TRABAJO PRACTICO INTEGRADOR DE PROGRAMACIÓN

Estructuras de datos avanzadas: Árboles.

ALUMNOS: Alan Jofre y Jazmin Herrera

PROFESOR: Martín Aristiaran

FECHA DE ENTREGA: 09/06/2025

Índice

1 - INTRODUCCIÓN

¿Por qué se eligió este tema?

Se eligió el tema de árboles en Python debido a su relevancia fundamental dentro de las estructuras de datos en programación. Los árboles son utilizados en múltiples áreas del desarrollo de software, incluyendo algoritmos de búsqueda, sistemas de archivos, inteligencia artificial, bases de datos, compiladores, y más. Estudiar árboles permite comprender cómo organizar y gestionar información de manera jerárquica y eficiente. Además, Python, al ser un lenguaje de alto nivel y de fácil sintaxis, resulta ideal para representar y manipular estas estructuras, lo cual facilita el aprendizaje de conceptos avanzados sin distraerse con complejidades del lenguaje.

¿Qué importancia tiene en la programación?

Los árboles tienen una gran importancia en la programación porque permiten representar relaciones jerárquicas entre datos, algo que no es posible hacer de forma eficiente con estructuras lineales como listas o arreglos. Gracias a su estructura, los árboles permiten realizar búsquedas, inserciones y eliminaciones en tiempos reducidos, especialmente en árboles balanceados. Son la base de estructuras como árboles binarios de búsqueda (BST), heaps, árboles AVL, B-trees, y se utilizan en algoritmos de toma de decisiones, en motores de juegos, en parsers de lenguajes y en almacenamiento de datos jerárquicos (como XML o JSON). El manejo adecuado de árboles mejora significativamente la eficiencia, escalabilidad y organización de los programas.

¿Qué objetivos se propone alcanzar con el desarrollo del trabajo?

Con el desarrollo de este trabajo práctico integrador, se propone alcanzar los siguientes objetivos:

- **Comprender en profundidad la estructura de datos tipo árbol**, sus componentes (nodos, raíces, ramas, hojas), y su funcionamiento general.

- **Aplicar los conocimientos adquiridos en Python**, implementando árboles con listas anidadas o clases, desarrollando funciones para insertar, recorrer y visualizar estructuras jerárquicas.

- **Fomentar el pensamiento lógico y recursivo**, a través de la implementación de recorridos (preorden, inorden, postorden) y la resolución de problemas aplicados.

- **Desarrollar habilidades prácticas y analíticas**, que permitan al estudiante modelar soluciones reales mediante árboles, promoviendo una visión integral de la programación estructurada y orientada a objetos.

- **Integrar teoría y práctica** a través de la construcción de un caso funcional, el desarrollo del código, su documentación y la exposición de los resultados obtenidos.

2 - MARCO TEÓRICO

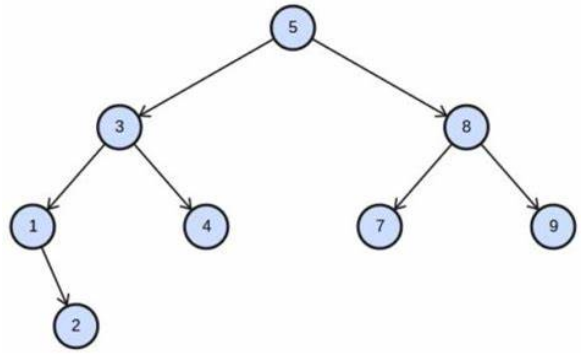
¿Qué son los árboles?

Los árboles son estructuras de datos jerárquicas que se utilizan para organizar y almacenar información de manera eficiente. Son representados, en general, a través de grafos que tienen nodos conectados entre sí por arcos o ramas. Un nodo es una unidad básica de un árbol. Cada nodo contiene dos componentes principales: un valor y un puntero a sus nodos hijos, si es que los tiene. Los nodos están conectados a través de ramas.

¿Qué son los árboles binarios?

Un árbol binario es una estructura en la que cada nodo puede tener hasta dos descendientes: un hijo a la izquierda y otro a la derecha. En un árbol binario, cada nodo se ramifica en, como máximo, dos direcciones. El hijo izquierdo y el hijo

derecho representan subárboles independientes que pueden a su vez tener sus propios hijos.



Para entender mejor, debemos saber los siguientes conceptos:

NODOS: Los Nodos son centros de datos, los cuales contienen la información que deseamos en su interior. Están representados en formas de bolitas.

RAMAS: Las ramas son las que conectan los nodos.

Clasificación de nodos según su relación con otros nodos

Nodo Padre: Se utiliza este término para llamar a todos aquellos nodos que tiene al menos un hijo.

Nodo Hijo: Los hijos son todos aquellos nodos que tiene un padre.

