

Programacion1

Integrantes

El Trabajo Practico Integrador: **Estructuras de datos avanzados: Arboles**, de la materia Programacion 1 esta conformado por Alumnos de la Comisión 14:

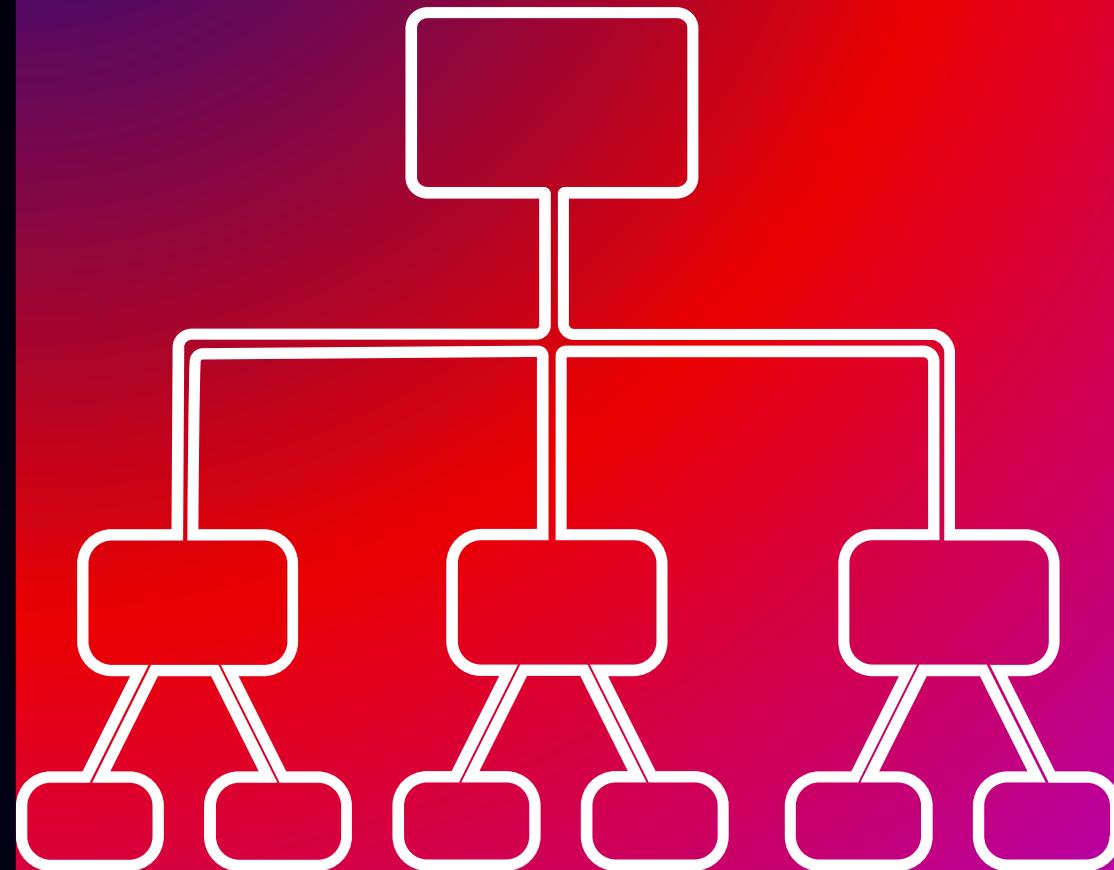


Garcia Ramiro



Godoy Tomas

Árbol



Introducción

Datos Avanzados: Arboles

En problemas complejos, las estructuras de datos avanzadas como los árboles son clave para organizar información jerárquica y realizar operaciones eficientes.

Aunque Python ofrece bibliotecas específicas, es posible representar árboles de forma didáctica usando listas: cada nodo es una lista con su valor y sus hijos.

Este enfoque simple permite comprender cómo funcionan internamente los árboles y cómo manipularlos desde cero.

Marco Teórico

Las estructuras de datos avanzadas son clave para diseñar algoritmos eficientes y resolver problemas complejos.

Entre ellas, los árboles destacan por su capacidad para representar relaciones jerárquicas y su uso en áreas como motores de búsqueda, sistemas de archivos o inteligencia artificial.

