

Trabajo Práctico Integrador – Datos Avanzados

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

Programación I

Alumnos - Comision 3

Joaquin Ignacio Juarez - joaquinignaciojuarez@gmail.com

Tahiel Noé Heinze – tahielnoeheinze@gmail.com

Profesor

Julieta Trapé

Tutor

Tomás Ferro

Fecha de Entrega

09 de Junio de 2025

Índice

1. Introducción
 2. Marco teórico
 3. Caso práctico
 4. Metodología Utilizada
 5. Resultados Obtenidos
 6. Conclusiones
 7. Bibliografía
 8. Anexo
 9. Link al video de Youtube
-

1. Introducción

- Elegimos estudiar árboles con listas como estructura de datos avanzada por su potencial para manejar información compleja de manera eficiente y su aplicación práctica en desarrollo de software.
 - Este tema tiene relevancia en la Programación debido a que es clave utilizarlo para:
 - Optimizar algoritmos.
 - Reducir tiempos de ejecución.
 - Gestionar datos jerárquicos.
 - El objetivo final del trabajo es el de aprender un concepto totalmente "nuevo" para nosotros y a su vez:
 - Entender la implementación de los datos avanzados.
 - Analizar su impacto en eficiencia y sentar las bases para futuras puestas en prácticas.
-

2. Marco teórico

- **Árbol:** Es una estructura de datos no lineal en la que cada nodo puede apuntar a uno o varios nodos
- **Nodo:** Unidad básica de un árbol
- **Nodo raíz:** es el punto de entrada a la estructura, generalmente ubicado en la parte más superior del árbol.
- **Nodo hoja:** es cualquier nodo que no tenga hijos.

- **Nodo rama o interno:** es cualquier nodo que tiene un padre y, al menos, un hijo.
- **Nodo Padre:** aquél nodo que tiene uno o más nodos hijos conectados a él.
- **Nodo hijo:** aquél nodo que está conectado con un nodo padre de modo que se encuentra directamente debajo de él en la jerarquía.
- **Nodo hermano:** aquellos nodos que comparten el mismo nodo padre.

Propiedades de los árboles

- **Longitud de un camino:** número de ramas o arcos que hay que transitar para llegar de un nodo a otro.
 - **Profundidad:** longitud de camino entre el nodo raíz y él mismo.
 - **Altura:** máximo nivel del mismo.
 - **Nivel:** longitud del camino que lo conecta al nodo raíz más uno.
 - **Grado:** número de hijos que tiene dicho nodo
 - **Orden:** máxima cantidad de hijos que puede tener cada nodo
 - **Peso:** número total de nodos que tiene un árbol.
-

3. Caso Práctico

Para aplicar lo aprendido sobre “datos avanzados ” decidimos realizar un árbol binario en el cual:

- El usuario ingresará valores numéricos.
 - El sistema organizará los datos automáticamente.
 - Se mostrarán los resultados mediante recorrido in-order.
-

4. Metodología Utilizada

Se decidió seguir un enfoque estructurado y modular, combinando la programación orientada a objetos y recursividad para lograr que el código sea eficiente y con claridad en la implementación.

Definimos un árbol binario el cual se le pedirá al usuario que agregue los datos necesarios para ir completandolo, como la raíz y sus hijos, luego se mostrará su recorrido en formato in-order (mostrando los valores de: izquierda → raíz → derecha)

5. Resultados Obtenidos

Al ejecutar el programa, el usuario podrá observar dos tipos de resultados claramente estructurados.

Primero, una representación gráfica del árbol binario que muestra la jerarquía de nodos mediante símbolos (└─, ┆─), donde la raíz aparece en la parte superior y los nodos hijos se van desplegando de forma indentada según su nivel en la estructura.

Segundo, se mostrará el recorrido in-order de los valores, presentados automáticamente en orden ascendente. Esta salida demuestra cómo el árbol organiza internamente los datos ingresados.

6. Conclusiones

Este proyecto nos permitió adentrarnos en las estructuras de datos avanzadas, específicamente en el funcionamiento de los árboles binarios de búsqueda.

A través de la implementación pudimos trabajar con una estructura no lineal y eficiente para organizar información, diferente a las estructuras lineales que habíamos usado antes.

Durante el desarrollo aplicamos conceptos de “class” en Python, lo que nos ayudó a entender mejor cómo encapsular lógica y datos de manera ordenada.

También implementamos recursividad para operaciones como inserción y recorrido, que al principio nos resultó un poco desafiante pero luego se volvió más claro. La creación de la visualización interactiva fue particularmente útil porque hizo más tangible un concepto que puede resultar abstracto.

