Trabajo Práctico Integrador N°2

Programación I

Tema elegido: Estructuras de datos avanzados - Árboles en Python utilizando listas

Alumnos
Alejandro Pedrosa
Luciano de la Rubia

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

2 de junio de 2025

Introducción

Las estructuras de datos permiten organizar y manipular información de manera eficiente dentro de un programa. Entre ellas, los **árboles** son estructuras jerárquicas ampliamente utilizadas en diversas áreas como algoritmos de búsqueda, representación de jerarquías, bases de datos, inteligencia artificial, análisis léxico y más.

En este trabajo integrador, se propone estudiar y aplicar una forma alternativa de implementación de árboles binarios en Python utilizando **listas anidadas**, en lugar de clases u objetos. Este enfoque permite comprender de forma accesible y didáctica los principios fundamentales de esta estructura.

El objetivo principal es **integrar los conceptos teóricos con una implementación práctica en Python**, desarrollando funciones para construir árboles, insertar nodos, recorrerlos e imprimirlos, reflexionando sobre sus ventajas y limitaciones.

Objetivos

Objetivo general:

- Investigar y aplicar el concepto de árboles binarios como estructura de datos avanzada en Python, utilizando listas como representación base.

Objetivos específicos:

- Comprender la estructura y funcionamiento de los árboles binarios.
- Implementar funciones para insertar y recorrer árboles binarios representados mediante listas.
- Analizar las ventajas y desventajas de esta implementación frente a enfoques orientados a objetos.
- Desarrollar un ejemplo práctico funcional y documentado en Python.
- Reflexionar sobre la utilidad didáctica de este enfoque como herramienta para el aprendizaje de estructuras de datos.

Marco teórico

Un **árbol** es una estructura de datos no lineal compuesta por nodos organizados jerárquicamente. El nodo principal se denomina **raíz**, y cada nodo puede tener **cero, uno o más hijos**. Uno de los tipos más comunes de árboles es el **árbol binario**, donde cada nodo tiene como máximo dos hijos: el izquierdo y el derecho.

Características de un árbol binario:

- **Raíz**: es el nodo principal del árbol, desde el cual se derivan todos los demás nodos.
- **Nodos hoja**: aquellos que no tienen hijos.
- **Altura**: cantidad de niveles desde la raíz hasta el nodo más profundo.
- **Subárbol**: cualquier nodo junto con sus descendientes forma un subárbol.

En lenguajes como C++ o Java, es común representar árboles usando estructuras de clases y punteros. Sin embargo, en Python es posible utilizar **listas anidadas**, aprovechando que en este lenguaje las listas pueden contener otras listas (también podríamos usar javascript). Esta técnica representa un nodo como una lista de tres elementos: `[valor, subarbol_izquierdo, subarbol_derecho]`.

Esta forma de representación es particularmente útil cuando queremos explicar el concepto de estructuras de datos como árboles con fines educativos, ya que permite focalizarse en la lógica de la estructura sin la complejidad adicional de la programación orientada a objetos. No obstante, presenta limitaciones cuando se desea escalar o manipular árboles más complejos.

Recorridos en Árboles Binarios

Los árboles binarios pueden ser recorridos de diferentes maneras, cada una con sus propias aplicaciones y características. Los tres métodos principales son:

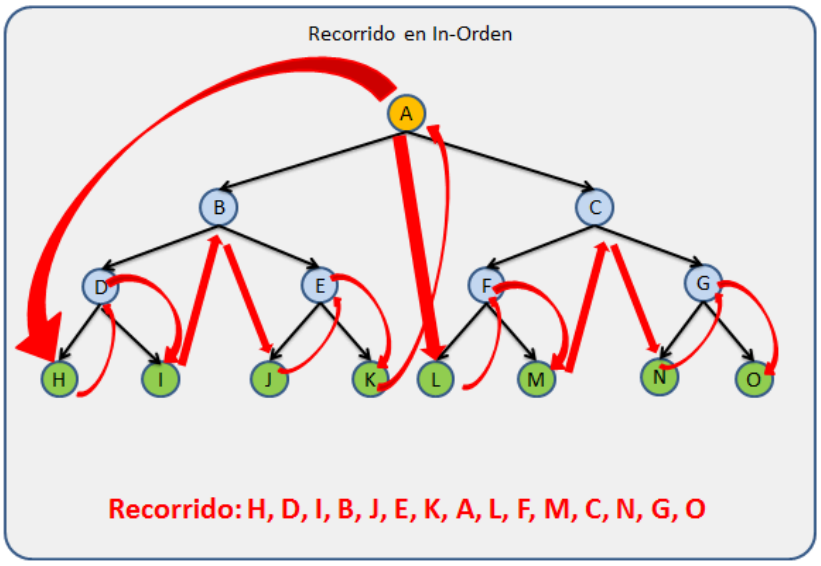
In-Orden (Izquierda - Raíz - Derecha):