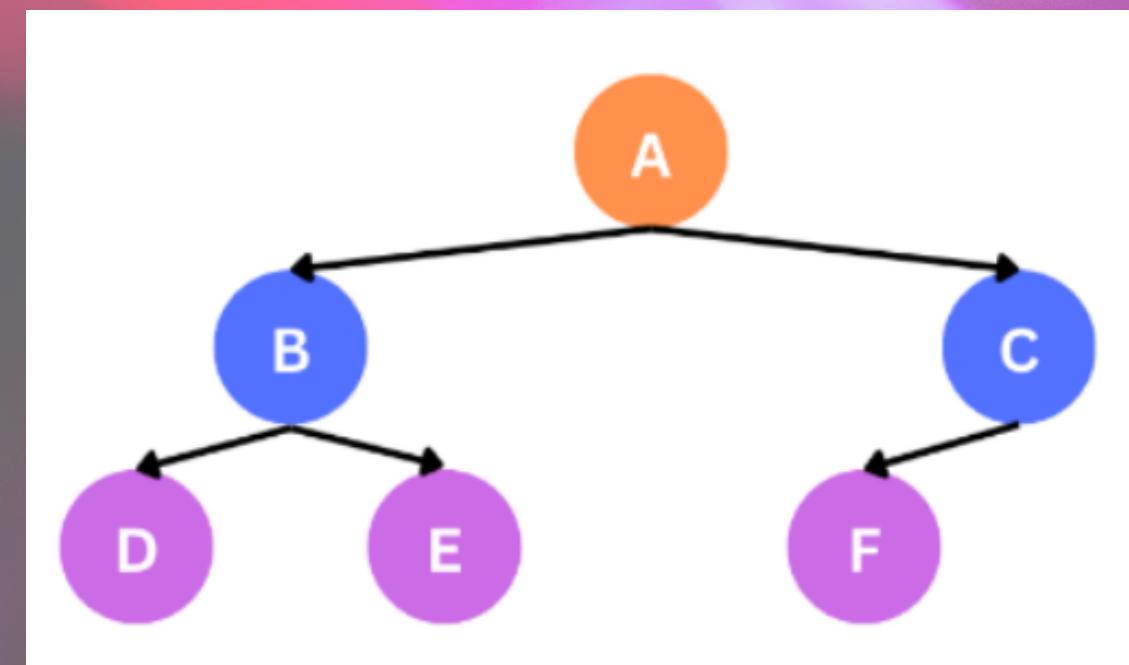
Un árbol es una estructura no lineal formada por nodos conectados:

Tiene una raíz como punto de partida.

- Cada nodo (salvo la raíz) tiene un único parente.
- Puede tener varios hijos.
- No tiene ciclos.

El árbol binario es una forma simplificada de árbol donde cada nodo tiene como máximo dos hijos: uno izquierdo y uno derecho.

Esta estructura es muy versátil y sirve de base para muchos algoritmos eficientes, especialmente en búsqueda y ordenamiento.



Si bien Python no incluye en su biblioteca estándar un tipo de datos nativo para árboles binarios, existen múltiples formas de representarlos. Las más comunes son a través del uso de listas, clases y objetos.

Propiedades de los Arboles

Camino y Longitud

Un camino es la secuencia de ramas entre dos nodos.

Solo se puede "caminar" por ramas existentes.

La longitud del camino es la cantidad de ramas recorridas.

Profundidad

Es la cantidad de ramas desde la raíz hasta un nodo.

La raíz tiene profundidad 0.

Cada nivel inferior suma +1.

Nivel

Es la profundidad + 1.
La raíz está en el nivel 1.

Es útil para visualizar el árbol como pisos de un edificio.

Altura del Árbol

Es el nivel más alto que contiene nodos.
Se mide desde la raíz hasta el nodo más profundo.

Grado y Orden

Grado de un nodo: cantidad de hijos que tiene.

Grado del árbol: máximo grado entre todos los nodos.

Orden del árbol: máximo número de hijos permitido por nodo (definido antes de construirlo).

Peso

Es la cantidad total de nodos del árbol.

Importante para:
Medir uso de memoria.

Evaluar tiempos de ejecución.

Balancear el árbol para mejorar rendimiento.

El programa desarrollado permite representar un árbol jerárquico utilizando listas en Python. A través de esta estructura, se modelan relaciones entre nodos, como padre, hijos y nietos. Además, se implementaron funciones para calcular propiedades fundamentales de los árboles, como el grado de un nodo, el grado del árbol, y se imprimió la estructura de forma visual, facilitando su comprensión. Este enfoque es útil para introducir los conceptos básicos de árboles sin necesidad de usar clases u otras estructuras más complejas.

