Segundo Estudio De Caso: Árboles AVL.

Estudiante:

Minor Andrés Mora Vargas

Escuela de ingeniería del Software, U CENFOTEC

Estructuras de Datos, SOFT-10

Prof. Romario Salas Cerdas

San José, 30 de noviembre de 2025.

Tabla de Contenido

Introducción

El árbol AVL es una de las estructuras de datos más relevantes dentro de la programación moderna, porque combina orden, eficiencia y estabilidad en un solo mecanismo. A diferencia de un árbol binario tradicional, cuyos niveles pueden crecer de manera desequilibrada y volver lentas sus operaciones, el árbol AVL incorpora un sistema de auto balanceo que mantiene su forma siempre controlada. Es decir, cada vez que un dato entra o sale, el árbol revisa su propio estado y decide si debe reorganizarse para seguir siendo eficiente. Esta capacidad de ajustarse constantemente convierte a los AVL en herramientas ideales cuando se trabaja con grandes cantidades de información y se requiere velocidad en todo momento.

La clave de su funcionamiento está en el cálculo del factor de balance y en la aplicación de rotaciones específicas que corrigen cualquier desajuste. Cada inserción o eliminación no solo afecta a un nodo, sino que puede alterar la estructura completa; por eso el AVL actúa de inmediato para mantener el equilibrio entre sus subárboles. Entender cómo ocurre todo este proceso es fundamental, pues revela cómo una estructura aparentemente simple puede sostener el rendimiento de operaciones complejas como búsquedas, inserciones y eliminaciones, manteniéndolas siempre en tiempo logarítmico.

En este trabajo se desarrolla la lógica interna detrás de estos árboles auto balanceados, analizando paso a paso cómo se detectan los desbalances, cómo se decide qué tipo de rotación aplicar y cómo cada ajuste contribuye a conservar la armonía del árbol. Más que una descripción técnica, se busca comprender la idea que sostiene a los árboles AVL: la importancia del equilibrio como principio fundamental para asegurar que el flujo de datos sea rápido, ordenado y confiable. Al final, se evidencia cómo este mecanismo, basado en pequeñas correcciones locales, es capaz de mantener bajo control una estructura completa y garantizar que cada operación retorne exactamente el resultado esperado.

Marco Teórico

Los árboles AVL son una variante de los árboles binarios de búsqueda que incorporan un mecanismo automático de balanceo para mantener su eficiencia. Su propósito principal es asegurar que ninguna de sus ramas crezca significativamente más que otra, ya que esto afectaría directamente el rendimiento de operaciones como buscar, insertar o eliminar. Para evitar ese desbalance, los AVL utilizan el concepto de factor de balance, el cual se obtiene restando la altura del subárbol derecho a la altura del subárbol izquierdo. Mientras el valor se mantenga entre -1 y 1, el árbol está equilibrado; si llega a -2 o +2, se debe corregir la estructura.

El equilibrio del árbol depende de la actualización de las alturas de cada nodo después de cualquier modificación. Una vez que se inserta o elimina un elemento, se recalculan las alturas desde el nodo afectado hasta la raíz. Si en este proceso se detecta un desbalance, se aplican algoritmos de rotación, que permiten reorganizar los nodos para recuperar la forma adecuada sin perder la propiedad del árbol binario de búsqueda.

Las rotaciones que puede realizar un árbol AVL son cuatro: rotación simple a la izquierda, rotación simple a la derecha, rotación doble izquierda y derecha y rotación doble derecha e izquierda. Cada una responde a un tipo específico de desbalance y se ejecuta únicamente cuando el factor de balance del nodo y su hijo indican el patrón de desajuste. Estas rotaciones son movimientos pequeños pero muy eficientes que ajustan la estructura sin alterar el orden de los elementos.

En cuanto a sus operaciones, la búsqueda funciona igual que en cualquier árbol binario de búsqueda, mientras que la inserción y la eliminación deben completar pasos adicionales para verificar y restaurar el balance del árbol. Esto garantiza que, sin importar cuántas modificaciones se realicen, la altura del AVL se mantenga controlada y sus operaciones sigan ejecutándose en tiempo.

