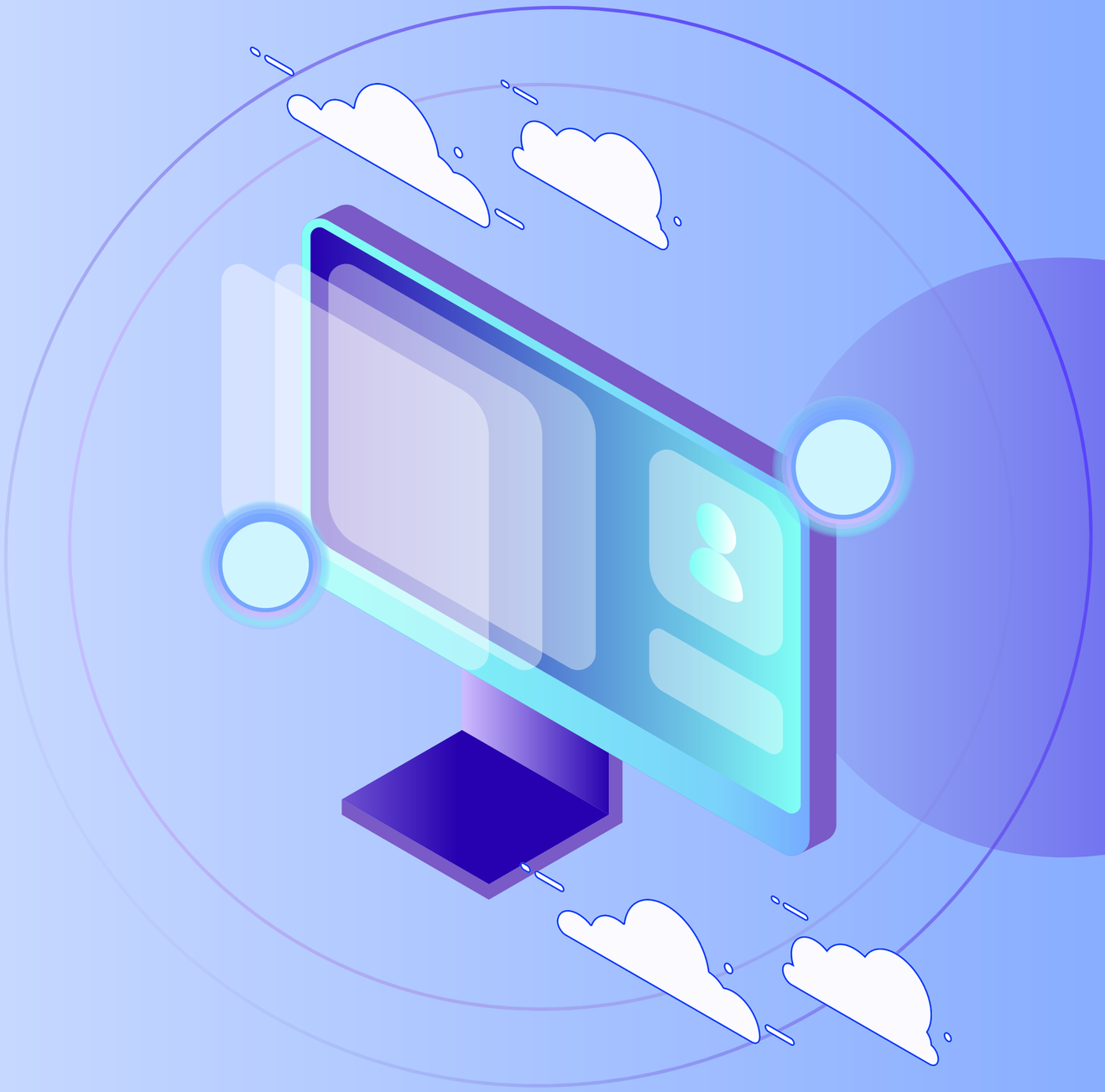
ESTRUCTURA DE DATOS EN PYTHON

ARBOLES



Alumnos (Comisión 4)

- Ovelar Javier
- Rodriguez Karina

¿QUE ES UN ARBOL?

En Python, un árbol es una estructura de datos no lineal en la que cada nodo puede apuntar a uno o varios nodos.

Tiene forma de árbol invertido ya que su nodo inicial llamado raíz esta en la parte superior y de él se desprenden nodos formando las ramas del árbol.

Es ideal para representar información de forma jerárquica, representando relaciones y niveles entre los elementos.

