Segundo Estudio de Caso

Curso: Estructuras de Datos

Código: SOFT-10
Periodo: 2025-C3

Estudiante:

Andreina Leal Vega

Profesor:

Romario Salas Cerdas

Fecha de entrega:

30 de noviembre del 2025

Introducción

Los árboles AVL son una de las estructuras de datos auto balanceadas más utilizadas en informática debido a su capacidad para mantener operaciones eficientes sin importar el orden en que los datos ingresen. A diferencia de un árbol binario de búsqueda (ABB) normal, en el que la inserción de datos desordenados puede producir estructuras degeneradas similares a una lista enlazada, los árboles AVL garantizan un equilibrio constante mediante el cálculo del factor de balance y el uso de rotaciones, optimizando las operaciones de búsqueda, inserción y eliminación.

Este estudio de caso presenta una investigación completa sobre el funcionamiento del factor de balance, los criterios para detectar desbalances y las rotaciones necesarias para corregirlos. Además, se incluyen las aplicaciones más relevantes de los árboles AVL y la justificación de por qué esta estructura es adecuada para dichas tareas.

Factor de balance en un árbol AVL

¿Qué es el factor de balance?

El factor de balance (FB) es una medida utilizada en los árboles AVL para determinar si un nodo está equilibrado o desbalanceado. Su fórmula es:

$FB = altura\_subarbol\_izquierdo - altura\_subarbol\_derecho$

Un árbol AVL está balanceado si, para cada nodo:

$-1 \leq FB \leq 1$

Cuando el factor de balance de un nodo cae fuera de ese rango, el árbol necesita ser ajustado mediante rotaciones.

Cálculo de alturas

La altura de un nodo se define como:

- 0 si el nodo es una hoja.
- 1 + altura del subárbol más alto, para nodos internos.

Después de cada operación (insertar o eliminar), se deben actualizar las alturas de los nodos afectados, desde abajo hacia arriba.

Ejemplo gráfico del cálculo del factor de balance

Inserción de valores: 30, 20, 10

Paso 1: Insertamos 30
Árbol:

```
30
```

FB(30) = 0

Paso 2: Insertamos 20

```
  30
 /
20
```

FB(30) = 1
FB(20) = 0

Paso 3: Insertamos 10

```
    30
   /
  20
 /
10
```

FB(30) = 2 (desbalanceado)
FB(20) = 1
FB(10) = 0

Este es un caso **Izquierda–Izquierda (LL)** → requiere **rotación simple a la derecha**.

Rotaciones en árboles AVL

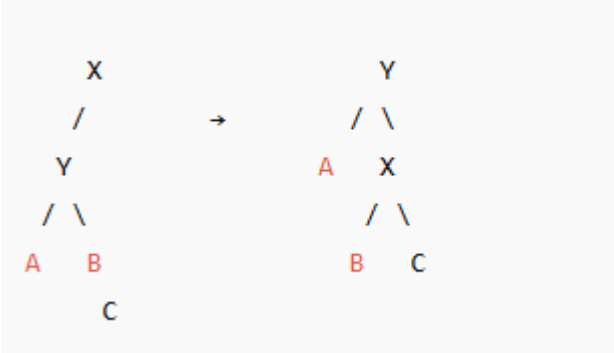
Las rotaciones son operaciones que reorganizan los nodos para restaurar el equilibrio. Existen cuatro tipos:

1. Rotación simple a la derecha
2. Rotación simple a la izquierda
3. Rotación doble izquierda–derecha
4. Rotación doble derecha–izquierda

Todas reciben un nodo desbalanceado y devuelven la nueva raíz del subárbol corregido.

Rotación simple a la derecha (LL)

Se usa cuando el desbalance se produce en el subárbol izquierdo del hijo izquierdo.

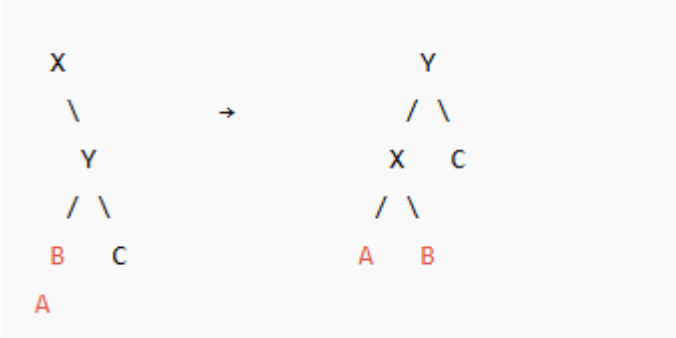


Proceso:

1. Y pasa a ser la nueva raíz.
2. X se convierte en el hijo derecho de Y.
3. B pasa a ser el hijo izquierdo de X.

Rotación simple a la izquierda (RR)

Se usa cuando el desbalance se produce en el *subárbol derecho del hijo derecho*.



