

Universidad CENFOTEC

Escuela de Software

Curso: SOFT-04 – Estructuras de Datos

Sección: SCV2

Docente: Romario Salas Cerdas

Estudiante: Santiago Hernandez

Proyecto: Segundo Estudio de Caso - Árboles AVL

Fecha: 30 de Noviembre del 2025

¿Qué es un Árbol AVL?

Un árbol AVL es una estructura de datos que pertenece a la familia de los árboles binarios de búsqueda autobalanceados. Su objetivo principal es garantizar que las operaciones de búsqueda, inserción y eliminación mantengan un tiempo de ejecución óptimo, incluso cuando el conjunto de datos crece considerablemente.

Fue propuesto en 1962 por Adelson-Velsky y Landis, quienes establecieron las reglas matemáticas que permiten detectar desbalances y corregirlos automáticamente mediante rotaciones.

En un árbol binario de búsqueda tradicional, el rendimiento puede deteriorarse cuando los datos se insertan en un orden desfavorable, provocando que el árbol se convierta en una estructura degenerada semejante a una lista. Los árboles AVL resuelven este problema imponiendo una restricción fundamental: la diferencia entre las alturas de los subárboles izquierdo y derecho de cada nodo debe ser -1 , 0 o 1 .

Esta diferencia recibe el nombre de factor de balance.

¿Qué es el factor de balance?

El factor de balance (FB) es la métrica utilizada para determinar si un nodo y, por extensión, el árbol completo están equilibrados.

Se define como:

$FB(\text{nodo}) = \text{altura del subárbol izquierdo} - \text{altura del subárbol derecho}$

$FB = 0 \rightarrow$ equilibrio perfecto

$FB = 1 \rightarrow$ ligero inclinado a la izquierda, pero válido

$FB = -1 \rightarrow$ ligero inclinado a la derecha, pero válido

$FB = 2$ o $-2 \rightarrow$ desbalance; requiere rotación

Cada vez que se inserta o elimina un nodo, el árbol actualiza las alturas y recalcula los factores de balance a lo largo del camino desde el nodo modificado hasta la raíz.

Si alguno de los nodos presenta un FB absoluto mayor a 1, se ejecuta una de las rotaciones disponibles.

¿Cómo se detecta un desbalance?

Un árbol AVL recorre la ruta desde el nodo insertado/eliminado hasta la raíz.

Durante este recorrido:

1. Actualiza la altura del nodo.
2. Recalcula el factor de balance.
3. Identifica el primer nodo cuyo FB sea 2 o -2.
4. Determina qué tipo de rotación aplicar según:
 - si el desbalance está a la izquierda o derecha,
 - y si el subárbol interno creció hacia la derecha o hacia la izquierda.

Los cuatro casos principales son:

Tipo de Desbalance	Causa	Rotación requerida
Izquierda-Izquierda (II)	Inserción en subárbol izquierdo del hijo izquierdo	Rotación derecha simple
Derecha-Derecha (DD)	Inserción en subárbol derecho del hijo derecho	Rotación izquierda simple
Izquierda-Derecha (ID)	Inserción en subárbol derecho del hijo izquierdo	Rotación doble izquierda-derecha
Derecha-Izquierda (DI)	Inserción en subárbol izquierdo del hijo derecho	Rotación doble izquierda-derecha

Aplicaciones reales de los árboles AVL

Los árboles AVL se utilizan en escenarios donde el acceso rápido y consistente es esencial. Algunas aplicaciones comunes son:

1. Bases de datos y sistemas de indexación

Los motores de bases de datos que mantienen índices en memoria requieren estructuras balanceadas para garantizar búsquedas constantes. Los AVL aseguran tiempos $O(\log n)$.

2. Compiladores

Las tablas de símbolos utilizadas durante la compilación se benefician de estructuras que permitan inserciones y búsquedas eficientes.

3. Sistemas de archivos

Algunos sistemas gestionan metadatos mediante árboles autobalanceados.

4. Motores de búsqueda

Cuando el contenido se indexa en memoria, un AVL permite mantener orden y obtener resultados de forma inmediata.