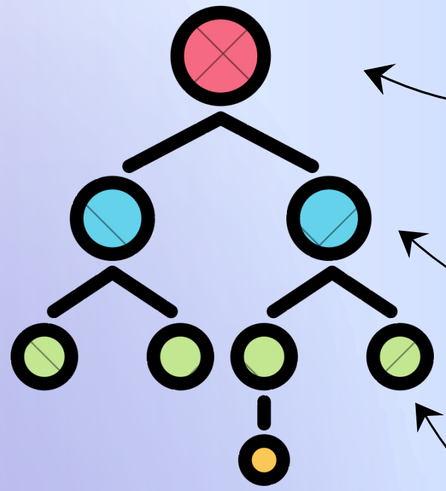
En programación, entender como funcionan es esencial para mejorar la eficiencia de los algoritmos y la optimización de recursos.

IMPORTANCIA

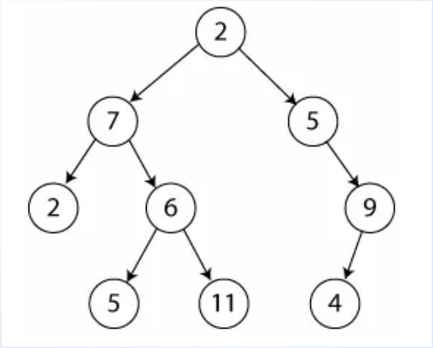
Es una estructura fundamental ya que tiene una amplia variedad de aplicaciones, por ejemplo:

- Sistemas de archivos
- Arboles genealógicos
- Bases de datos jerárquicas
- Algoritmos de búsqueda y toma de decisiones
- IA

ELEMENTOS CLAVES



- **Nodo Raíz**: es el único punto de entrada a la estructura, ubicado en la parte superior del árbol.
- **Nodo Rama o interno**: es cualquier nodo que tenga padre y al menos un hijo. Propaga la jerarquía del árbol.
- **Nodo Hoja**: es cualquier nodo que no tenga hijos, es decir, un nodo terminal del árbol.



ARBOLES BINARIOS

Es un tipo especial de árbol donde cada nodo puede tener como máximo 2 hijos (grado 2):

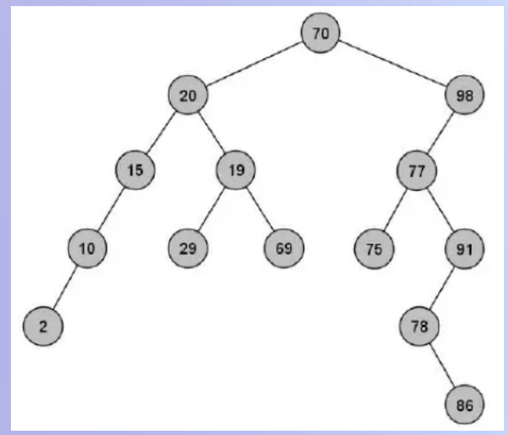
- Hijo izquierdo
- Hijo derecho

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

ARBOL DE BUSQUEDA BINARIA

Se cumplirá:

- Todos los valores de los nodos del subarbol izquierdo serán menores al nodo raíz
- Todos los valores de los nodos del subarbol derecho serán mayores al nodo raíz



VENTAJAS

Su eficiencia en búsquedas se debe a que en cada paso descarta la mitad de los elementos restantes, tomando considerablemente menos tiempo que en una búsqueda lineal en una lista.

Al igual que en la búsqueda, las operaciones de inserción y eliminación son más rápidas, permitiendo agregar o quitar elementos de forma más eficiente.

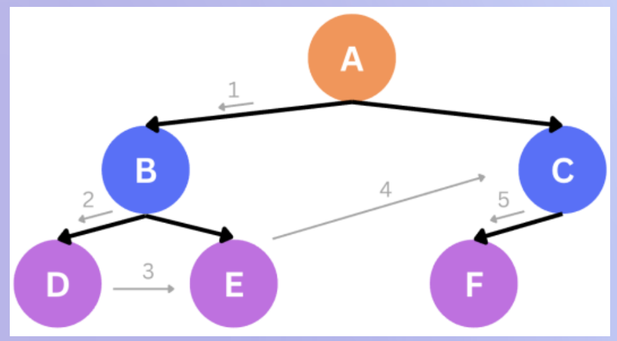
DESVENTAJAS

Si las inserciones no están bien balanceadas el árbol puede perder eficiencia.
Para mantener la eficiencia se deben aplicar algoritmos de balanceo como en árboles AVL.

RECORRIDOS

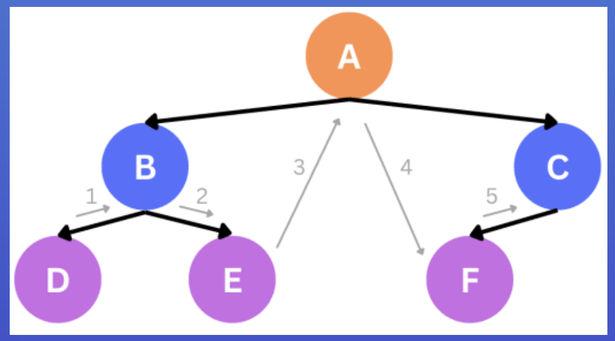
PREORDEN

Es útil cuando necesitas realizar alguna operación en el nodo antes de procesar a sus hijos.
Se comienza por la raíz, luego se recorre el subarbol izquierdo y finalmente el subarbol derecho.



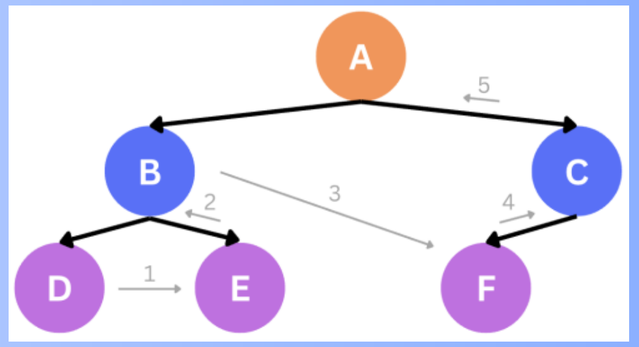
INORDEN

Es útil para los árboles de búsqueda binaria, ya que garantiza que los nodos se visiten en orden ascendente.
Procesa el subárbol izquierdo, luego el nodo actual y finalmente el subárbol derecho.



POSTORDEN

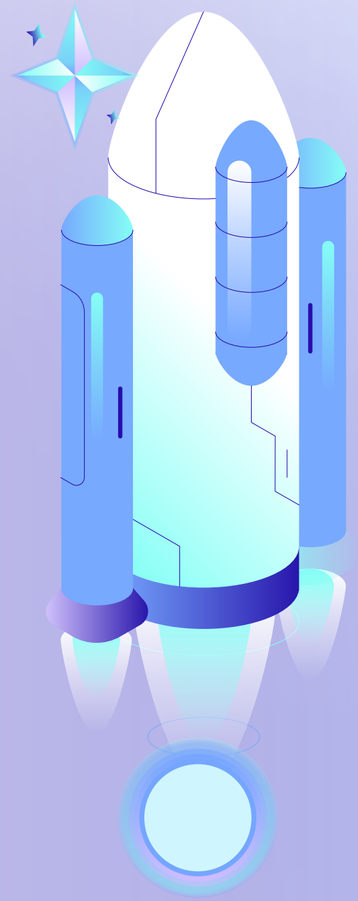
Es útil cuando quieres realizar alguna acción en los nodos hijos de un nodo antes de procesar el nodo mismo.
Común en estructuras jerárquicas.
Primero se visita el subárbol izquierdo, luego el subárbol derecho, y finalmente la raíz.

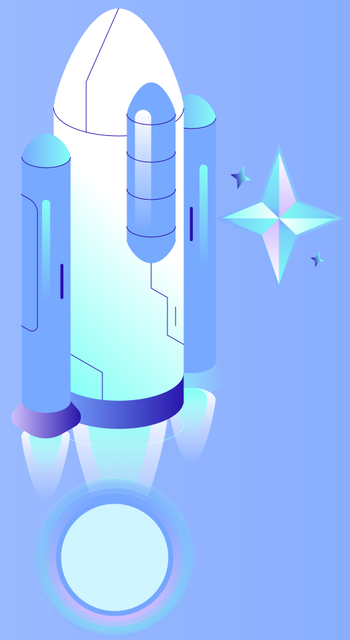


RESULTADOS OBTENIDOS

En su conjunto, el proyecto logró su objetivo principal de demostrar el funcionamiento básico de un árbol binario en un contexto práctico de gestión de datos.
Sirvió como una clara ilustración de cómo estas estructuras permiten organizar, buscar y manipular información de manera eficiente, superando las limitaciones de otras estructuras lineales para ciertos escenarios.

CONCLUSIONES





A partir de la metodología propuesta y el análisis del código se pueden extraer las siguientes conclusiones:

1. Aplicación práctica de Estructuras de Datos Complejas
2. Desarrollo y aplicación de algoritmos básicos para la manipulación y recorrido de árboles
3. Eficiencia en la gestión de datos
4. Potencial de mejora en Balanceo

GRACIAS!

Ovelar Javier
Rodriguez Karina