Resumen de Arboles AVL

Primer árbol binario de búsqueda equilibrado.
Se tratan de arboles binarios de búsqueda con una condición adicional de equilibrio. La profundidad del árbol sea siempre **O(log N)**. La idea más sencilla es exigir que los subárboles izquierdo y derecho tengan la misma profundidad.

Propiedades

Un árbol AVL es un árbol binario de búsqueda con una propiedad adicional de equilibrio, según la cual, las alturas de los hijos derecho y izquierdo solo pueden diferir, a lo sumo, en una unidad. Como es usual, tomamos -1 como la altura del árbol vacío.

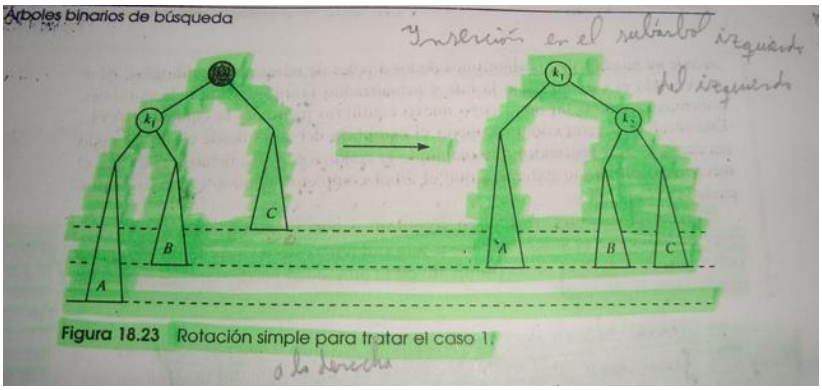
La condición de equilibrio AVL implica que el árbol tiene siempre una profundidad algorítmica. Todas las operaciones de búsqueda de un árbol AVL tienen cota algorítmica en el caso peor. la dificultad estriba en que las operaciones que modifican el árbol, como insertar y eliminar puede destruir el equilibrio de varios nodos del árbol. Se debe recuperar en el equilibrio del árbol antes de considerar finalizada la operación.
Una observación clave es que tras una inserción, solo los nodos que se encuentran en el camino desde el punto de inserción hasta la raíz pueden ver alterado su equilibrio, ya que solo se modifican subárboles de dichos nodos.

Supongamos que el nodo cuyo equilibrio debemos ajustar es X. Como cualquier nodo tiene a lo sumo dos hijos, y la diferencia de las profundidades de los subárboles de X es 2, el incumplimiento de la propiedad puede darse en uno de los cuatro casos siguientes:

1. Una inserción en el subárbol izquierdo del hijo izquierdo de X (rotación simple)
2. Una inserción en el subárbol derecho del hijo izquierdo de X (rotación doble)
3. Una inserción en el subárbol izquierdo del hijo derecho de X (rotación doble)
4. Una inserción en el subárbol derecho del hijo derecho de X (rotación simple)

Los casos **1** y **4** son simétricos respecto a **X**, al igual que los casos **2** y **3**.
El equilibrio se recupera mediante rotaciones del árbol. Una rotación simple intercambia los papeles de los padres y los hijos manteniendo la ordenación del árbol.

Rotación simple



El nodo **k2** incumple la propiedad de equilibrio AVL ya que su subárbol izquierdo es dos niveles más profundo que su subárbol derecho (las líneas punteadas marcan los niveles).
La rotación simple resuelve los casos extremos (caso **1** y **4**). Realizamos la rotación entre los nodos y su hijo. el resultado es un árbol binario de búsqueda que satisface la propiedad AVL.
Para equilibrar de forma correcta el árbol, queremos subir **A** un nivel y bajar **C** otro tanto.
Queda ilustrado por el siguiente razonamiento abstracto: consideremos el árbol como una estructura flexible, ponemos un peso en el nodo **k2,** cerramos los ojos; y dejamos que actué la gravedad reequilibrando la balanza. El resultado es que **k1** será la nueva raíz.
La propiedad de los arboles binarios de búsqueda indica que en el árbol inicial, **k2** > **k1**, así que en el nuevo árbol **k2** se convierte en el hijo derecho de **k1**.
A sube un nivel, **B** permanece en el mismo y **C** desciende un nivel. **k1** y **k2** no solo satisfacen la propiedad AVL, sino que además sus subárboles son de la misma altura. Más aún, la nueva altura del subárbol completo es *exactamente la misma* que la altura del subárbol previo a la inserción, que ha provocado el crecimiento de **A**.