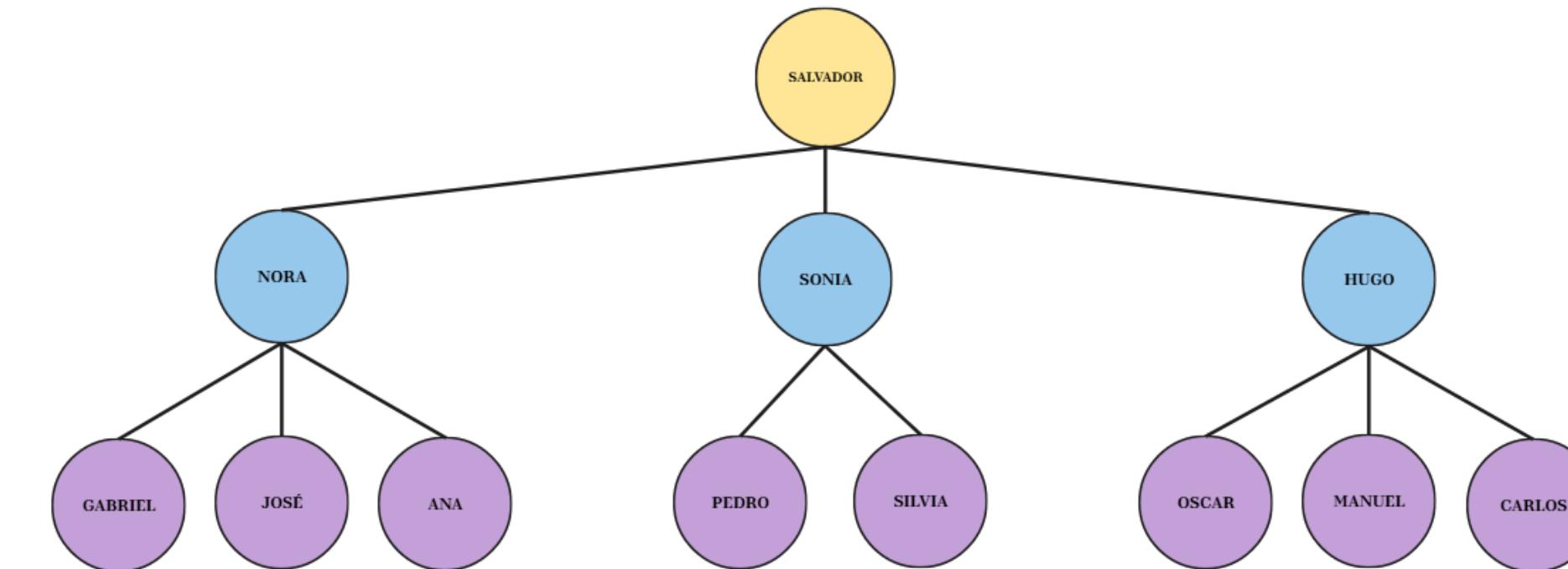
Caso Practico



≡ VAMOS ≡
AL CODIGO ≡

Listas Anidadas *

Diagrama de Arbol



```
[ "Salvador",
  [ "Nora",
    [ "Gabriel", [] ],
    [ "Jose", [] ],
    [ "Ana", [] ]
  ],
  [ "Sonia",
    [ "Pedro", [] ],
    [ "Silvia", [] ]
  ],
  [ "Hugo",
    [ "Oscar", [] ],
    [ "Manuel", [] ],
    [ "Carlos", [] ]
  ]
]
```

],

```
[ "Sonia",
  [ "Pedro", [] ],
  [ "Silvia", [] ]
]
```

],

```
[ "Hugo",
  [ "Oscar", [] ],
  [ "Manuel", [] ],
  [ "Carlos", [] ]
]
```

]

]

Conclusion



El estudio de los árboles binarios y su implementación, incluso con una estructura tan básica como las listas de Python, es clave para comprender cómo se organizan y procesan datos jerárquicos. Aunque las listas no son la opción más eficiente para aplicaciones reales que requieren árboles binarios, permiten entender los conceptos fundamentales de forma clara. Cuando se necesiten soluciones más robustas y escalables, es recomendable migrar a implementaciones basadas en clases o utilizar bibliotecas especializadas.

GRACIAS