Lo más valioso fue comprender cómo estas estructuras optimizan procesos en sistemas reales, desde motores de búsqueda hasta aplicaciones de inteligencia artificial, y ver su aplicación práctica más allá del ámbito académico.

7. Bibliografía

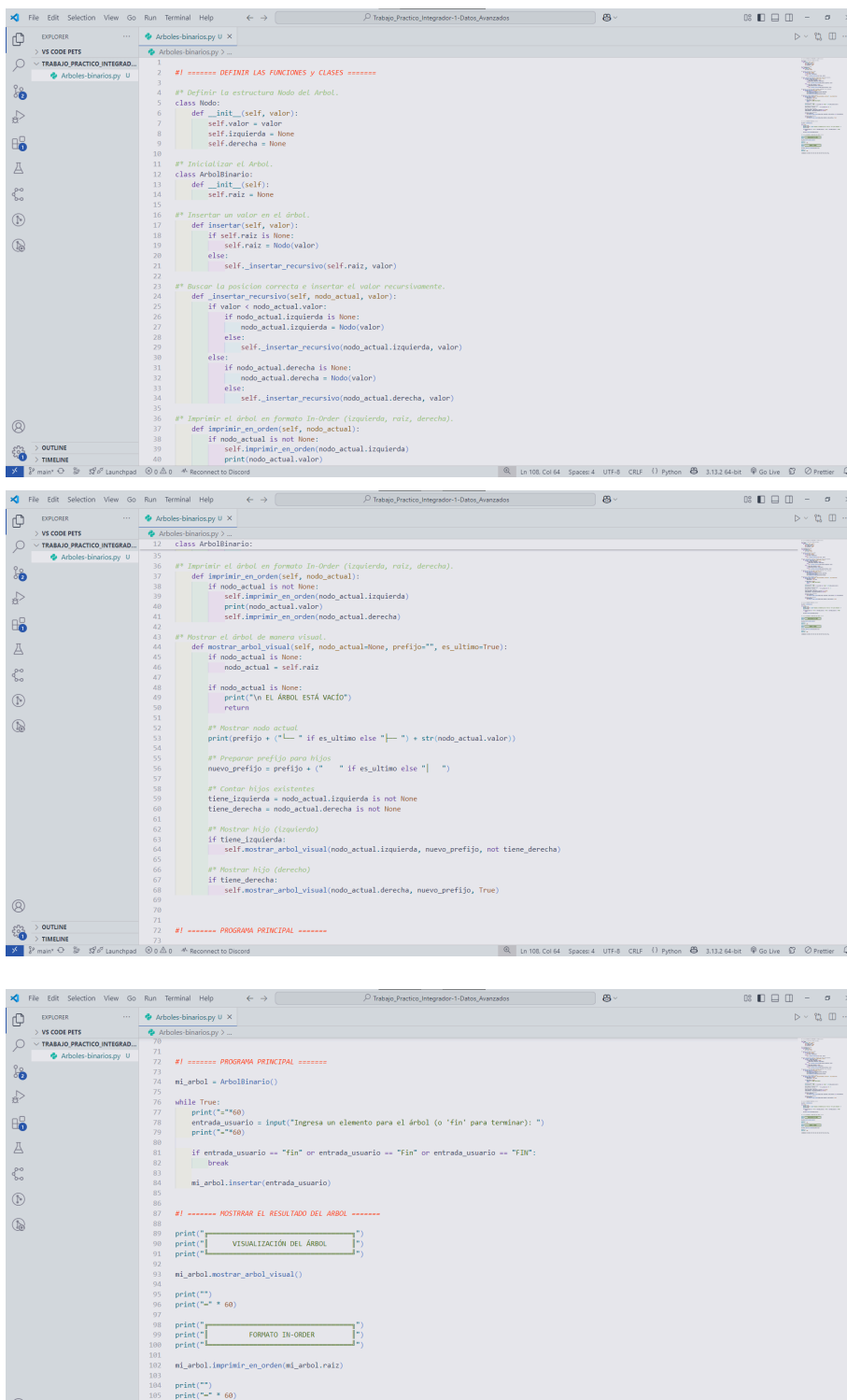
- *Programación orientada a objetos en Python. El Libro de Python.*
<https://ellibrodepython.com/programacion-orientada-a-objetos-python>
- *Material didáctico sobre Árboles como estructura de datos.*
https://docs.google.com/document/d/10k16oL15EeyOaq92aoi4qwK3t_22X29-FSV2iV-8N1U/edit?tab=t.0
- *Canal: Tecnicatura - Árboles como estructura de datos:*
https://www.youtube.com/watch?v=cVm51pO35qA&list=PLy5wpwhsM-2IIY-qe_fALJ4K_XAhLZ2I-&index=4
<https://www.youtube.com/watch?v=kqmn7AYYdSA>

https://www.youtube.com/watch?v=I4IKGp0PsO0&list=PLy5wpwhsM-2IIY-qe_fALJ4K_XAhLZ2l-&index=5

https://www.youtube.com/watch?v=O5qvB-LWyhY&list=PLy5wpwhsM-2IIY-qe_fALJ4K_XAhLZ2l-&index=4