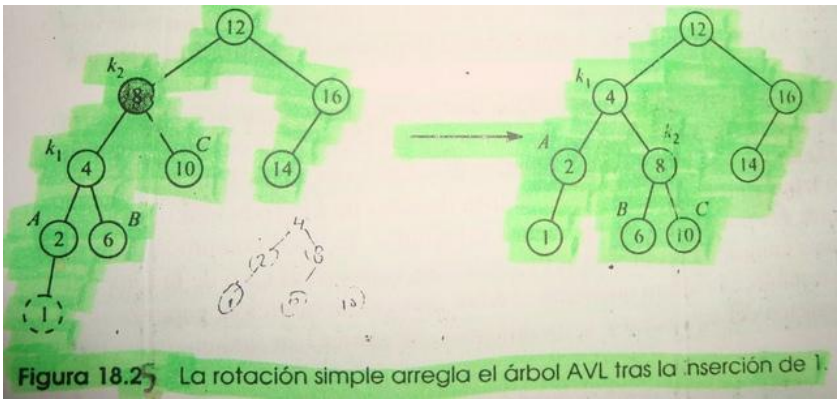


Figura 18.25 La rotación simple arregla el árbol AVL tras la inserción de 1.

En la Figura 18.25 muestra que después de la inserción de 1 en el árbol AVL, el nodo 8 esta desequilibrado. Claramente estamos en el primer caso, ya que 1 está en el subárbol izquierdo del hijo izquierdo de 8. En consecuencia estamos realizando una rotación simple entre 8 y 4, obteniendo se el árbol de la derecha.
Se menciono anteriormente que le caso 4 es simétrico a éste.

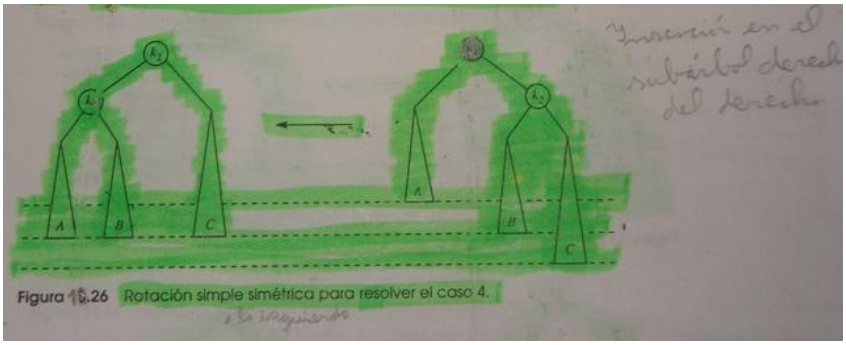


Figura 18.26 Rotación simple simétrica para resolver el caso 4.

Una rotación es suficiente para resolver los casos **1** y **4** en los arboles AVL.

Rotación doble

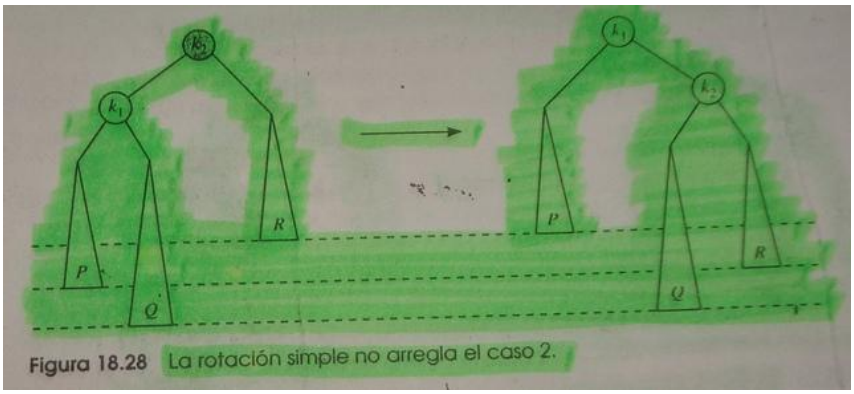


Figura 18.28 La rotación simple no arregla el caso 2.

La rotación simple no resuelve los casos interiores (**2** y **3**). Estos casos requieren una rotación doble, que manipula tres nodos y cuatro subárboles.

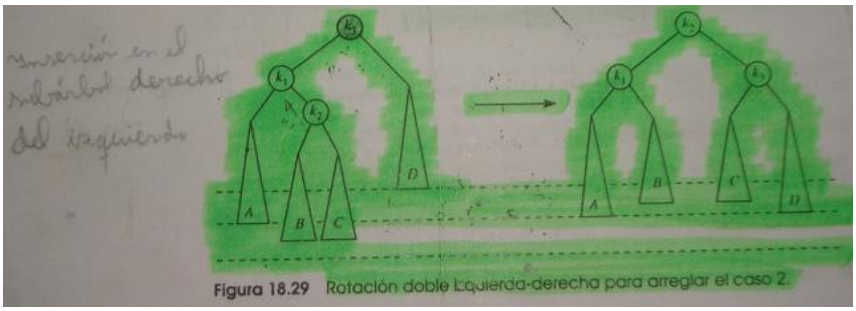


Figura 18.29 Rotación doble izquierda-derecha para arreglar el caso 2.