5. Aplicaciones en tiempo real

Cualquier software donde los tiempos de respuesta deben ser predecibles utiliza estructuras balanceadas para evitar picos de latencia.

¿Por qué los AVL son adecuados?

- Garantizan equilibrio estricto (más estricto que un Red-Black Tree).
- Mantienen la altura mínima posible.
- Reducen al máximo el peor caso.
- Funcionan muy bien en escenarios dominados por búsqueda más que por inserción masiva.

Caso 1. Rotación Simple Derecha (LL Case)

Para ejemplificar la rotación simple hacia la derecha, se utilizaron los siguientes valores en orden de inserción: 30, 20, 10. Esta secuencia provoca un desbalance Izquierda–Izquierda (LL) en el nodo 30.

Estado del árbol antes del desbalance: $30 > 20 > 10$.

Cálculo de alturas (h):

- $h(10) = 1$
- $h(20) = 2$
- $h(30) = 3$

Factor de Balance (FB)

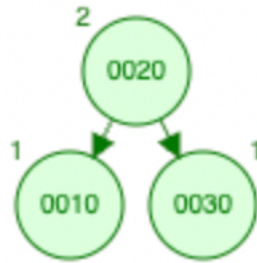
$FB(n) = altura(izq) - altura(der)$

- $FB(10) = 0$
- $FB(20) = 1$
- $FB(30) = 2 \rightarrow$ desbalance detectado

El FB de 30 es 2 y el de 20 es 1, por lo que el caso se clasifica como: Caso LL \rightarrow Rotación Simple Derecha.

Para corregir el desbalance, el nodo 20 se convierte en la nueva raíz del subárbol, 30 pasa a ser su hijo derecho, y 10 permanece como hijo izquierdo de 20.

Estado del árbol después de la rotación



Conclusión de la rotación LL:

La rotación simple hacia la derecha corrige desbalances del tipo Izquierda–Izquierda, donde:

- el nodo desbalanceado tiene $FB = +2$
- el hijo izquierdo tiene $FB \geq 0$

Este procedimiento garantiza que el árbol mantenga las propiedades AVL y reduzca su altura.

Caso 2. Rotación Doble Izquierda–Derecha (LR Case)

Para ilustrar una rotación doble en un árbol AVL, se utilizó la siguiente secuencia de inserciones: 40>20>30>25.

Esta secuencia genera un desbalance tipo Izquierda–Derecha (LR) en el nodo 40 cuando se inserta el valor 30, lo que obliga a aplicar dos rotaciones consecutivas: primero una rotación simple izquierda, seguida de una rotación simple derecha.

Cálculo de alturas: (antes del ajuste final)

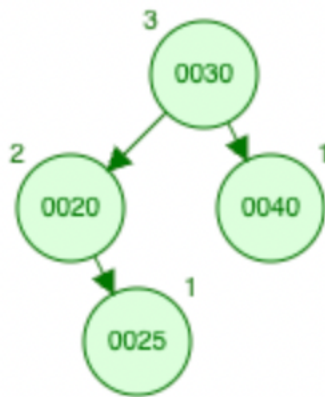
- $h(25) = 1$ (aún no ha sido añadido)
- $h(20) = 2$
- $h(40) = 1$
- $h(30) = 3$

Factores de balance:

- $FB(20) = 0 - 1 = -1$
- $FB(40) = 0$
- $FB(30) = 2 - 1 = +1$

Todos los factores de balance están dentro del rango permitido, así que el árbol se mantiene perfectamente balanceado.

Estado del árbol después de la rotación



Conclusión del caso LR:

El caso Izquierda–Derecha (LR) se produce cuando:

- El nodo desbalanceado posee $FB = +2$
- El hijo izquierdo posee FB negativo

La solución requiere dos rotaciones:

1. Rotación simple izquierda sobre el hijo izquierdo
2. Rotación simple derecha sobre el nodo desbalanceado

El resultado final es un árbol AVL correctamente balanceado, con alturas mínimas y cumpliendo las propiedades de búsqueda y balance.

Repositorio de GitHub: <https://github.com/sant1agozs/EDEstudioDeCaso2>