<https://www.youtube.com/watch?v=-D4SxeHQQIq>

8. Anexo



```

1  #! ===== DEFINIR LAS FUNCIONES Y CLASES =====
2
3  #! Definir la estructura Nodo del Arbol.
4  class Nodo:
5      def __init__(self, valor):
6          self.valor = valor
7          self.izquierda = None
8          self.derecha = None
9
10 #! Inicializar el Arbol.
11 class ArbolBinario:
12     def __init__(self):
13         self.raiz = None
14
15 #! Insertar un valor en el árbol.
16 def insertar(self, valor):
17     if self.raiz is None:
18         self.raiz = Nodo(valor)
19     else:
20         self._insertar_recursivo(self.raiz, valor)
21
22 #! Buscar la posición correcta e insertar el valor recursivamente.
23 def _insertar_recursivo(self, nodo_actual, valor):
24     if valor < nodo_actual.valor:
25         if nodo_actual.izquierda is None:
26             nodo_actual.izquierda = Nodo(valor)
27         else:
28             self._insertar_recursivo(nodo_actual.izquierda, valor)
29     else:
30         if nodo_actual.derecha is None:
31             nodo_actual.derecha = Nodo(valor)
32         else:
33             self._insertar_recursivo(nodo_actual.derecha, valor)
34
35 #! Imprimir el árbol en formato In-Order (izquierda, raíz, derecha).
36 def imprimir_en_orden(self, nodo_actual):
37     if nodo_actual is not None:
38         self._imprimir_en_orden(nodo_actual.izquierda)
39         print(nodo_actual.valor)
40         self._imprimir_en_orden(nodo_actual.derecha)
41
42 #! Mostrar el árbol de manera visual.
43 def mostrar_arbol_visual(self, nodo_actual=None, prefijo="", es_ultimo=True):
44     if nodo_actual is None:
45         nodo_actual = self.raiz
46
47     if nodo_actual is None:
48         print("En El ÁRBOL ESTÁ VACÍO")
49         return
50
51     #! Mostrar nodo actual
52     print(prefijo + ("└─ " if es_ultimo else "├─ ") + str(nodo_actual.valor))
53
54     #! Preparar prefijo para hijos
55     nuevo_prefijo = prefijo + (" " if es_ultimo else "| ")
56
57     #! Contar hijos existentes
58     tiene_izquierda = nodo_actual.izquierda is not None
59     tiene_derecha = nodo_actual.derecha is not None
60
61     #! Mostrar hijo (izquierda)
62     if tiene_izquierda:
63         self.mostrar_arbol_visual(nodo_actual.izquierda, nuevo_prefijo, not tiene_derecha)
64
65     #! Mostrar hijo (derecha)
66     if tiene_derecha:
67         self.mostrar_arbol_visual(nodo_actual.derecha, nuevo_prefijo, True)
68
69 #! ===== PROGRAMA PRINCIPAL =====
70
71 mi_arbol = ArbolBinario()
72
73 while True:
74     print("\n=====")
75     entrada_usuario = input("Ingresa un elemento para el árbol (o 'fin' para terminar): ")
76     print("\n=====")
77
78     if entrada_usuario == "fin" or entrada_usuario == "Fin" or entrada_usuario == "FIN":
79         break
80
81     mi_arbol.insertar(entrada_usuario)
82
83 #! ===== MOSTRAR EL RESULTADO DEL ÁRBOL =====
84
85 print("\n=====")
86 print("VISUALIZACIÓN DEL ÁRBOL")
87 print("\n=====")
88
89 mi_arbol.mostrar_arbol_visual()
90
91 print("\n=====")
92 print("FORMATO IN-ORDER")
93 print("\n=====")
94
95 mi_arbol.imprimir_en_orden(mi_arbol.raiz)
96
97 print("\n=====")
98 print("FIN")
99 print("\n=====")
100

```

9. Link al video de Youtube

<https://youtu.be/d2JmphgpazY>

10. Link al repositorios de GitHub

https://github.com/joaaajrz/Tp_Integrador-Programacion1