Exactamente uno de los arboles **B** o **C** es dos niveles más profundo que **D**, aunque no podemos asegurar cuál de ellos es. Esto carece de importancia por lo que, en la figura, tanto **B** como **C** se han dibujado 1,5 niveles por debajo de **D**.
Para recuperar el equilibrio, observemos que es imposible dejar **k3** como raíz, y además que la rotación entre **k3** y **k1** no funciona. La única alternativa es colocar a **k2** como nueva raíz. Esto fuerza a **k1** a ser el hijo izquierdo de **k2** y a **k3** a ser el hijo derecho de **k2**.
Como en el caso de la rotación simple, esto restablece la altura que tenía el árbol antes de la inserción, garantizando que la recuperación del equilibrio y la molificación de alturas han sido completadas.

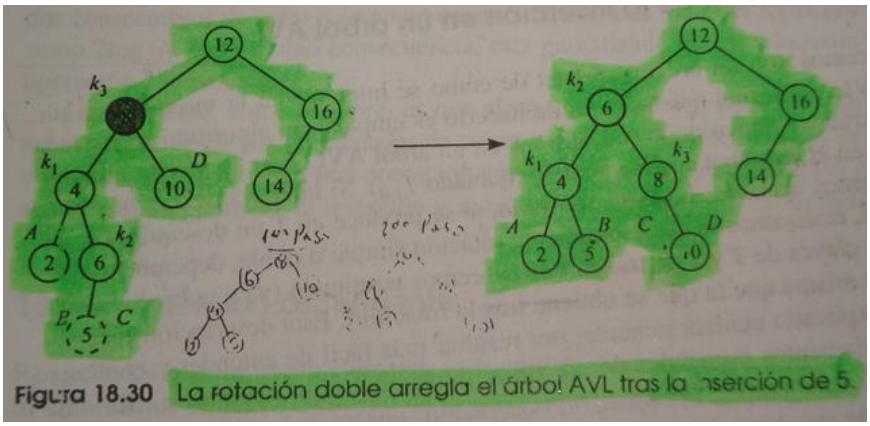
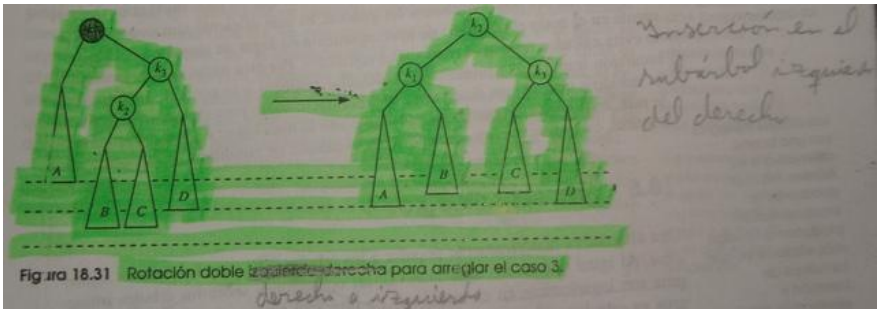


Figura 18.30 La rotación doble arregla el árbol AVL tras la inserción de 5.

Muestra el resultado de insertar 5 en un árbol AVL. El desequilibrio de las alturas se produce en el nodo 8, por lo que estamos en el caso 2. Realizamos una rotación doble en ese nodo, generando el árbol de la derecha.



Muestra el caso simétrico 3, que también puede ser arreglado con una rotación doble.

OBS:
Nótese que aunque la rotación doble parezca compleja, es equivalente a hacer lo siguiente:

- Una rotación simple entre el hijo de X y su nieto, seguida de
- una rotación simple entre X y su nuevo hijo.

Resumen de la inserción en un árbol AVL
La forma más sencilla de hacerlo es emplear un algoritmo recursivo.
Para insertar un nuevo nodo con clave X en un árbol AVL **T**, se le inserta recursivamente en el subárbol adecuado de **T** (llamado $\mathbf{T_{1R}}$). Si la altura de $\mathbf{T1_R}$ no cambia, ya hemos acabado. En caso contrario, si se produce en **T** un desequilibrio de las alturas, realizamos la correspondiente rotación simple o doble, dependiendo de **X** y de las claves de **T** y $\mathbf{T_{LR}}$, tras lo que habremos terminado (ya que la altura original es la misma que la que se obtiene tras la rotación).

Una implementación directa del criterio que gobierna los arboles AVL resulta bastante sencilla, aunque no es demasiado eficiente. Han sido descubiertos otros tipos de arboles de búsqueda equilibrados mejores, por lo que en la práctica no vale la pena implementar los arboles AVL.

Fuente: Estructuras de datos en Java - Mark Allen Weiss
Blog: http://proyectosbeta.blogspot.com/
Mail: josego85@gmail.com o al proyectosbeta@gmail.com