Resumen: Árboles AVL y su Implementación

1. Introducción a los Árboles AVL

Los árboles AVL son una estructura de datos jerárquica que pertenece al grupo de árboles binarios de búsqueda balanceados. Fueron propuestos por los matemáticos Adelson-Velsky y Landis, de ahí el acrónimo AVL. Estos árboles buscan corregir un problema común en los árboles binarios de búsqueda tradicionales: el desbalance que puede presentarse al insertar o eliminar elementos, lo cual degrada el rendimiento de las operaciones de búsqueda, inserción y eliminación.

El desbalance en un árbol se manifiesta cuando las alturas de los subárboles izquierdo y derecho de cualquier nodo difieren significativamente, afectando la eficiencia general del árbol. El árbol AVL garantiza que esta diferencia no supere la unidad, es decir, que la **condición de equilibrio** se mantenga con una **diferencia máxima de altura (Factor de Equilibrio, FE) igual a -1, 0 o 1** entre los subárboles.

2. Características y Condición de Equilibrio

Las principales características de un árbol AVL incluyen:

- Es un árbol binario de búsqueda.
- Se mantiene balanceado de forma automática tras operaciones de inserción o eliminación.
- Cada nodo contiene un valor, punteros a sus hijos y un campo adicional con el factor de equilibrio.

El **factor de equilibrio (FE)** se calcula como la diferencia entre la altura del subárbol izquierdo y el derecho. Si esta diferencia es menor que -1 o mayor que 1, el árbol está desbalanceado, y se deben aplicar rotaciones para corregirlo.

3. Tipos de Rotaciones en AVL

Cuando se rompe la condición de equilibrio, el árbol AVL aplica rotaciones para restaurar su balance. Existen cuatro tipos fundamentales de rotaciones:

1. **Rotación Simple a la Derecha (RSR):** Se aplica cuando el desbalance ocurre en el subárbol izquierdo del subárbol izquierdo.
2. **Rotación Simple a la Izquierda (RSI):** Se usa cuando el desbalance se da en el subárbol derecho del subárbol derecho.
3. **Rotación Doble Izquierda-Derecha (RDI):** Ocurre cuando el desbalance está en el subárbol derecho del subárbol izquierdo.
4. **Rotación Doble Derecha-Izquierda (RDD):** Se presenta cuando el desbalance aparece en el subárbol izquierdo del subárbol derecho.

Estas rotaciones reestructuran el árbol mediante movimientos de enlaces entre nodos, asegurando que el árbol mantenga sus propiedades de búsqueda y equilibrio.

4. Operaciones Fundamentales

En un árbol AVL, las operaciones clásicas de búsqueda, inserción y eliminación se adaptan con mecanismos para preservar el equilibrio:

- **Inserción:** Se realiza como en un árbol binario de búsqueda. Luego, se verifica el FE de los nodos ascendentes desde el nodo insertado. Si algún nodo queda fuera del rango permitido (-1, 0, 1), se aplica una rotación.
- **Eliminación:** Similar a la inserción, pero tras eliminar un nodo, el árbol puede desbalancearse en sentido opuesto. También requiere revisar y posiblemente rotar.

Durante estas operaciones, la complejidad de búsqueda y mantenimiento del equilibrio sigue siendo **O(log n)**, lo que garantiza eficiencia incluso en grandes volúmenes de datos.

5. Ejemplos y Ejercicios

Las diapositivas presentan una serie de ejercicios prácticos donde se deben construir árboles AVL a partir de secuencias de números. Estos ejercicios permiten visualizar cómo se aplican las rotaciones para mantener el equilibrio:

- Ejemplos con secuencias como:
  10, 100, 20, 80, 40, 70
  5, 10, 20, 30, 40, 50, 60
  100, 29, 71, 82, 48, 39, 101, 22, 46

En cada caso, se realiza la inserción ordenada de nodos y, en caso de desbalance, se aplican las rotaciones correspondientes. Además, se pide calcular la **altura del árbol** y analizar la **complejidad de búsqueda**, consolidando el aprendizaje práctico del estudiante.

6. Conclusión

Los árboles AVL son una herramienta esencial para mantener la eficiencia en estructuras de datos dinámicas. Su principal ventaja radica en el equilibrio automático que garantiza tiempos de acceso óptimos. Las rotaciones, tanto simples como dobles, son el mecanismo clave que permite conservar este equilibrio sin comprometer la estructura de árbol binario de búsqueda.

Comprender e implementar árboles AVL no solo mejora las habilidades de programación orientada a estructuras de datos, sino que también sienta las bases para el desarrollo de sistemas más complejos y eficientes en el manejo de información.