Ejercicio de Laboratorio: Implementación de Árboles AVL

Curso: Estructura de Datos
Estudiantes: Eduardo Zambrano - 2241636 | Joseph Emanuel Sanchez - 2240959

Introducción

En el estudio de las estructuras de datos, los árboles binarios de búsqueda (BST) son fundamentales por su eficiencia en operaciones como búsqueda, inserción y eliminación. Sin embargo, un árbol desbalanceado puede degradar el rendimiento de O(log n) a O(n).

Para superar esta limitación, se implementan los Árboles AVL (Adelson-Velsky y Landis), que son BSTs auto-balanceados que garantizan que la diferencia de altura entre los subárboles izquierdo y derecho nunca sea mayor a 1.

Errores Identificados en el Código Original

Se proporcionó un código inicial para implementar un árbol AVL en Python, el cual presentaba los siguientes errores:

- Las rotaciones (rotate_left, rotate_right) no actualizaban la referencia del nodo padre, por lo que el balanceo no se aplicaba correctamente.
- No se incluía la función de eliminación (delete), requerida para cumplir con el funcionamiento completo de un AVL.
- Faltaba una función de recorrido in-order.
- No había una función para visualizar el árbol y verificar el factor de balance y altura de cada nodo.

Correcciones y Funcionalidades Completadas

Las siguientes correcciones y mejoras fueron aplicadas al código:

1. Se corrigieron las funciones de rotación para que devuelvan el nuevo nodo raíz del subárbol, y se actualizaron sus llamadas.
2. Se implementó delete, que elimina un nodo y reequilibra el árbol.
3. Se añadió inorder_traversal, que devuelve los valores en orden ascendente.
4. Se implementó print_tree, una función opcional que muestra la estructura del árbol y ayuda a visualizar su balance.

Casos de Prueba

Se realizaron las siguientes pruebas para verificar el correcto funcionamiento del árbol AVL:

- **Inserciones**: [10, 20, 30, 40, 50, 25]
  Se esperaban rotaciones para mantener el balance del árbol tras cada inserción.
- **Eliminaciones**: [50, 30]
  Se eliminaron nodos internos y hojas, verificando que el árbol se reequilibra correctamente.

Se utilizó print_tree para visualizar el árbol y inorder_traversal para asegurar el orden correcto de los elementos.

Conclusión

El código final fue corregido con éxito para cumplir con los requisitos pedidos.