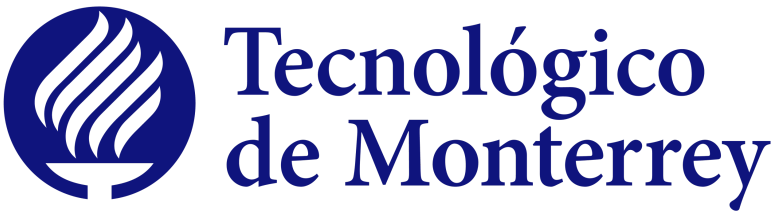
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey



“**Programación de estructuras de datos y algoritmos fundamentales (Gpo 608)**”

David Cantu
22 de Octubre, 2024

“Arboles AVL”

Estudiante:

Oscar Cardenas Valdez A01741965

Árboles Binarios de Búsqueda AVL

Introducción

Un **árbol AVL** es una estructura de datos que pertenece a la categoría de árboles binarios de búsqueda (BST, por sus siglas en inglés) y fue introducida por los matemáticos soviéticos Adelson-Velskii y Landis en 1962. Este tipo de árbol se caracteriza por ser autobalanceado, lo que significa que en todo momento la diferencia de alturas entre los subárboles izquierdo y derecho de cualquier nodo es como máximo 1. Esto asegura que la estructura no se degrade a una lista enlazada, manteniendo así la eficiencia en las operaciones de búsqueda, inserción y eliminación en O(logn)O(logn).

Atributos del ADT

- **Size**: Cantidad de nodos presentes en el árbol.
- **Capacity**: La capacidad máxima de nodos que puede almacenar (no siempre se usa en todas las implementaciones).
- **Data**: El valor almacenado en cada nodo.
- **Height**: Altura del árbol, definida como la longitud del camino más largo desde la raíz hasta una hoja.
- **Root**: El nodo raíz del árbol.
- **Leaves**: Los nodos que no tienen hijos.
- **Depth**: La profundidad de un nodo, es decir, la distancia desde la raíz.
- **Balance Factor**: Factor que indica el balance del nodo, definido como la diferencia entre la altura del subárbol izquierdo y derecho de un nodo.

Métodos Comunes

1. **Insert (Insertar)**: Este método añade un nuevo nodo al árbol AVL. Después de la inserción, el árbol puede desbalancearse, por lo que se ajusta mediante rotaciones.

2. **Remove (Eliminar)**: Elimina un nodo del árbol AVL. Similar a la inserción, puede causar desbalance, el cual se corrige con rotaciones.
3. **Search (Buscar)**: Este método busca un elemento específico dentro del árbol AVL siguiendo la lógica de los árboles binarios de búsqueda, donde los elementos menores están a la izquierda y los mayores a la derecha.
4. **isEmpty (Está Vacío)**: Comprueba si el árbol no contiene ningún nodo.
5. **getSize (Obtener Tamaño)**: Retorna el número de nodos que contiene el árbol.
6. **getRoot (Obtener Raíz)**: Retorna el nodo raíz del árbol.
7. **Rotate (Rotar)**: Realiza las rotaciones necesarias para reequilibrar el árbol cuando está desbalanceado.

Tipos de Rotaciones

- **Rotación simple a la izquierda**: Se utiliza cuando el subárbol derecho es más alto que el subárbol izquierdo.
- **Rotación simple a la derecha**: Se utiliza cuando el subárbol izquierdo es más alto que el subárbol derecho.
- **Rotación doble izquierda-derecha**: Se utiliza cuando el subárbol izquierdo del subárbol derecho es más alto.
- **Rotación doble derecha-izquierda**: Se utiliza cuando el subárbol derecho del subárbol izquierdo es más alto.

Balanceo al Insertar un Nodo

Cuando se inserta un nuevo nodo en un árbol AVL, los pasos son los siguientes:

1. **Inserción**: El nodo se inserta siguiendo las reglas básicas de los árboles binarios de búsqueda.
2. **Actualización de alturas**: Tras la inserción, se actualizan las alturas de los nodos ancestros del nuevo nodo.
3. **Cálculo del factor de balance**: El balance se revisa desde el nodo insertado hasta la raíz, usando la fórmula:

Factor de balance=Altura del subarbol izquierdo−Altura del subarbol derechoFactor de balance=Altura del subarbol izquierdo−Altura del subarbol derecho

4. **Rotación si es necesario**: Si el factor de balance es mayor a 1 o menor a -1, el árbol está desbalanceado y se aplica una rotación para restaurar el equilibrio.

Ejemplo

Supongamos que insertamos el valor 50 en un árbol AVL ya balanceado. Tras la inserción, se actualizan las alturas y si el balance en algún nodo se altera (por ejemplo, un factor de balance de -2 en un nodo), se realiza una rotación para restaurar el balance.

Balanceo al Eliminar un Nodo

Cuando se elimina un nodo, el proceso es el siguiente:

1. **Eliminación**: Se eliminan los nodos utilizando las reglas de un árbol binario de búsqueda.
2. **Actualización de alturas**: Tras la eliminación, se actualizan las alturas de los nodos ancestros del nodo eliminado.
3. **Cálculo del factor de balance**: Al igual que en la inserción, se revisa el factor de balance desde el nodo afectado hasta la raíz.
4. **Rotación si es necesario**: Si en algún nodo el factor de balance no está en el rango [-1, 1], se aplica una rotación.

Ejemplo

Si eliminamos el nodo 70 de un árbol AVL, las alturas de los nodos ancestros de 70 se actualizan, y si el balance de algún nodo se ve alterado (por ejemplo, un factor de balance de +2 en un nodo), se aplica una rotación.

Uso Aplicativo del ADT

Los árboles AVL son especialmente útiles en aplicaciones donde se requiere realizar búsquedas, inserciones y eliminaciones frecuentes con alta eficiencia. Algunos ejemplos incluyen:

- **Sistemas de gestión de bases de datos**: Los árboles AVL son ideales para gestionar índices en bases de datos, asegurando que las operaciones de búsqueda y acceso sean rápidas.
- **Compiladores**: Los compiladores utilizan árboles AVL para gestionar tablas de símbolos, lo que facilita la búsqueda rápida de variables y funciones.
- **Sistemas de archivos**: En algunos sistemas de archivos, los árboles AVL se utilizan para mantener el control de las ubicaciones de los archivos, proporcionando acceso rápido a los mismos.

En conclusión, los árboles AVL son una estructura de datos poderosa que garantizan una complejidad logarítmica en las operaciones más comunes, asegurando que no se degraden a una lista enlazada y manteniendo un balance constante en sus ramas mediante rotaciones.

Referencias

GeeksforGeeks, (2024) AVL Tree Data Structure. Recuperado de: https://www.geeksforgeeks.org/introduction-to-avl-tree/

Pozo, S. (2011). Árboles AVL: Definición. Con Clase. Recuperado de: https://conclase.net/c/edd/cap8