)

Altura	1	2	3	4	5	6
Número de nodos	1	2	4	7	12	20
Grafico						

Se observa una relación entre la cantidad mínima de nodos para un árbol AVL de determinada altura y las cantidades correspondientes a los precedentes en altura.

Si llamamos $F(h)$ a la menor cantidad de nodos para un árbol AVL de altura h , observamos que se verifica:

$$F(0)=0$$

$$F(1)=1$$

$$F(h) = F(h-1) + F(h-2) + 1$$

Esta relación tiene alguna similitud con la sucesión de Fibonacci. Por eso, **a estos árboles se los llama árboles de Fibonacci.**

Un árbol de Fibonacci es un árbol AVL con la cantidad de nodos mínima.

Se demuestra que, si $F(h)=N$, entonces

h es aproximadamente $1.44 * (\log N)$

Estos árboles representan la mayor 'deformación posible' de un AVL, en el sentido de que 'desperdician' la mayor cantidad de espacio disponible en los niveles.

Entonces, un árbol AVL tiene una altura que, **a lo sumo** (si es árbol de Fibonacci) es de $1.44 * \log N$

Significa que la variación que puede tener la altura de un AVL con respecto a la 'altura ideal' de un árbol binario de búsqueda con N nodos, es de menos del 45%, lo cual indica que la complejidad del alta, la baja y la búsqueda en un AVL es $O(\log N)$, que es una situación optima (o dicho de otro modo, el numero de pasos que se llevan a cabo tanto en el algoritmo de alta, como en el de baja o de búsqueda, es directamente proporcional a $\log N$).