Lindsay Mishelle Quirós Enríquez
30 de noviembre de 2025

Caso II:

Algoritmos de Evaluación: AVL

SOFT - 10

SCV2

Profesor: Romario Salas Cerdas

Índice

Introducción

El árbol AVL es un árbol binario de búsqueda auto-balanceado introducido por Adelson-Velsky y Landis en 1962. Esta estructura garantiza que la diferencia

de alturas entre los subárboles izquierdo y derecho de cualquier nodo no exceda ±1, manteniendo así un rendimiento óptimo en búsqueda, inserción y eliminación. Gracias a este balance estricto, el árbol AVL asegura una altura O(log n), lo que resulta fundamental en aplicaciones donde el tiempo de acceso debe ser consistente.

Según Goodrich, Tamassia & Goldwasser (2014) (pp. 497–505), los árboles AVL cumplen la height-balance property, lo que asegura que ningún subárbol tendrá una altura muy superior al otro, manteniendo un rendimiento eficiente incluso en los peores casos.

Conceptos fundamentales

El factor de balance se define como la altura del subárbol izquierdo menos la altura del subárbol derecho:

$$FB(v) = h(hijo\ izquierdo) - h(hijo\ derecho)$$

Donde:

- Si $|FB(v)| \leq 1 \rightarrow$ el nodo está balanceado
- Si $|FB(v)| \geq 2 \rightarrow$ el nodo está desbalanceado y requiere una rotación

Este concepto es la base del funcionamiento de los árboles AVL. Cuando este valor excede el rango permitido, se ejecutan operaciones de rotación que redistribuyen los nodos para restaurar el equilibrio del árbol.

Fórmulas Esenciales del Árbol AVL

Número mínimo de nodos para una altura h

Se define:

- $n(1) = 1$
- $n(2) = 2$

Y para $h \geq 3h$:

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

Esta ecuación es similar a la sucesión de Fibonacci. Con ella, se demuestra que:

$$n(h) > 2^{\frac{h}{2} - 1}$$

Aplicando logaritmos:

$$log\, n(h) > \frac{h}{2} - 1$$

Entonces:

$$h < 2\, log(n) + 2$$

Rotaciones en árboles AVL

Las rotaciones son mecanismos que permiten corregir desbalances estructurales generados por inserciones o eliminaciones.

1. Rotación simple a la derecha (RR): Se utiliza cuando un nodo presenta un desbalance hacia la izquierda debido a inserciones en el subárbol izquierdo-izquierdo.

$$FB(z) = 2, FB(y) = 1$$

2. Rotación simple a la izquierda (LL): Ocurre cuando la inserción se realiza en el subárbol derecho-derecho.

$$FB(z) = -2, FB(y) = -1$$

3. Rotación doble izquierda-derecha (LR):
   Caso:
   - El nodo se desbalancea hacia la izquierda
   - Pero el hijo izquierdo está inclinado hacia la derecha

El proceso consiste en:

   a. Rotar izquierda sobre el hijo izquierdo
   b. Rotar derecha sobre el nodo desbalanceado

4. Rotación doble derecha-izquierda (RL): Corrige desbalances del subárbol derecho-izquierdo mediante una rotación derecha seguida de una izquierda.
   a. Rotar derecha sobre el hijo derecho
   b. Rotar izquierda sobre el nodo desbalanceado

Estas rotaciones mantienen la propiedad de orden del árbol binario de búsqueda y preservan la eficiencia del árbol tras cada operación.

Aplicaciones de los árboles AVL

- Los árboles AVL se utilizan ampliamente en:
- Sistemas de bases de datos: permiten búsquedas rápidas y consistentes.
- Sistemas embebidos y tiempo real: requieren eficiencia y tiempos predecibles.
- Motores de índices: su estructura balanceada los hace útiles para manejar grandes cantidades de datos con tiempos de acceso mínimos.
- Implementación de estructuras multiconjunto y diccionarios.

Conclusiones

Los árboles AVL se destacan por su capacidad de garantizar tiempos óptimos y consistentes en operaciones de búsqueda, inserción y eliminación. Aunque su implementación es más compleja que un árbol binario de búsqueda tradicional, su eficiencia en escenarios con acceso intensivo a datos los convierte en una estructura valiosa. La literatura visitada para este caso de estudio demuestra que esta estructura continúa siendo relevante y competitiva gracias a su balance estricto y rendimiento búsquedas.

Bibliografía

Brown, R. A. (2025). Comparative performance of the AVL tree and three variants of the red-black tree. En *arXiv [cs.DS]*. http://arxiv.org/abs/2406.05162

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). *Data Structures and Algorithms in Java*. https://dl.ebooksworld.ir/motoman/Wiley.Data.Structures.and.Algorithms.in.Java.6th.Edition.www.EBooksWorld.ir.pdf

Mehta, D. P., & Sahni, S. (Eds.). (2018). *Handbook of Data Structures and Applications*. https://api.pageplace.de/preview/DT0400.9781498701884_A34045701/preview-9781498701884_A34045701.pdf

Self-balancing binary search trees. (2018, junio 6). GeeksforGeeks. https://www.geeksforgeeks.org/dsa/self-balancing-binary-search-trees/

Yijiang, L. (2024, junio 25). Liu, Y. (2024). *Statistical Machine Learning Lecture Notes*. Nagoya University. https://www.math.nagoya-u.ac.jp/~richard/teaching/s2024/SML_Liu_2.pdf