Árboles AVL: Funcionamiento del Factor de Balance, Algoritmos de Rotación y Aplicaciones

1. Introducción

Los árboles AVL son estructuras de datos de tipo árbol binario de búsqueda que se caracterizan por mantenerse equilibrados de manera automática. Fueron introducidos por **Georgy Adelson-Velsky y Evgenii Landis** en 1962 y se consideran uno de los primeros árboles de búsqueda balanceados.

El propósito de un árbol AVL es garantizar que las operaciones de búsqueda, inserción y eliminación se realicen en tiempo **O(log n)**, mediante un control estricto del balance entre las alturas de los subárboles de cada nodo.

2. Factor de Balance (FB) de los Nodos

2.1 Definición

El **factor de balance (FB)** de un nodo se define como:

FB(nodo)=Altura(Suba´rbol Izquierdo)−Altura(Suba´rbol Derecho)FB(nodo) = Altura(Subárbol\ Izquierdo) - Altura(Subárbol\ Derecho)FB(nodo)=Altura(Suba´rbol Izquierdo)−Altura(Suba´rbol Derecho)

Los árboles AVL imponen la regla:

FB(nodo)∈{−1,0,+1}FB(nodo) ∈ \{-1, 0, +1\}FB(nodo)∈{−1,0,+1}

Si un nodo presenta un FB fuera de ese rango, el árbol está desbalanceado y debe aplicarse una **rotación** para corregirlo.

2.2 Cálculo de la Altura

La altura de un nodo se define como:

- 0 → si el nodo es hoja
- 1 + máximo(altura hijo izquierdo, altura hijo derecho)

Algoritmo (pseudocódigo)

```
Altura(nodo):
  si nodo = nulo:
```

```
        retornar -1
    retornar 1 + max(Altura(nodo.izq), Altura(nodo.der))
```

2.3 Cálculo del Factor de Balance

Algoritmo (pseudocódigo)

```
FactorBalance(nodo):
    si nodo = nulo:
        retornar 0
    retornar Altura(nodo.izq) - Altura(nodo.der)
```

3. Detección de Desbalance

Un desbalance ocurre cuando:

- FB = +2 (subárbol izquierdo pesa más)
- FB = –2 (subárbol derecho pesa más)

Y cada caso tiene dos posibles subcasos, según la forma en que el nuevo nodo ingresó:

Caso	Descripción
LL	Inserción en rama izquierda del subárbol izquierdo
RR	Inserción en rama derecha del subárbol derecho
LR	Inserción en rama derecha del subárbol izquierdo
RL	Inserción en rama izquierda del subárbol derecho

4. Rotaciones AVL

Las rotaciones permiten reacomodar el árbol para devolverle el balance.

4.1 Rotación Simple a la Derecha (Caso LL)

Se aplica cuando:

- FB del nodo desbalanceado = +2
- FB del hijo izquierdo = +1

Diagrama del caso LL

```
    A (+2)
    /
   B
  /
 C
```

Después de la rotación

```
    B
   / \
  C   A
```

Algoritmo

```
RotarDerecha(A):
   B = A.izq
   A.izq = B.der
   B.der = A
   actualizar alturas(A)
   actualizar alturas(B)
   retornar B
```

4.2 Rotación Simple a la Izquierda (Caso RR)

Se aplica cuando:

- FB = –2

- FB del hijo derecho = –1

Antes

Después

4.3 Rotación Doble Izquierda-Derecha (Caso LR)

Ocurre cuando:

- FB = +2
- Hijo izquierdo tiene FB = –1

Antes

Proceso

1. Rotación simple a la izquierda en B
2. Rotación simple a la derecha en A

Después

```
 / \
B   A
```

4.4 Rotación Doble Derecha-Izquierda (Caso RL)

Ocurre cuando:

- FB = –2
- Hijo derecho tiene FB = +1

Antes

```
  A
   \
    B
   /
  C
```

Proceso

1. Rotación simple a la derecha en B
2. Rotación simple a la izquierda en A

Después

```
   C
  / \
 A   B
```

5. Paso a Paso Ejemplos Gráficos

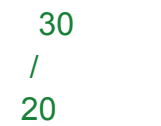
5.1 Ejemplo con desbalance LL

Insertar: 30, 20, 10

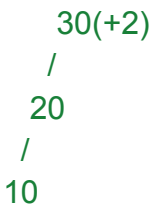
Insertar 30:

```
 30
```

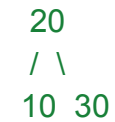
Insertar 20:



Insertar 10:



Rotación derecha → árbol balanceado



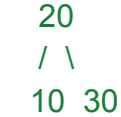
5.2 Ejemplo con desbalance LR

Insertar: 30, 10, 20

Rotación doble:

1. Rotación izquierda en 10
2. Rotación derecha en 30

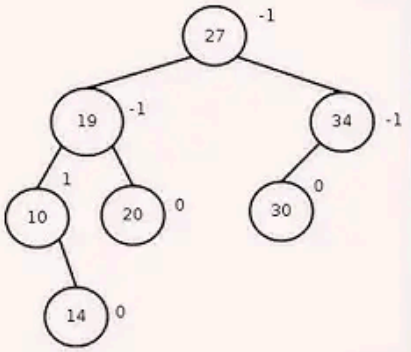
Resultado:



Factor de equilibrio

Este factor de equilibrio significa un cambio en la estructura de los arboles que usualmente usamos en computación.