- Primero se recorre el subárbol izquierdo

- Luego se visita la raíz

- Finalmente se recorre el subárbol derecho

- Es útil para obtener los valores en orden ascendente en árboles de búsqueda binaria



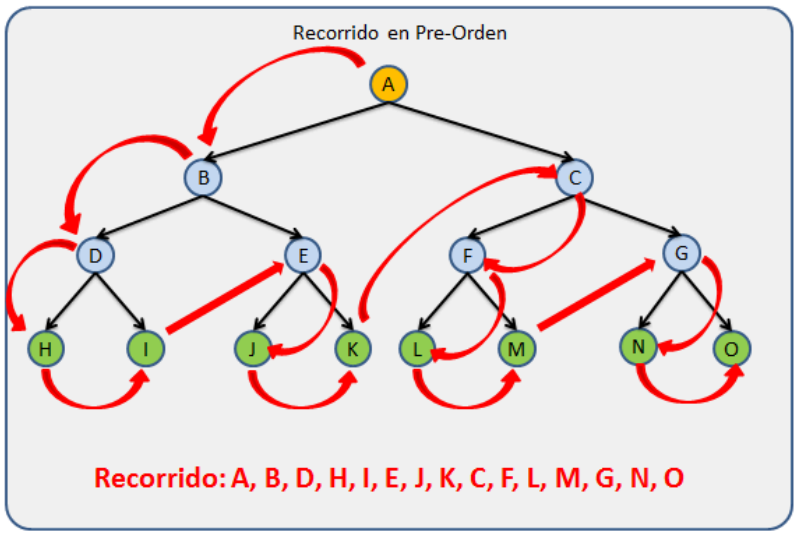
Pre-Orden (Raíz - Izquierda - Derecha):

- Primero se visita la raíz

- Luego se recorre el subárbol izquierdo

- Finalmente se recorre el subárbol derecho

- Útil para crear una copia del árbol o para expresiones prefijas



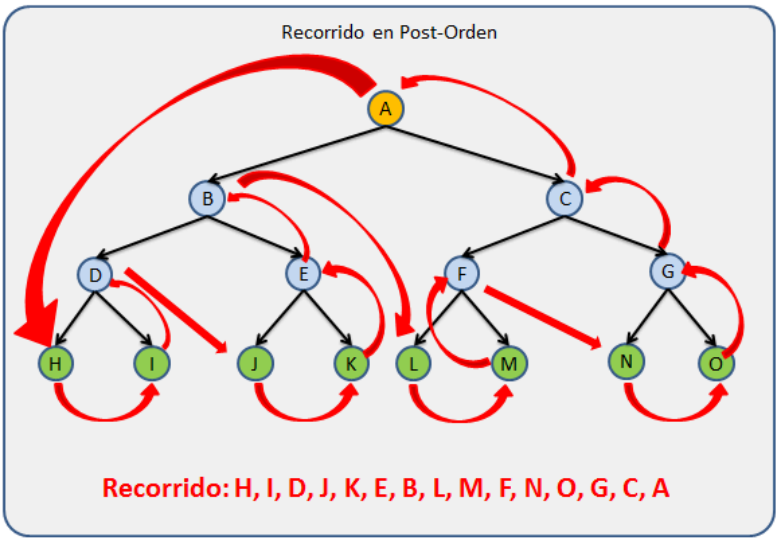
Post-Orden (Izquierda - Derecha - Raíz):