Gracias a su estabilidad y eficiencia, los árboles AVL son ampliamente utilizados en sistemas donde se necesita acceso rápido y ordenado a la

información, como bases de datos, índices de búsqueda y estructuras de almacenamiento internas de distintos lenguajes de programación.

Rotaciones en Árboles AVL:

Las rotaciones son los mecanismos principales que permiten corregir los desbalances que aparecen en un árbol AVL después de insertar o eliminar un nodo. Cada vez que se modifica la estructura, los factores de balance de los nodos pueden alterarse; si alguno alcanza un valor de +2 o -2, el árbol identifica que existe una desviación significativa y debe reacomodarse. Las rotaciones no modifican el orden de los elementos, sino únicamente la forma del árbol, garantizando que continúe siendo un árbol binario de búsqueda.

En total, un árbol AVL utiliza cuatro tipos de rotaciones. Cada una corresponde a un caso distinto de desbalance dependiendo de cómo estén distribuidos los nodos en los subárboles izquierdo y derecho. Las rotaciones simples se aplican cuando el desbalance ocurre de manera directa hacia un lado, mientras que las rotaciones dobles se utilizan cuando el desequilibrio combina dos direcciones diferentes dentro del mismo subárbol.

1. Rotación Simple a la Derecha (Caso LL)

La rotación simple a la derecha se utiliza cuando el desbalance ocurre hacia el lado izquierdo del árbol. Específicamente, sucede cuando un nodo tiene un factor de balance de +2 y su hijo izquierdo presenta un factor de balance igual o mayor que 0. Este caso indica que el subárbol izquierdo creció más de lo permitido y debe reorganizarse para reducir su altura. La rotación reacomoda los nodos de manera que el hijo izquierdo pasa a ser la nueva raíz del subárbol, mientras que el nodo desbalanceado se convierte en su hijo derecho.

2. Rotación Simple a la Izquierda (Caso RR)

Este caso es el opuesto al anterior. La rotación simple a la izquierda se utiliza cuando el desbalance aparece hacia el lado derecho. Ocurre cuando un nodo presenta un factor de balance de -2 y el hijo derecho tiene un factor de balance igual o menor que 0. En esta situación, el subárbol derecho ha crecido demasiado y se debe girar hacia la izquierda. Tras aplicar la rotación, el hijo

derecho se convierte en la nueva raíz del subárbol y el nodo desbalanceado pasa a ser su hijo izquierdo.

3. Rotación Doble Izquierda–Derecha (Caso LR)

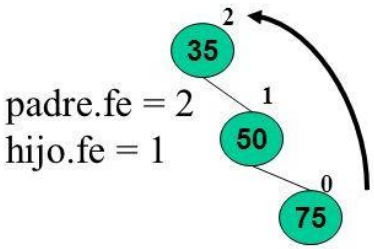
La rotación doble izquierda–derecha se aplica cuando el desbalance parece estar hacia la izquierda, pero el subárbol izquierdo muestra una tendencia hacia la derecha. Esto sucede cuando un nodo tiene un factor de balance de +2, pero el hijo izquierdo muestra un factor de balance de -1. En este caso, no basta una rotación simple porque el subárbol interno también está inclinado. La corrección se realiza en dos pasos: primero se aplica una rotación simple a la izquierda sobre el hijo izquierdo, y luego una rotación simple a la derecha sobre el nodo principal. Este ajuste doble permite alinear correctamente la estructura para recuperar el equilibrio.

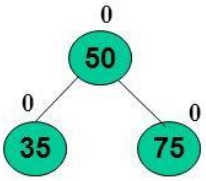
4. Rotación Doble Derecha–Izquierda (Caso RL)

Por último, la rotación doble derecha–izquierda se utiliza cuando el desbalance ocurre hacia la derecha y el hijo derecho muestra inclinación hacia la izquierda. Esto sucede cuando el nodo desbalanceado tiene un factor de balance de -2 y su hijo derecho presenta un factor de +1. Como en el caso anterior, tampoco es suficiente una rotación simple. Primero se ejecuta una rotación simple a la derecha sobre el hijo derecho, y luego se aplica una rotación simple a la izquierda sobre el nodo desbalanceado. El resultado final es un subárbol equilibrado donde la nueva raíz queda correctamente posicionada entre sus valores menores y mayores.

