¡BIENVENIDOS!

TRABAJO PRÁCTICO – ÁRBOLES BINARIOS DE BÚSQUEDA

ALUMNAS: PASCUTTI VALENTINA Y RAMOS ANGELA.

INTRODUCCIÓN

En informática, las estructuras de datos son esenciales para organizar y manejar información de forma eficiente. Entre ellas, los **árboles** destacan por su capacidad para representar relaciones jerárquicas y múltiples niveles de dependencia, como sucede en sistemas de archivos, bases de datos o algoritmos de búsqueda.

Este trabajo se centra en los árboles como estructuras avanzadas, abordando una implementación particular: el uso de **listas anidadas**.

ALGUNAS DEFINICIONES CLAVE:

Árbol: Es una estructura de datos **no lineal**, donde cada nodo puede conectarse con varios otros.
A diferencia de las listas, su forma es **ramificada**, no secuencial.

Árbol binario: Es un tipo de árbol donde **cada nodo tiene hasta dos hijos**: uno izquierdo y uno derecho.
Se le conoce como árbol de **grado dos**.

Grafo: Estructura que representa **relaciones entre objetos**, usando **nodos** (puntos) y **aristas** (conexiones).

PROPIEDADES DE LOS ÁRBOLES

- Longitud
- Altura
- Peso
- Profundidad
- Grado
- Nivel
- Orden

FORMAS DE RECORRER ARBOLES BINARIOS

- PREORDEN
- INORDEN
- POSTORDEN

CASO PRÁCTICO

Crear un Árbol Binario de Búsqueda a partir de una lista de valores enteros ingresados por el usuario. Además, implementar un recorrido postorden y dibujar visualmente la estructura del árbol en la consola.

PROGRAMA EN PYTHON

arbol_binario.py > main

```python
# Programa: Árbol Binario de Búsqueda.
# Crear un árbol binario de búsqueda a partir de una lista de valores enteros ingresados por el usuario.
# Implementar un recorrido postorden y una función para dibujar el árbol de forma visual.

class Nodo:
    """Clase que representa un nodo en un árbol binario de búsqueda."""
    def __init__(self, valor):
        self.valor = valor
        self.izq = None
        self.der = None

def insertar(raiz, valor):
    """Inserta un nuevo valor en el árbol binario de búsqueda."""
    """Si el árbol está vacío, crea un nuevo nodo. Si no, inserta recursivamente."""
    if raiz is None:
        return Nodo(valor)
    if valor < raiz.valor:
        raiz.izq = insertar(raiz.izq, valor)
    else:
        raiz.der = insertar(raiz.der, valor)
    return raiz

def postorden(nodo):
    """Realiza un recorrido en postorden del árbol."""
    if nodo:
        postorden(nodo.izq)
        postorden(nodo.der)
        print(nodo.valor, end=' ')

```

1

```python
def dibujar_arbol(nodo, prefijo="", es_izq=True):
    """Dibuja el árbol binario de búsqueda de forma visual (de arriba hacia abajo)."""
    if nodo is not None:
        if nodo.der:
            dibujar_arbol(nodo.der, prefijo + "    ", False)
        print(prefijo + ("└── " if es_izq else "┌── ") + str(nodo.valor))
        if nodo.izq:
            dibujar_arbol(nodo.izq, prefijo + "    ", True)

```

2

```python
# Programa principal.
def main():
    """Función principal que solicita al usuario los valores del árbol y ejecuta las operaciones."""
    print("Bienvenido al programa de Árbol Binario de Búsqueda.")

    entrada = input("Ingresá los valores del árbol separados por comas (ej: 8,3,10,1,6): ")

    valores = []
    for v in entrada.split(","):
        v = v.strip()
        if v.isdigit() or (v.startswith('-') and v[1:].isdigit()):
            valores.append(int(v))
        else:
            print(f"Advertencia: '{v}' no es un número válido y será ignorado.")

    if not valores:
        print("No se ingresaron valores válidos.")
        exit()

    raiz = None
    for v in valores:
        raiz = insertar(raiz, v)

    print("\nRecorrido postorden:")
    postorden(raiz)

    print("\n\nÁrbol binario representado visualmente:")
    dibujar_arbol(raiz)

if __name__ == "__main__":
    main()
```

3

RESULTADOS OBTENIDOS

- Se construyó un **árbol binario de búsqueda** funcional.
- Se permite la **inserción dinámica** de valores (incluyendo negativos), con validación.
- El **recorrido postorden** funciona correctamente.
- Se implementó una **visualización en consola** clara y didáctica del árbol.
- El programa cumple con su objetivo como herramienta educativa.

CONCLUSIÓN

Este trabajo permitió aplicar y entender los árboles binarios como estructuras jerárquicas, logrando su construcción, recorrido y visualización. La experiencia reforzó su importancia en la organización de datos y en el desarrollo de algoritmos eficientes.

¡GRACIAS!