Nodo Hermano: Los nodos hermanos son aquellos nodos que comparten a un mismo padre en común dentro de la estructura.

Clasificación de nodos según su ubicación en el árbol

Nodo raíz: es el punto de entrada a la estructura, generalmente ubicado en la parte más superior del árbol. No tiene un nodo padre y es único, es decir, cada árbol tiene un solo nodo raíz.

Nodo hoja: es cualquier nodo que no tenga hijos, es decir, un nodo terminal del árbol. Los nodos hoja están en los extremos de la estructura de un árbol y no tienen más ramificaciones debajo de ellos.

Nodo rama o interno: es cualquier nodo que tiene un padre y, al menos, un hijo. A diferencia de un nodo hoja, un nodo rama no es terminal y sigue propagando la jerarquía del árbol hacia abajo.

Propiedades de los árboles

Grado y orden

- **Grado**: El grado de un árbol indica la cantidad de hijos que posee un nodo.
- **Orden**: El orden de un árbol es el número máximo de hijos que puede tener un nodo en ese árbol.

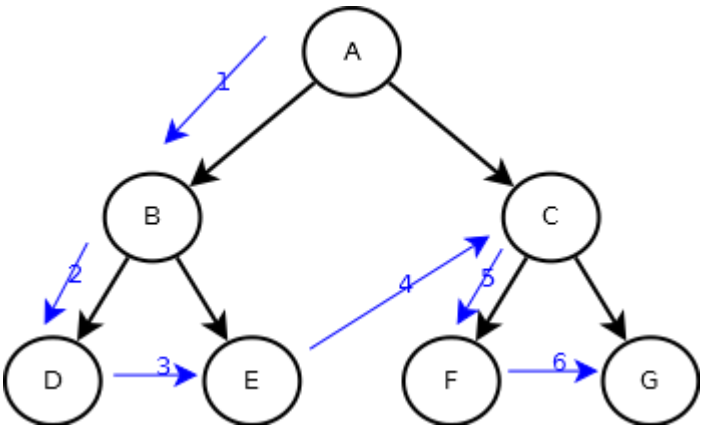
Nivel y altura

- **Nivel**: El nivel de un nodo es la longitud del camino que lo conecta al nodo raíz más uno. Un nivel puede contener uno o más nodos.
- **Altura**: La altura de un árbol es la longitud del camino más largo desde la raíz hasta una hoja (nodo sin hijos).

Formas de recorrer árboles binarios

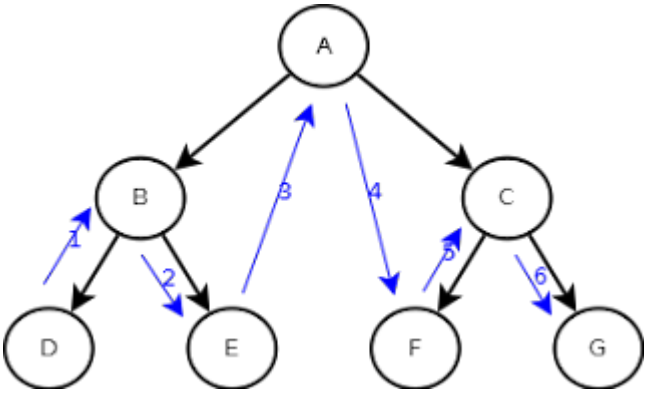
Preorden

1. Se comienza por la raíz bajando hacia el hijo izquierdo de la raíz.
2. Se recorre recursivamente el subárbol izquierdo.
3. Se sube hasta el hijo derecho de la raíz.
4. Se recorre recursivamente el subárbol derecho.



Inorden

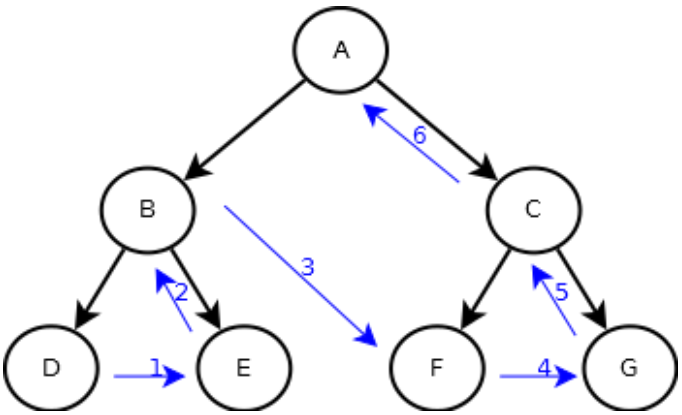
1. Se comienza por el nodo hoja que se encuentre más a la izquierda de todos.
2. Se sube hacia su nodo padre.
3. Se baja hacia el hijo derecho del nodo recorrido en el paso 2.
4. Se repiten los pasos 2 y 3 hasta terminar de recorrer el subárbol izquierdo.
5. Se visita el nodo raíz.
6. Se recorre el subárbol derecho de la misma manera que se recorrió el subárbol izquierdo.