Ejemplos gráficos de estas las operaciones AVL:

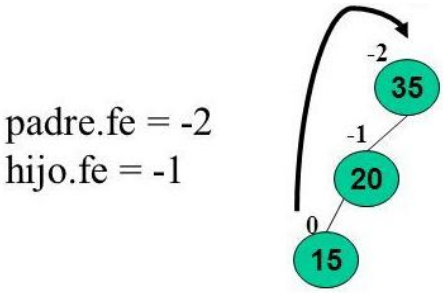
Rotación Simple a la Derecha:

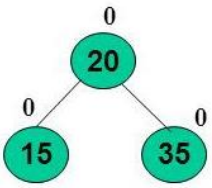




padre.der = hijo.izq
hijo.izq = padre
padre = hijo

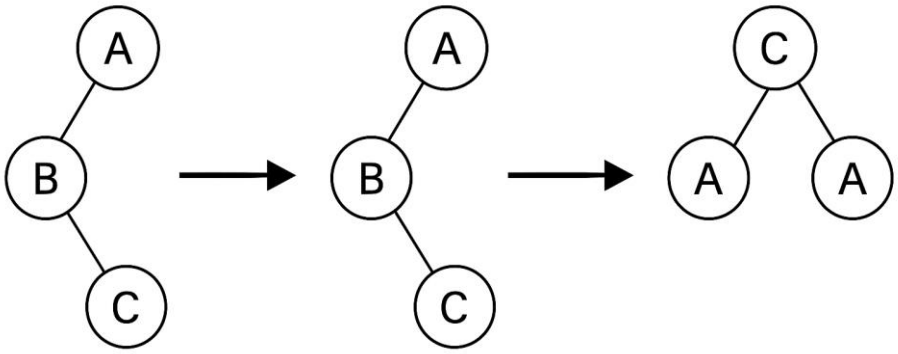
Rotación Simple a la Izquierda:



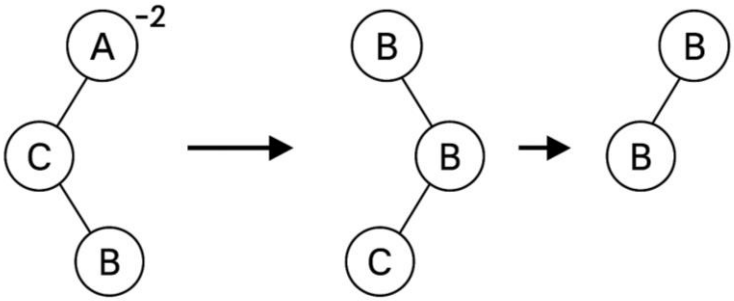


padre.izq = hijo.der
hijo.der = padre
padre = hijo

Rotación Doble Izquierda–Derecha:



Rotación Doble Derecha–Izquierda:



Aplicaciones de los Árboles AVL

Los árboles AVL tienen un papel fundamental en muchos sistemas reales donde es necesario acceder a la información de forma rápida, ordenada y constante. Su capacidad para mantenerse balanceados sin importar la cantidad de inserciones o eliminaciones los convierte en una estructura ideal para escenarios donde el rendimiento debe ser estable. En diferentes herramientas tecnológicas, como bases de datos, motores de búsqueda internos, sistemas de archivos, compiladores, enrutamiento de redes y aplicaciones que requieren índices ordenados, los AVL permiten que cada operación se mantenga en tiempo logarítmico incluso en los peores casos. Esto significa que la estructura no se degrada con el uso continuo, algo vital en aplicaciones que procesan miles o millones de operaciones por segundo. Su mecanismo de rotaciones y control de alturas garantiza que el sistema pueda insertar, eliminar o buscar datos sin que la eficiencia se vea afectada, lo que los convierte en una opción segura, robusta y confiable para cualquier sistema que dependa de un acceso rápido y predecible a la información.