- Primero se recorre el subárbol izquierdo

- Luego se recorre el subárbol derecho

- Finalmente se visita la raíz

- Útil para eliminar el árbol o para expresiones postfijas



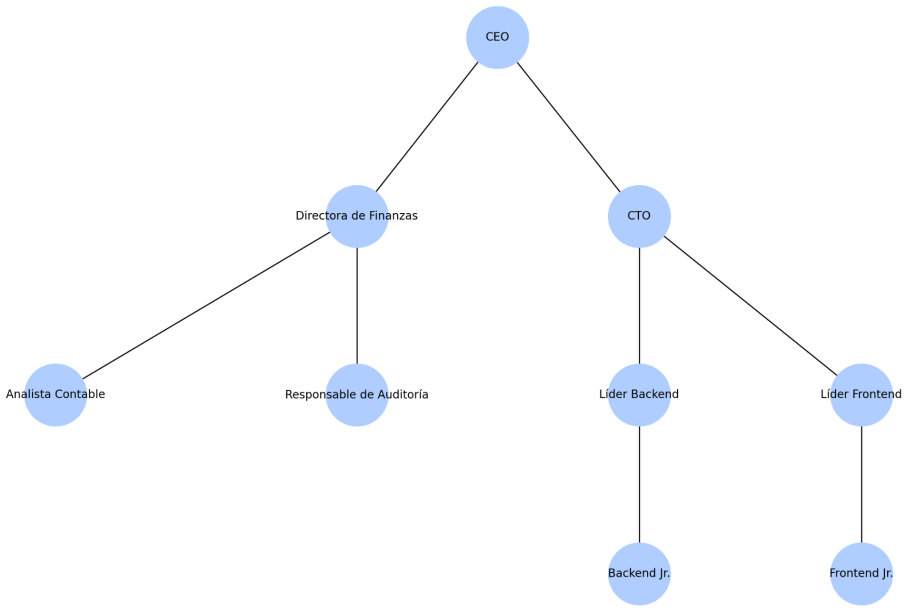
Metodología y caso práctico

El trabajo se desarrolló en torno a la investigación y aplicación práctica de estructuras de datos avanzadas, enfocándonos en los árboles binarios representados mediante listas anidadas en Python. Utilizamos la versión de python 3.

La metodología consistió en los siguientes pasos:

1. Selección del enfoque: se eligió representar árboles sin clases ni objetos, utilizando listas de la forma [valor, subárbol izquierdo, subárbol derecho] por su simplicidad y valor didáctico.
2. Investigación teórica: se consultaron fuentes académicas y documentación oficial de Python para comprender la estructura y operaciones fundamentales de los árboles.
3. Diseño del caso práctico: se propuso como ejemplo una estructura jerárquica empresarial representada mediante un árbol binario, simulando niveles de dirección, áreas y empleados.
4. Desarrollo e implementación: se programaron funciones para crear nodos, insertar hijos, realizar recorridos (preorden, inorden, postorden), e imprimir el árbol con formato visual horizontal y rotado 90°.
5. Visualización y pruebas: se generaron distintas salidas en consola para validar el funcionamiento del árbol, identificando correctamente la raíz, los subárboles y los nodos hoja.
6. Documentación y reflexión: se organizaron los resultados, observaciones y aprendizajes en este documento y se preparó un archivo README junto al código fuente documentado.

El caso práctico elegido representa la estructura jerárquica de una empresa, comenzando por el CEO, pasando por los niveles de Dirección de Finanzas y Tecnología (CTO), y llegando a cargos técnicos específicos. Esta simulación demuestra cómo se puede representar una jerarquía real con árboles simples, facilitando su análisis y navegación.



Análisis de Resultados

La implementación resultó exitosa al demostrar la funcionalidad de los árboles binarios representados con listas. Se logró:

- Crear estructuras jerárquicas claras, incluyendo distintos niveles de profundidad.
- Identificar fácilmente la raíz, los subárboles y los nodos hoja mediante funciones recursivas.
- Visualizar el árbol tanto horizontalmente como rotado 90°, lo que facilita la comprensión de las relaciones jerárquicas.
- Mantener bajo nivel de complejidad sintáctica, favoreciendo la claridad del código para estudiantes que se inician en estructuras de datos.

Se comprobó que, aunque esta implementación es limitada en cuanto a escalabilidad y flexibilidad (por ejemplo, no permite más de dos hijos por nodo sin adaptar la estructura), resulta sumamente útil para ilustrar conceptos básicos de árboles binarios en un entorno educativo.

Se implementaron los tres tipos principales de recorridos de árboles binarios (in-orden, pre-orden y post-orden), demostrando cómo cada uno produce un orden diferente al visitar los nodos. Esto resulta particularmente útil para diferentes aplicaciones: el recorrido in-orden es ideal para obtener una secuencia ordenada, el pre-orden para copiar la estructura del árbol, y el post-orden para operaciones que requieren procesar los hijos antes que el nodo padre, como en el cálculo de expresiones matemáticas.

Conclusión

El trabajo permitió comprender en profundidad el funcionamiento y utilidad de los árboles binarios como estructura de datos avanzada. A través del uso de listas en Python, se logró una implementación clara, funcional y accesible que sirvió para modelar jerarquías reales como la de una organización empresarial.

La representación con listas, aunque limitada para escenarios complejos, ofreció una herramienta pedagógica poderosa al permitir enfocarse en la estructura lógica del árbol sin las abstracciones propias de la programación orientada a objetos.

La implementación de los diferentes tipos de recorrido demostró la versatilidad de los árboles binarios para procesar información en distintos órdenes según las necesidades del problema, reforzando su utilidad en diversos contextos algorítmicos.

Se concluye que este enfoque es ideal para introducir el tema de árboles binarios en cursos iniciales de programación, y sienta las bases para futuras exploraciones con árboles balanceados, genéricos o representaciones orientadas a objetos.

Bibliografía

- Arboles
- Implementación de arboles binarios en python como listas anidadas
- Python Software Foundation. https://docs.python.org
- Video de Charly Cimino