Implementación de Árboles Binarios en Python

Alumnos:

Alex Pereyra – lexthus@gmail.com
Lorenzo Ojeda – lorenzomkt.0@gmail.com

Materia: Programación I

Profesor: AUS Bruselario, Sebastián
Tutora: Cimino, Virginia

Fecha de entrega: 09/06/2025

Contenido

1. Introducción

Este trabajo aborda la implementación de árboles binarios en Python utilizando listas como estructura principal. Se eligió este enfoque por su valor didáctico y su aplicabilidad en escenarios donde se requiere una jerarquización eficiente de datos. Los objetivos del trabajo incluyen comprender la estructura de los árboles binarios, realizar distintos tipos de recorrido y representar visualmente la estructura para mejorar su comprensión.

2. Marco Teórico

Un árbol binario es una estructura de datos jerárquica en la que cada nodo tiene como máximo dos hijos: uno a la izquierda y otro a la derecha. Cada nodo contiene un valor y enlaces a sus hijos. Esta estructura se utiliza en múltiples contextos como bases de datos jerárquicas, sistemas de archivos, compiladores y algoritmos de búsqueda binaria.

Según Cormen et al. (2009), los árboles binarios permiten realizar operaciones eficientes como búsqueda, inserción y eliminación. En particular, el árbol binario de búsqueda (BST) mantiene sus elementos ordenados, lo que permite obtener un rendimiento logarítmico promedio.

Los recorridos típicos son:
- Preorden (Raíz - Izquierda - Derecha)
- Inorden (Izquierda - Raíz - Derecha)
- Postorden (Izquierda - Derecha - Raíz)

En este trabajo, la estructura fue representada con listas anidadas en lugar de clases, lo que facilita su aprendizaje sin necesidad de manejar programación orientada a objetos.

3. Caso Práctico

Se construyó un árbol binario con los nodos A, B, C, D, E, F y G. Para representarlo se usó una estructura de listas anidadas, donde cada nodo tiene la forma [valor, hijo_izquierdo, hijo_derecho].

Se implementaron funciones para insertar nodos a la izquierda o derecha respetando la jerarquía. También se desarrollaron funciones recursivas para realizar los recorridos preorden, inorden y postorden.

Además, se incorporó una función `imprimir_arbol()` que rota el árbol 90 grados para mostrarlo en consola de forma jerárquica y visualmente clara.

Este enfoque permitió aplicar los conceptos teóricos de forma concreta, reforzando la comprensión de estructuras recursivas y organización de datos.

4. Metodología Utilizada

El desarrollo se realizó con Python 3.x utilizando listas anidadas para representar los nodos. Se organizaron las tareas en etapas: definición del modelo de nodo, funciones de inserción, funciones de recorrido, y visualización. Las pruebas se realizaron ejecutando recorridos y observando los resultados en consola para verificar su correcto funcionamiento.

5. Resultados Obtenidos

El árbol binario fue construido correctamente, y los recorridos generaron los resultados esperados en consola. La visualización rotada permitió validar visualmente la jerarquía del árbol. Se comprobó que las funciones de inserción y recorrido funcionan correctamente, respetando la lógica de la estructura.

6. Conclusiones

La implementación de árboles binarios mediante listas fue efectiva para fines educativos. Permitió representar la estructura sin necesidad de clases y enfocarse en la lógica de funcionamiento. Se lograron los objetivos propuestos y se reforzaron conocimientos sobre recursividad, jerarquía de datos y organización estructurada.

7. Bibliografía

- Cormen, T. et al. (2009). Introduction to Algorithms.
- Python Software Foundation. https://docs.python.org/3/
- Miller, B. & Ranum, D. Problem Solving with Algorithms and Data Structures using Python.

8. Anexos

- Capturas de pantalla del árbol y los recorridos.
- Link al video: [agregar enlace de YouTube acá]
- Repositorio GitHub: https://github.com/alexmp2602/tp-integrador-arboles-python