Aplicación en la vida real	Descripción y uso	¿Por qué los árboles AVL son adecuados?
Motores de búsqueda internos.	Se utilizan para mantener colecciones de datos ordenadas, como conjuntos y mapas.	Mantienen siempre los elementos ordenados y permiten búsquedas y actualizaciones rápidas en tiempo logarítmico.
Bases de datos y sistemas de indexación.	Los índices de tablas deben permitir búsquedas rápidas por llaves.	Evitan que las búsquedas se vuelvan lentas, incluso con millones de registros. El auto balanceo mantiene la eficiencia.
Sistemas operativos.	Se utilizan para organizar rutas, directorios e	El equilibrio garantiza acceso constante a rutas,

	inodos, donde se requiere encontrar archivos rápidamente.	sin importar cuántos archivos o carpetas existan.
Redes y telecomunicaciones.	Las rutas deben insertarse, eliminarse y consultarse constantemente.	AVL mantiene las operaciones consistentes y evita latencias en la toma de decisiones sobre rutas.
Compiladores.	Se almacenan identificadores, variables y funciones con acceso inmediato durante la compilación.	Aseguran que las búsquedas sean rápidas incluso con programas grandes con miles de símbolos.
Aplicaciones financieras.	Se almacenan eventos ordenados por tiempo, precio o prioridad.	Previenen que los grandes volúmenes de datos ralenticen cálculos, búsquedas o reportes.
Herramientas de análisis de datos.	Se utilizan para mantener datos ordenados antes de ser procesados.	Garantizan acceso eficiente cuando se requiere filtrar, ordenar o buscar repetidamente.
Sistemas de videojuegos.	Se gestionan objetos ordenados por prioridades o distancias.	Proveen acceso rápido y estable en sistemas que deben responder en tiempo real.

Conclusión

Los árboles AVL representan una solución sólida y eficiente para el manejo de información ordenada dentro de la programación. Su capacidad de auto balanceo garantiza que las operaciones esenciales buscar, insertar y eliminar se mantengan en tiempo logarítmico incluso cuando la cantidad de datos crece considerablemente. A lo largo del trabajo se demostró cómo el cálculo del factor de balance, la actualización de alturas y la aplicación de rotaciones permiten que el árbol se reorganice de manera precisa y controlada, corrigiendo cualquier desbalance sin comprometer la estructura original del árbol binario de búsqueda. Este proceso, aunque sencillo en apariencia, es la clave que sostiene la eficiencia y estabilidad del AVL frente a escenarios cambiantes y dinámicos.

Además, se analizaron diversas aplicaciones reales en las que los árboles AVL resultan especialmente útiles, como bases de datos, motores de búsqueda, sistemas de archivos, estructuras internas de lenguajes de programación y plataformas de almacenamiento digital. En estos contextos, la rapidez de acceso y la consistencia al manipular grandes volúmenes de datos son fundamentales, y los AVL cumplen perfectamente con dichas exigencias. Su diseño garantiza que los tiempos de respuesta se mantengan óptimos incluso ante operaciones frecuentes y cambios continuos en la estructura.

En conjunto, estudiar los árboles AVL no solo permite comprender una de las estructuras de datos más elegantes y robustas, sino también apreciar la importancia del equilibrio dentro del diseño de algoritmos. La lógica detrás de los AVL muestra cómo pequeñas correcciones locales pueden sostener el funcionamiento eficiente de toda una estructura global, convirtiéndolos en herramientas esenciales para cualquier programador que busque soluciones estables, escalables y confiables.

Referencias

DataCamp. (2024, July 29). AVL Tree: Complete Guide With Python Implementation. https://www.datacamp.com/tutorial/avl-tree

GeeksforGeeks. (2023). AVL Tree (Self-Balancing Binary Search Tree). https://www.geeksforgeeks.org/avl-tree-set-1-insertion/

Programiz. (2023). AVL Tree – Insertion, Deletion, and Rotations Explained. https://www.programiz.com/dsa/avl-tree

TutorialsPoint. (2023). Data Structures – AVL Trees. https://www.tutorialspoint.com/data_structures_algorithms/avl_tree_algorithm.htm