Postorden

1. Se comienza por el nodo hoja que se encuentre más a la izquierda de todos.
2. Se visita su nodo hermano.
3. Se sube hacia el padre de ambos.
4. Si tuviera hermanos, se visita su nodo hermano.
5. Se repiten los pasos 3 y 4 hasta terminar de recorrer el subárbol izquierdo.

6. Se recorre el subárbol derecho, comenzando por el nodo hoja que se encuentre más a la izquierda y siguiendo el mismo procedimiento que con el subárbol izquierdo
7. Se visita el nodo raíz.



3 - CASO PRÁCTICO

Descripción del problema: en este caso se realizará un código en python donde el usuario pueda ingresar datos de cualquier tipo para poder desarrollar un árbol en el que pueda:

1. Insertar un nodo a la izquierda o a la derecha
2. Visualizar árbol
3. Recorrer el árbol en preorden, inorden, postorden
4. Eliminar nodo (hoja o con un solo nodo hijo)
5. Contar los nodos hoja.
6. Calcular la altura del árbol creado.

4 - METODOLOGÍA UTILIZADA

1. Elegimos el tema a desarrollar.
2. Buscamos información útil para la estructura del trabajo.
3. Redactamos el material teórico (Herramienta utilizada: **Google docs**)
4. Elegimos un caso práctico que abarque lo visto anteriormente.
5. Redactamos códigos. (Herramienta utilizada: **VCS con Python 3.11**)

6. Evaluamos que los códigos funcionen.
7. Controlamos los resultados obtenidos.
8. Nos dividimos el material de forma equitativa para poder desarrollar el video.
9. Diseñamos la presentación. (Herramienta utilizada: **Canva**)
10. Grabamos el video. (Herramienta utilizada:**Windows Game Bar**)

5 - RESULTADOS OBTENIDOS

El caso práctico permitió implementar y comprender en profundidad el funcionamiento de estructuras de datos tipo árbol, particularmente los árboles binarios. Se desarrollaron funciones para crear árboles, insertar nodos a izquierda y derecha, y realizar distintos tipos de recorridos (preorden, inorden y postorden). Además, se logró representar relaciones jerárquicas y organizar datos de manera estructurada y eficiente.

Logramos que funcionen correctamente, implementando la estructura de árbol utilizando listas anidadas. Las funciones de inserción y visualización del árbol funcionaron correctamente con distintos conjuntos de datos de prueba. Y los algoritmos de recorrido (inorden, preorden y postorden) devolvieron los resultados esperados, validando la lógica implementada.

Se generaron una serie de dificultades a la hora de probar el código ya que la visualización del árbol no se veía como deseábamos. También nos costó colocar el código para que nos devuelva lo que pedíamos.

6 - CONCLUSIÓN

El desarrollo de este trabajo integrador permitió comprender en profundidad el funcionamiento y la importancia de las estructuras de datos tipo árbol en el ámbito de la programación. A través del estudio teórico y la implementación práctica en Python, se logró representar relaciones jerárquicas entre datos de forma clara, eficiente y escalable. Se exploraron conceptos clave como nodos, raíces, hojas y recorridos (preorden, inorden y postorden), así como técnicas de inserción y visualización que fortalecen el pensamiento lógico y el diseño algorítmico.

Además, este trabajo contribuyó a consolidar habilidades fundamentales en programación, como el uso de funciones, estructuras recursivas y la organización modular del código. El análisis y la construcción de árboles también permitieron reflexionar sobre su aplicación en contextos reales como bases de datos, inteligencia artificial y estructuras de archivos.

En definitiva, el estudio de árboles en Python no solo amplió el conocimiento técnico, sino que también potenció la capacidad de resolución de problemas complejos mediante herramientas eficientes y reutilizables, representando un avance significativo en la formación como programador.

7- BIBLIOGRAFÍA

¿Qué son los árboles binarios?:

https://medium.com/@diego.coder/grafos-y-%C3%A1rboles-en-python-2d165dfc8bbd

Imágenes:
https://www.juliocesar.cloud/2013/09/arbol-binario-y-recorridos-preorden.html

Clasificación de nodos según su relación con otros nodos:

https://www.oscarblancarteblog.com/2014/08/22/estructura-de-datos-arboles/

Resto del material teórico:

https://docs.google.com/document/d/10k16oL15EeyOaq92aoi4qwK3t_22X29-FSV2iV-8N1U/edit?tab=t.0

8- Anexos

Link de YouTube:

https://youtu.be/MR5IR9nIr8o?feature=shared

Link de diapositiva:

https://www.canva.com/design/DAGp3kt2tGk/9F0B7DQGRfB8zk3b8ZfwYg/edit?utm_content=DAGp3kt2tGk&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton