

ESCUELA SUPERIOR DE COMPUTO
“ESCOM”

ARBOLES BINARIOS

REPORTE TECNICO

- RUPERTO HERMANDEZ DIEGO
- VENEGAS MARTINEZ MARIA JOSE

ICM5

MATEMATICAS DISCRETAS

FECHA DE ENTREGA: 6 DE ENERO DE 2025

Índice

1. Objetivo del proyecto.....	3
1. Introducción al tema.....	3
2. Implementación practica	3
3. Pruebas y ejecución.....	3
4. Resultado esperado.....	3
2. Introducción teórica.....	4
1. Arboles binarios.....	4
2. Clasificación de nodos.....	5
3. Recorrido pre-orden.....	6
4. Recorrido in-orden.....	7
5. Recorrido post-orden.....	7
3. Desarrollo del sistema.....	9
4. Pruebas y resultados.....	15
1. Prueba 1.....	15
2. Resultado 1.....	16
3. Prueba 2	16
4. Resultado 2.....	17
5. Conclusiones.....	18
6. Bibliografías.....	19

Objetivo del proyecto

Nuestro objetivo al realizar este proyecto es entender el tema de los árboles binarios, desde sus fundamentos teóricos hasta poder hacer una implementación práctica en código. Nuestro proyecto abarcará distintos temas, como lo son:

1. Introducción al tema y fundamentos de este mismo:

- Comprender todo el concepto de árboles binarios, incluyendo sus definiciones, la explicación de todos sus recorridos, características y la terminología asociada con ejemplos y explicación de cada rubro.
- Establecer la estructura de los nodos y de un árbol binario

2. Implementación practica de los árboles binarios

- Creación de un árbol binario
- Implementar los conceptos de estructura de nodos y de árboles en forma de código
- Implementar los tres tipos de recorrido: preorden, inorden y postorden

3. Pruebas y ejecución:

- Probar la implementación con ejemplos prácticos
- Mostrar pantallas de salida y gráficos correctos

4. Resultado esperado

Al final del proyecto, se contará con un código funcional que permita construir, manipular y recorrer arboles binarios, junto con una sólida comprensión teórica del tema.

Introducción teórica

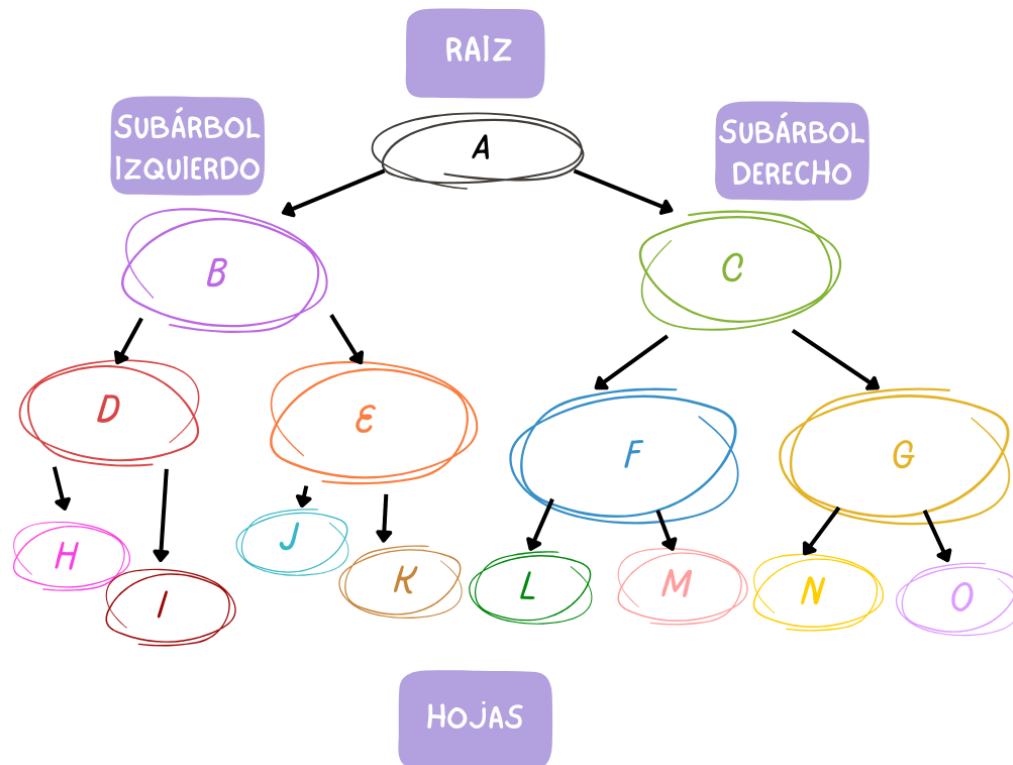
Arboles binarios

Estos se pueden definir como una estructura de datos que es utilizada en computación, esta inicia con una raíz y de esta se extienden dos ramificaciones hasta que estas terminan en una hoja. De igual manera en cómo lo hace un árbol. En un lenguaje más técnico, se podría decir que un árbol binario inicia con un nodo que funciona como raíz, de este nodo se originan los nuevos nodos o ramificaciones para formar esta hoja. Cada una de estas raíces puede tener dos ramificaciones, por esta razón se le denomina árbol binario.

Esta estructura de datos es diferente a las lineales, ya que relaciona la información de forma jerárquica, por eso se denomina árbol y es binario porque solo se desprenden dos ramas de cada raíz o nodo formado.

Volviendo un poco al lenguaje técnico, un árbol binario podría ser explicado de la siguiente manera; Los árboles binarios son estructuras muy similares a las listas, en este caso, a las listas doblemente enlazadas, en el sentido de que ambos tienen dos punteros que apuntan a otros elementos, pero no tienen una estructura de tipo lineal o secuencial como las listas, si no ramificada. En resumen; un árbol binario es una estructura de datos no lineal en la que cada nodo puede apuntar a uno o máximo a dos nodos, hay casos en los que se da la definición recursiva de que es una estructura compuesta por un dato y dos árboles.

De manera gráfica podemos ver la estructura general del árbol binario; A las ramas también se les puede dar el nombre de sub arboles ya que en estas también se pueden desprender ramificaciones que forman a su vez hojas, en este caso tenemos otro ejemplo de cómo sería la estructura de un árbol binario:



Para explicar esta representación, debemos saber que un árbol binario representa un conjunto finito de elementos divididos en tres partes separadas o subconjuntos, y como ya lo habíamos mencionado antes a cada elemento que forma este algoritmo se llama nodo del árbol y cuando un nodo tiene un nodo hijo o un subárbol, se le conoce como hoja. Otra forma para explicar su estructura podría ser:

- **Raíz:** Este representa el primer subconjunto y solo tiene un elemento.
- **Subárbol izquierdo:** Representa un segundo conjunto, se le conoce como subárbol por venir o desprenderse del árbol original.
- **Subárbol derecho:** Representa un tercer conjunto, se le conoce como subárbol por venir o desprenderse del árbol original.

Clasificación de nodos

Los nodos de los árboles binarios se pueden clasificar de la siguiente manera:

- **Nodo padre:** Este nodo es el que origina otros nodos a los que se les denomina hijos, este nodo podría decirse que es el primero en ser creado, este no tiene padre ni se origina de otro.
- **Nodo rama:** Este nodo tiene hijos y padre.
- **Nodo hoja:** Este es un nodo que tiene padre, pero no tiene hijos, se podría decir que este nodo es el último en ser creado y no deriva ninguno de él.

Esta clasificación también se puede especificar de manera más técnica.

- **Nodo hijo:** Este puede ser cualquier nodo apuntado por uno de los nodos del árbol.

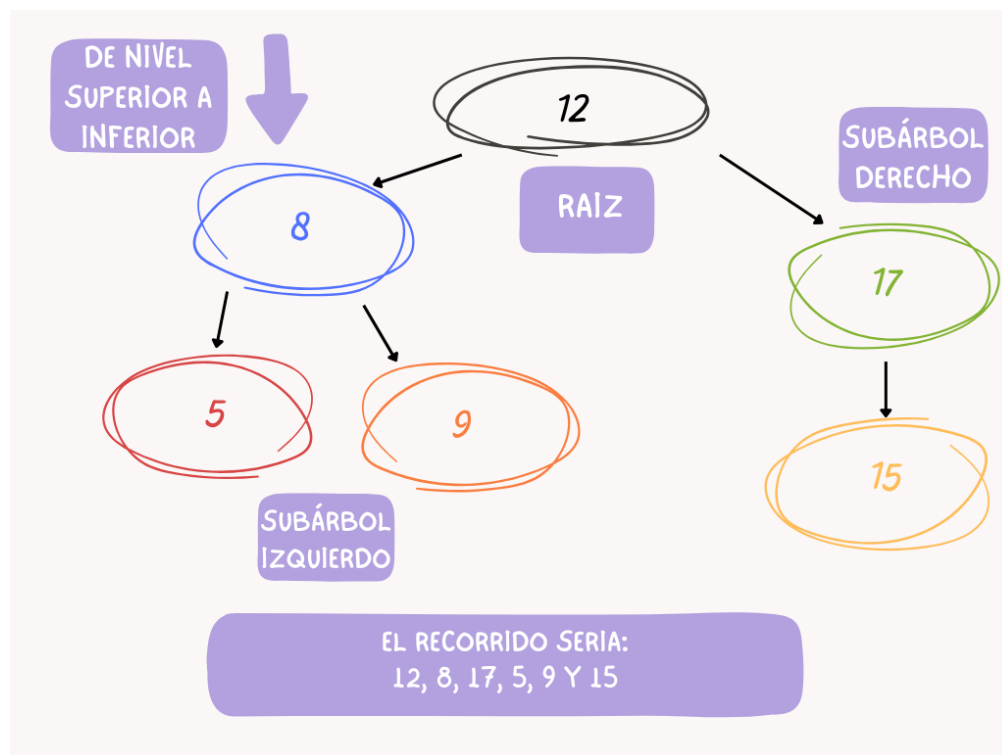
- **Nodo padre:** Este nodo contiene un puntero al nodo actual y apunta hacia otros nodos, estos serían sus nodos hijo.
- **Nodo raíz:** Este nodo no cuenta con un nodo padre, este nodo se utiliza para referirnos al árbol, este si apunta a otros nodos
- **Nodo hoja:** Este nodo no apunta a ningún otro, pero si cuenta con uno a que apunta hacia este.

Otra característica de los árboles binarios es que pueden ser recorridos, este recorrido es el proceso de orden o secuencia que se utiliza para visitar los nodos que lo componen, el recorrido de un árbol se lleva a cabo en tres diferentes sentidos: Pre-orden, in-orden y pos-orden.

Recorrido en Pre-orden o Amplitud:

Este recorrido se realiza desde el nivel superior, para posteriormente ir bajando hasta los niveles inferiores. Para empezar, se examina el dato del nodo raíz, para posteriormente recorrer el subárbol izquierdo en pre-orden y finalmente el subárbol derecho de igual forma en pre-orden, es decir, que para cada subárbol se debe conservar el mismo pre-orden, primero la raíz, luego la parte izquierda y posteriormente la parte derecha.

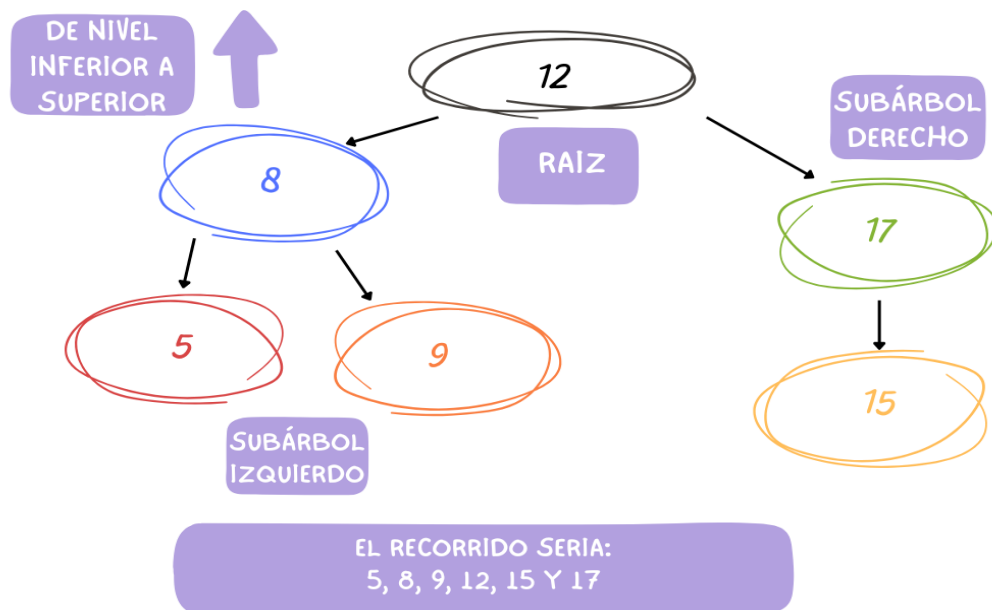
Para entender mejor esto podemos ver el siguiente ejemplo:



Recorrido en In-orden:

Este recorrido es un recorrido en pre-orden, pero se podría decir que, de manera invertida, para empezar, se recorre el subárbol izquierdo en in-orden, luego se examina el dato del nodo raíz y finalmente se recorre el subárbol derecho en in-orden. Esto quiere decir que para cada subárbol se debe conservar el recorrido en in-orden, primero se visita la parte izquierda, luego la raíz y posteriormente la parte derecha.

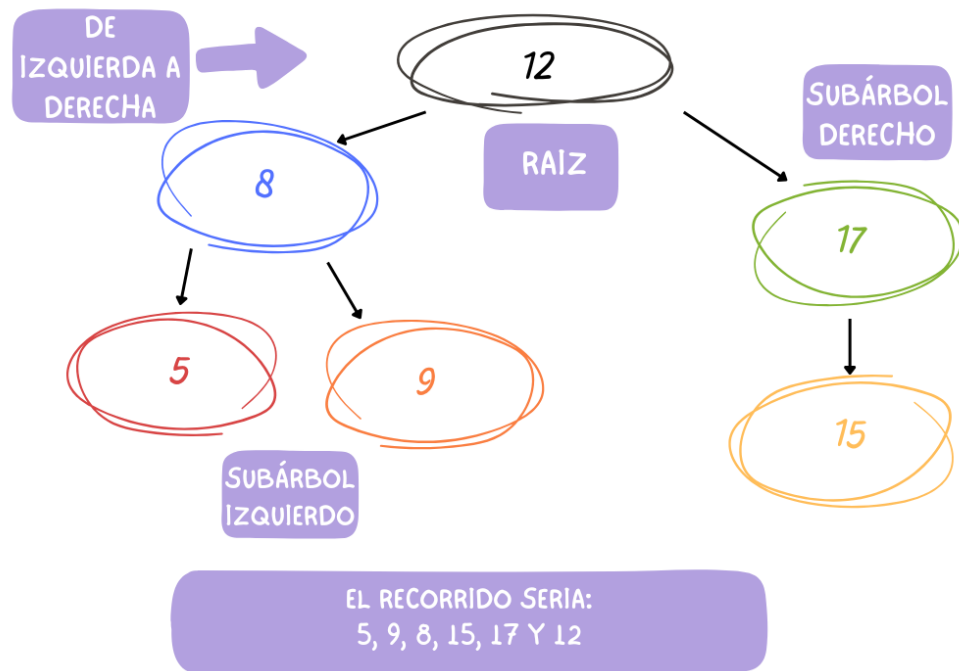
De igual manera podemos ver mejor este recorrido en el siguiente ejemplo:



Recorrido en Postorden:

Este recorrido consiste en recorrer el subárbol izquierdo en Postorden, luego el subárbol derecho de igual manera en Postorden y finalmente se visita el nodo raíz, es decir, primero se visitan los nodos de la parte izquierda, luego los de la parte derecha y por último la raíz.

Por último, tenemos el ejemplo de este recorrido:



Desarrollo del sistema

En el siguiente apartado agregaremos el código ya documentado, en este se explica de forma sencilla que se está realizando.

La explicación de cada una de las funciones se encuentra de color verde en forma de comentario dentro del Código.

```
1 import tkinter as tk
2 from tkinter import filedialog, messagebox, simpledialog
3 import networkx as nx
4 import matplotlib.pyplot as plt
5 import csv
6
7 # Clase que representa un nodo del árbol binario
8 class ArbolBinario:
9     def __init__(self, valor):
10         """Inicializa un nodo del árbol binario con un valor y punteros a sus hijos."""
11         self.valor = valor
12         self.izquierdo = None
13         self.derecho = None
14
15     def insertar(self, valor):
16         """Inserta un valor en el árbol binario respetando su estructura."""
17         if valor < self.valor:
18             if self.izquierdo is None:
19                 self.izquierdo = ArbolBinario(valor)
20             else:
21                 self.izquierdo.insertar(valor)
22         else:
23             if self.derecho is None:
24                 self.derecho = ArbolBinario(valor)
25             else:
26                 self.derecho.insertar(valor)
27
28     def preorden(self):
29         """Recorrido preorden del árbol binario."""
30         resultado = [self.valor]
31         if self.izquierdo:
32             resultado.extend(self.izquierdo.preorden())
33         if self.derecho:
34             resultado.extend(self.derecho.preorden())
35         return resultado
```

```

1  def inorden(self):
2      """Recorrido inorden del árbol binario."""
3      resultado = []
4      if self.izquierdo:
5          resultado.extend(self.izquierdo.inorden())
6      resultado.append(self.valor)
7      if self.derecho:
8          resultado.extend(self.derecho.inorden())
9      return resultado
10
11  def postorden(self):
12      """Recorrido postorden del árbol binario."""
13      resultado = []
14      if self.izquierdo:
15          resultado.extend(self.izquierdo.postorden())
16      if self.derecho:
17          resultado.extend(self.derecho.postorden())
18      resultado.append(self.valor)
19      return resultado
20
21  def construir_arbol_desde_lista(datos):
22      """Construye un árbol binario a partir de una lista de valores."""
23      if not datos:
24          return None
25      raiz = ArbolBinario(datos[0])
26      for valor in datos[1:]:
27          raiz.insertar(valor)
28      return raiz

```

```

1 def dibujar_arbol(raiz):
2     """Dibuja el árbol binario usando la librería NetworkX y matplotlib."""
3     def agregar_aristas(nodo, grafo, posiciones, x=0, y=0, nivel=1):
4         """Agrega los nodos y las aristas al grafo para representar el árbol."""
5         if nodo is not None:
6             grafo.add_node(nodo.valor)
7             posiciones[nodo.valor] = (x, y)
8             if nodo.izquierdo is not None:
9                 grafo.add_edge(nodo.valor, nodo.izquierdo.valor)
10                agregar_aristas(nodo.izquierdo, grafo, posiciones, x - 1 / nivel, y - 1, nivel + 1)
11            if nodo.derecho is not None:
12                grafo.add_edge(nodo.valor, nodo.derecho.valor)
13                agregar_aristas(nodo.derecho, grafo, posiciones, x + 1 / nivel, y - 1, nivel + 1)
14
15    G = nx.DiGraph()
16    posiciones = {}
17    agregar_aristas(raiz, G, posiciones)
18
19    # Configuración y visualización del grafo
20    plt.figure(figsize=(8, 6))
21    nx.draw(G, pos=posiciones, with_labels=True, node_size=6000, node_color="pink", font_size=20, font_weight="bold", arrows=True)
22    plt.show()

```

```
1 def cargar_desde_csv():
2     """Carga una lista de valores desde un archivo CSV."""
3     ruta_archivo = filedialog.askopenfilename(filetypes=[("Archivos CSV", "*.csv")])
4     if not ruta_archivo:
5         return None
6
7     try:
8         with open(ruta_archivo, "r") as archivo:
9             lector = csv.reader(archivo)
10            datos = [int(fila[0]) for fila in lector if fila]
11            return datos
12    except Exception as e:
13        messagebox.showerror("Error", f"No se pudo cargar el archivo CSV: {e}")
14        return None
15
16 def entrada_manual():
17     """Permite al usuario ingresar una lista de valores manualmente."""
18     datos_entrada = simplifiedialog.askstring("Entrada", "Introduce números separados por comas:")
19     try:
20         return list(map(int, datos_entrada.split(",")))
21    except Exception as e:
22        messagebox.showerror("Error", f"Entrada no válida: {e}")
23        return None
```



```
1 def crear_arbol():
2     """Gestiona la creación del árbol binario a partir de datos proporcionados por el usuario."""
3     opcion = tk.messagebox.askquestion("Método de Entrada", "¿Quieres cargar datos desde un archivo CSV?")
4     if opcion == "yes":
5         datos = cargar_desde_csv()
6     else:
7         datos = entrada_manual()
8
9     if datos:
10        arbol = construir_arbol_desde_lista(datos)
11        dibujar_arbol(arbol)
12
13        # Mostrar recorridos en la ventana principal
14        preorden = arbol.preorden()
15        inorden = arbol.inorden()
16        postorden = arbol.postorden()
17
18        etiqueta_preorden.config(text=f"Preorden: {preorden}")
19        etiqueta_inorden.config(text=f"Inorden: {inorden}")
20        etiqueta_postorden.config(text=f"Postorden: {postorden}")
```

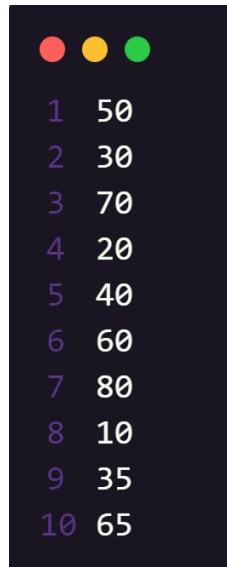


```
1 # Configuración de la interfaz gráfica
2 raiz = tk.Tk()
3 raiz.title("Árbol Binario")
4
5 # Marco principal
6 marco = tk.Frame(raiz, padx=20, pady=20)
7 marco.pack()
8
9 # Cambiar el ícono de la aplicación
10 raiz.iconbitmap("Arbolito.ico")
11
12 # Etiqueta del título
13 etiqueta = tk.Label(marco, text="Visualizador de Árbol Binario", font=("Sans", 80))
14 etiqueta.pack(pady=10)
15
16 # Botón para crear el árbol
17 boton = tk.Button(marco, text="Crear Árbol", command=crear_arbol, bg="lightgreen", font=("Sans", 40))
18 boton.pack(pady=10)
19
20 # Etiquetas para mostrar los recorridos
21 etiqueta_preorden = tk.Label(marco, text="Preorden: ", font=("Sans", 40))
22 etiqueta_preorden.pack(pady=5)
23
24 etiqueta_inorden = tk.Label(marco, text="Inorden: ", font=("Sans", 40))
25 etiqueta_inorden.pack(pady=5)
26
27 etiqueta_postorden = tk.Label(marco, text="Postorden: ", font=("Sans", 40))
28 etiqueta_postorden.pack(pady=5)
29
30 # Botón para salir de la aplicación
31 boton_salir = tk.Button(marco, text="Salir", command=raiz.quit, bg="salmon", font=("Sans", 40))
32 boton_salir.pack(pady=10)
33
34 # Inicio del bucle principal de la interfaz gráfica
35 raiz.mainloop()
```

Pruebas y resultados

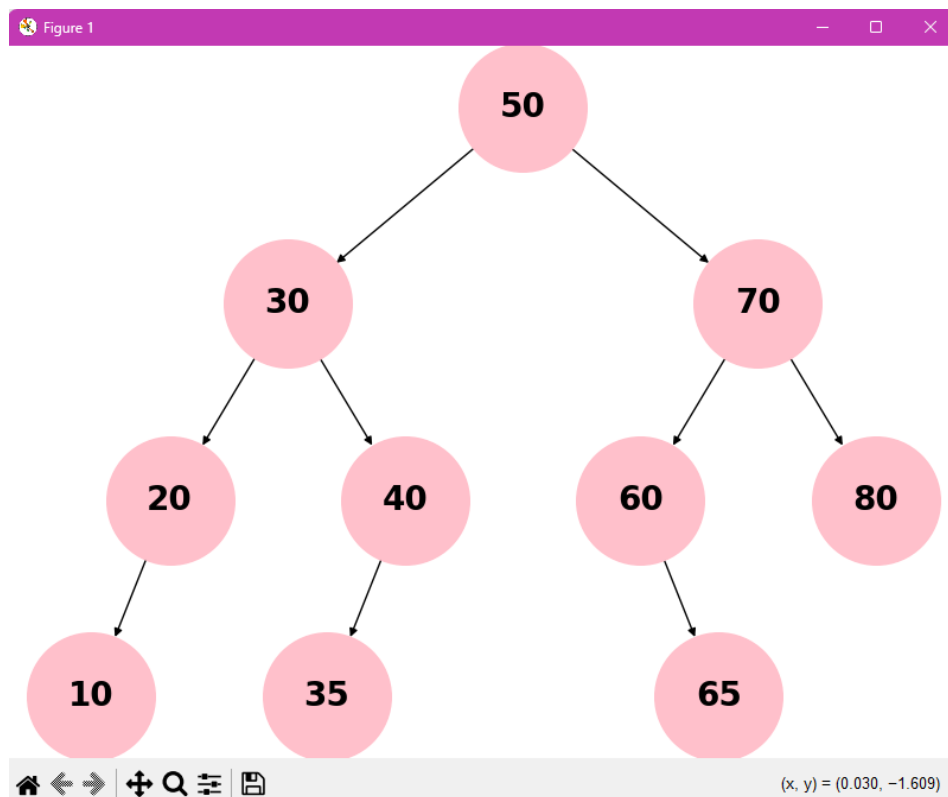
- Prueba 1:**

Para nuestra primera prueba ingresamos datos números dentro del CSV donde tomaremos como raíz al número 50.

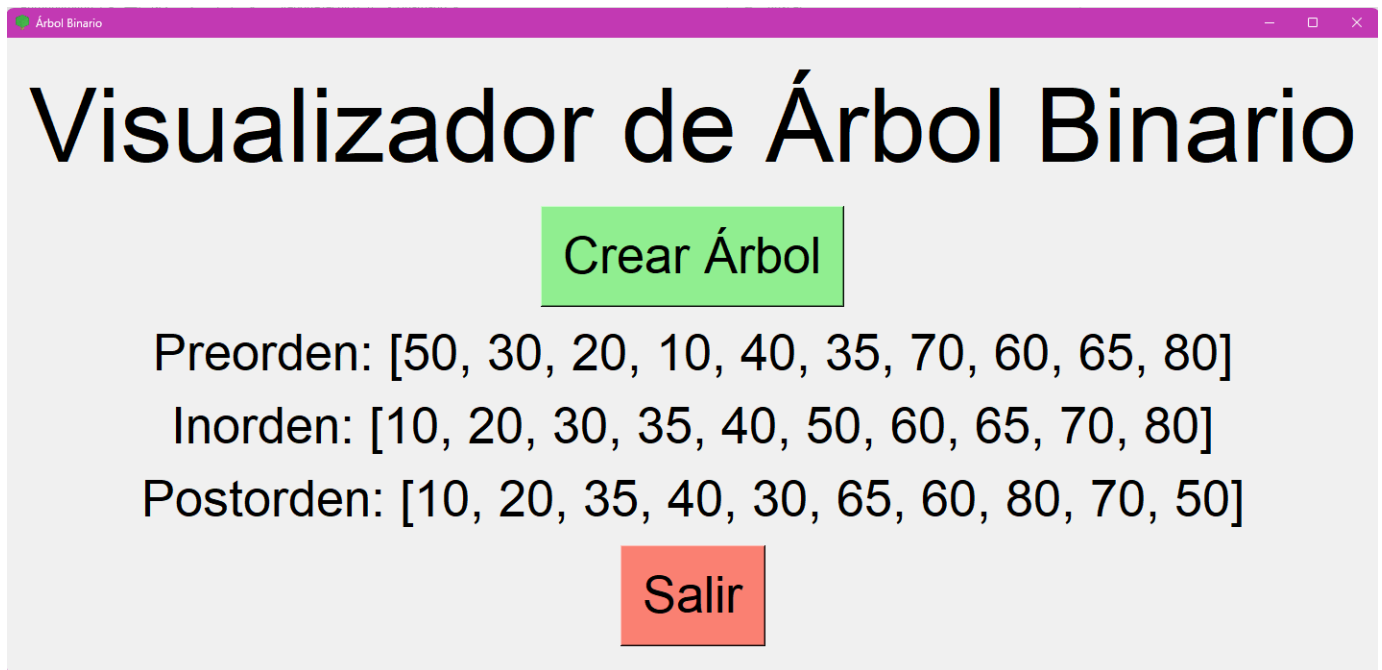


1	50
2	30
3	70
4	20
5	40
6	60
7	80
8	10
9	35
10	65

- Resultado 1:**



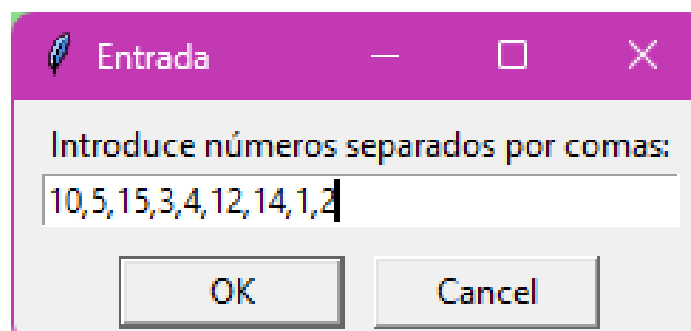
Nuestro primer resultado es una representación gráfica de nuestro árbol, podemos notar que el primer dígito introducido fue el 50 y como ya mencionamos, esto hace que sea la raíz del árbol, por ende, los siguientes dos datos son los subárboles y así sucesivamente.



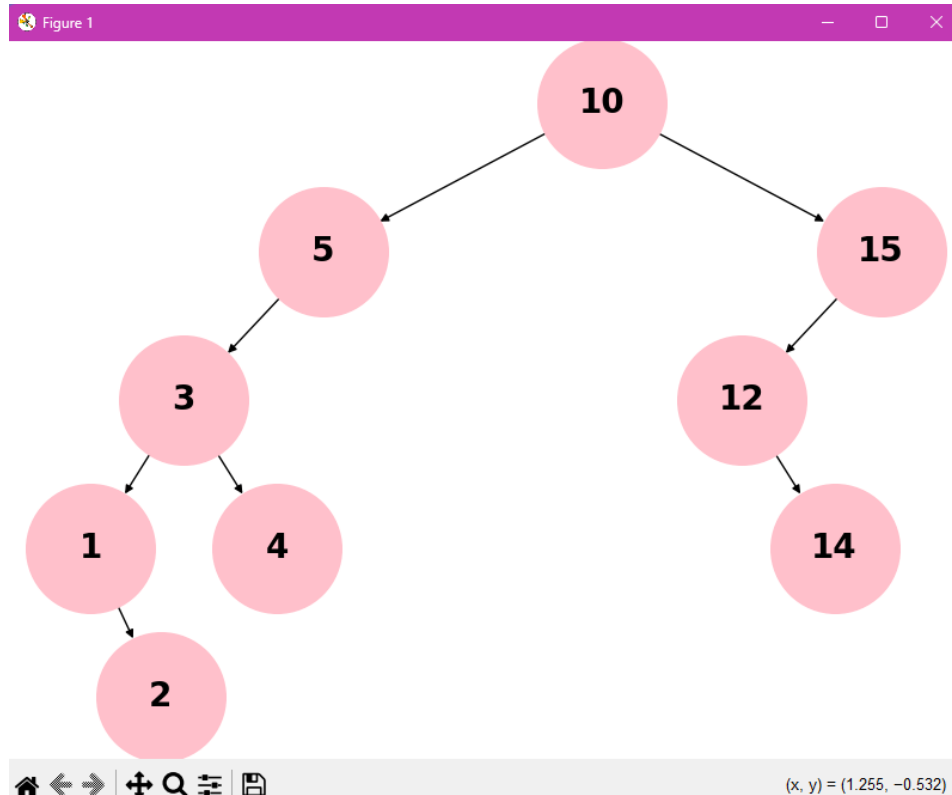
En este resultado nos arroja todos nuestros datos en los diferentes recorridos posibles para un árbol, utilizando nuestra teoría dicha previamente nos damos cuenta de que estos son correctos

- **Prueba 2:**

Para la segunda prueba ingresamos datos numéricos de manera manual donde tomaremos como raíz esta vez al número 10 como se muestra en la pantalla que son numero separados por comas.



- **Resultado 2:**



De igual manera en este resultado podemos notar distintas cosas, en esta gráfica la raíz es 400, después los dos números siguientes comienzan a ser los subárboles dependiendo de cuales son mayores y cuales son menores y cuales son menores, aquí podemos notar que acomoda los mayores a la derecha y los menores a la izquierda.

Árbol Binario

Visualizador de Árbol Binario

Crear Árbol

Preorden: [10, 5, 3, 1, 2, 4, 15, 12, 14]

Inorden: [1, 2, 3, 4, 5, 10, 12, 14, 15]

Postorden: [2, 1, 4, 3, 5, 14, 12, 15, 10]

Salir

Y de la misma forma, realiza los recorridos de nuestro árbol. Que al comprobarlos son correctos.

Conclusiones

Con este proyecto sentimos que realmente aprendimos muchas cosas y que, además de estudiar arduamente sobre el tema, también llegamos a aplicar este conocimiento en lo que se necesita para nuestra carrera, que es el código.

De forma gráfica, los árboles binarios nos ofrecen una buena forma de visualizar el ordenamiento de nodos, esto es sumamente útil al momento de realizar búsquedas y ordenar datos ya que esta estructura nos ayuda a ver de forma mas organizada y a su vez a relacionar estos entre sí.

Además de todo esto, el realizar todos los requerimientos para este proyecto nos ha hecho sentir aún más preparados para nuestro futuro en los siguientes semestres ya que esto es algo muy parecido a lo que nos pedirá en dicho futuro.

Nos sentimos orgullosos de haber cumplido con el objetivo de este proyecto y esperamos que pronto podamos aplicar más de nuestros conocimientos en más proyectos como este.

Bibliografía

Myriam Quiroa, 03 de julio, 2022
Árbol binario. Economipedia.com

Tutor de Estructuras de Datos Interactivo

Expósito López Daniel, Abraham García Soto, Martin Gómez Antonio José

Director de proyecto: Joaquín Fernández Valdivia

5º Licenciatura informática

ETSII 99/00 (Universidad de Granada).

Árboles binarios. (2014, 10 junio). Estructuras de Datos.

<https://hhmosquera.wordpress.com/arbolesbinarios/>