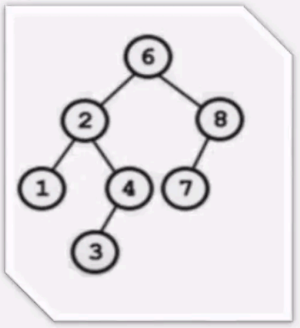
Se anexa un nuevo campo a cada nodo en el árbol, el cual indica su *factor de equilibrio*.



COSAS A TENER EN CUENTA

- Decimos que un Árbol Binario se encuentra en equilibrio si para todo Nodo la altura de sus Sub-árboles izquierdo y derecho pueden diferir 1 unidad, nombrando este valor como Factor de equilibrio(FE). La formula del Factor de equilibrio es:
- FE=Altura Sub-árbol Derecho – Altura Sub-árbol Izquierdo
- ; siendo el FE=0 Si se esta evaluando un Nodo Hoja.
- Altura= Nivel del nodo + 1
- En caso de que la Rotación Simple Derecha o Izquierda,

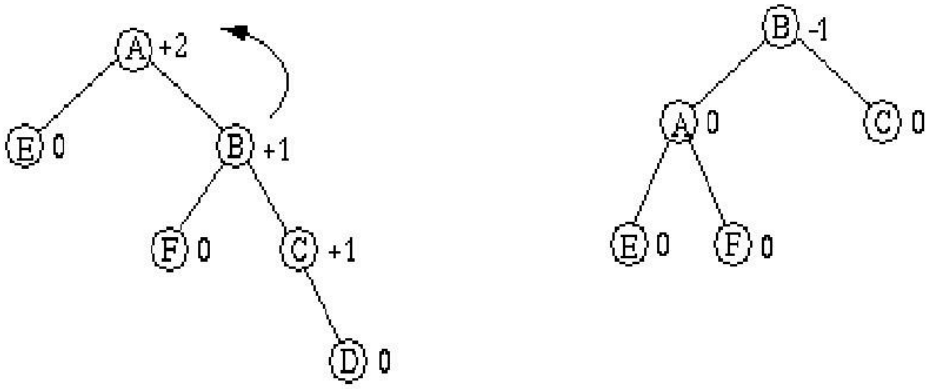
No funciones se utiliza la rotación doble



Balancear

9

- Caso 1: Rotación simple izquierda RSI
  - Si esta desequilibrado a la izquierda y su hijo derecho tiene el mismo signo (+) hacemos rotación sencilla izquierda.



6. Aplicaciones de los Árboles AVL

Los árboles AVL se usan cuando es crucial que las operaciones se mantengan cerca de tiempo **O(log n)** incluso en el peor caso.

6.1 Sistemas de Bases de Datos

Los índices de bases de datos requieren:

- Búsquedas rápidas y consistentes
- Ordenamiento interno de datos
- Alturas mínimas

Los AVL aseguran un rendimiento estable sin degradarse a listas.

6.2 Sistemas Operativos

Ejemplos:

- Gestión de memoria (buddy system)
- Planificadores
- Tablas de procesos

Se usan para mantener listas ordenadas de bloques o procesos.

6.3 Compiladores

Se emplean para:

- Tablas de símbolos
- Árboles de expresiones
- Gestión eficiente de variables

6.4 Redes y Telecomunicaciones

Enrutamiento y búsqueda rápida de:

- Direcciones
- Prefijos
- Rutas óptimas

6.5 Motores de Búsqueda y Autocompletado

Los AVL permiten:

- Búsquedas predictivas rápidas
- Mantenimiento eficiente de listas ordenadas
- Recuperación rápida de términos

6.6 Aplicaciones con Datos en Tiempo Real

Como:

- Monitoreo de sensores
- Aplicaciones financieras
- Sistemas de control

Necesitan inserciones rápidas sin perder el balance.

7. Conclusión

Los árboles AVL son estructuras sumamente eficientes cuando se necesita garantizar operaciones rápidas y estables sin importar la secuencia de inserciones. Gracias a su control rígido del factor de balance y al uso sistemático de rotaciones, mantienen una altura óptima que asegura un rendimiento consistente. Su aplicación en bases de datos, sistemas operativos, compiladores y muchos otros campos demuestra su relevancia en la informática moderna.