Rotación doble izquierda–derecha (LR)

Se usa cuando el desbalance se produce en el *subárbol derecho del hijo izquierdo*.

Es equivalente a:

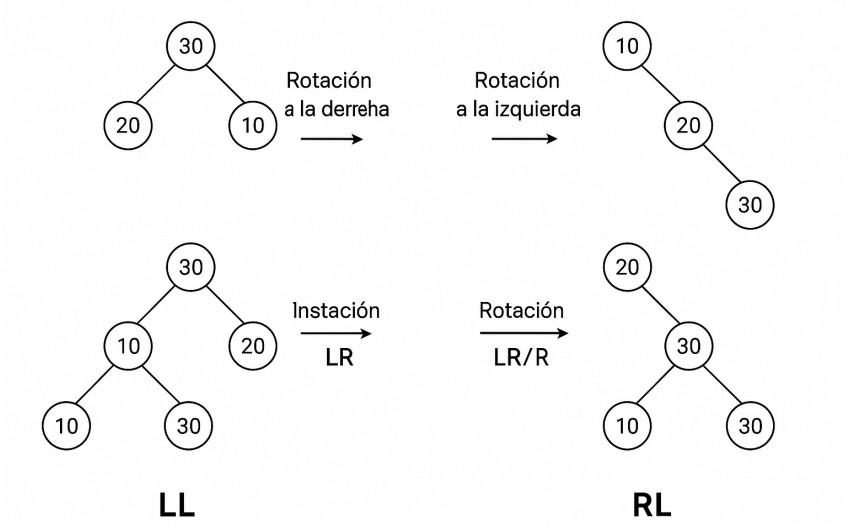
1. Rotación simple a la izquierda en el hijo izquierdo.
2. Rotación simple a la derecha en el nodo desbalanceado.

Rotación doble derecha–izquierda (RL)

Se usa cuando el desbalance ocurre en el *subárbol izquierdo del hijo derecho*.

Se compone de:

1. Rotación simple a la derecha en el hijo derecho.
2. Rotación simple a la izquierda en el nodo desbalanceado.



Aplicaciones de los árboles AVL

Los árboles AVL son altamente eficientes, con tiempos garantizados de:

- **Búsqueda: O(log n)**
- **Inserción: O(log n)**
- **Eliminación: O(log n)**

Gracias a su balanceo constante, se utilizan en sistemas donde la rapidez y estabilidad son indispensables.

Bases de datos

Los índices de ciertas bases de datos utilizan estructuras similares a los AVL para garantizar búsquedas rápidas aunque los datos estén desordenados.

Por qué AVL es adecuado:
– Mantiene acceso eficiente sin importar el orden de inserción.
– Evita la degeneración típica de un ABB normal.

Compiladores

Los compiladores utilizan árboles para almacenar tablas de símbolos, identificadores y estructuras semánticas.

Razón:
– Las consultas son constantes a medida que se corre el análisis sintáctico.
– Un ABB normal podría volverse lento, pero un AVL mantiene rendimiento predecible.

Sistemas de archivos

Muchos sistemas de archivos necesitan búsquedas rápidas para ubicar archivos por nombre o por hash.

AVL garantiza:
– tiempos estables,
– estructura eficiente para búsquedas secuenciales (recorridos en orden).

Motores de videojuegos

En motores gráficos se usan AVL para:

– manejo de entidades,
– colisiones,
– activación de objetos.

Son útiles porque permiten reorganizar datos dinámicamente sin perder eficiencia.

Conclusiones

Los árboles AVL representan una solución elegante y robusta para mantener datos ordenados de forma eficiente. Su capacidad de mantener el balance por medio del cálculo de alturas, factores de balance y rotaciones los convierte en una herramienta esencial en sistemas donde el rendimiento constante es crítico.

Además, sus aplicaciones en bases de datos, compiladores, sistemas de archivos y videojuegos demuestran su versatilidad y relevancia en el mundo real. Esta investigación permite comprender no solo su estructura y funcionamiento, sino también su importancia práctica y su ventaja frente a estructuras no balanceadas.