Trabajo Práctico – Estructuras de datos avanzadas: árboles.

Programación 1

- **Alumnos:** Fernando Gabriel Gómez - fernando.gomez@tupad.utn.edu.ar

  Rodrigo Agustín Galván - rodrigogalvann1@gmail.com
- **Profesor**: Ariel Enferrel

- **Fecha de Entrega**: 9/6/2025

Índice

---

1. Introducción

Para este trabajo práctico integrador, se eligió el tema **árboles en Python**, debido a la importancia de conocer estas estructuras de datos.

Su importancia radica en que son muy utilizados en aplicaciones que requieren una organización jerárquica de sus datos, como bases de datos (árboles de búsqueda binaria), sistemas operativos o algoritmos de optimización.

Es por esto, qué como técnicos en programación, profundizar en esta herramienta resulta indispensable.

A través de este trabajo, se tiene previsto explorar la teoría detrás de los **árboles** y aprender a implementarlos en **Python**.

2. Marco Teórico

¿Qué es un árbol?

Los **árboles** son estructuras de datos avanzadas las cuales están compuestas por nodos, dichos nodos pueden ser los siguientes:

- **Nodos:** se conocen como nodos a cada uno de los elementos dentro del árbol.
- **Nodo Raíz:** es el nodo principal del árbol y del cuál parten todos los demás nodos. Solo puede haber un solo nodo **raíz** en un árbol.
- **Nodo Padre:** un nodo padre es todo aquel nodo del cual parten otros nodos.
- **Nodo Hijo:** un nodo hijo es aquel que parte de un nodo padre.
- **Nodo Hermano:** se conocen como nodos hermanos a dos nodos que partan del mismo nodo padre.
- **Nodo Hoja:** son los nodos que no tienen ningún hijo.
- **Nodo Rama:** son todos los nodos intermedios entre el nodo **raíz** y los nodos **hijos**.

Cada nodo contiene dos componentes principales: un valor propio y un puntero que apunta a sus nodos hijos, si es que los tiene. Los nodos de un **árbol** están conectados entre sí mediante estas referencias, formando una estructura jerárquica.

Representación de árboles

Los árboles a menudo son representados como grafos que tienen nodos interconectados entre sí a través de ramas, tal como se muestra en la siguiente figura.

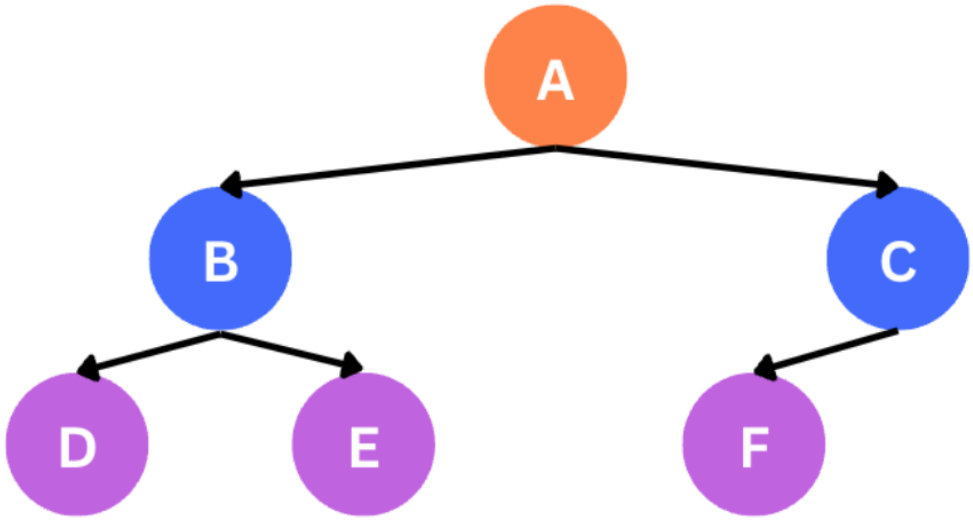

Figura 1. Representación de grafos de un árbol. Los círculos representan nodos y las flechas relaciones entre dichos nodos. **Fuente:** Apunte de cátedra "Árboles.pdf"

Propiedades de los árboles

Además de los **nodos**, estas estructuras cuentan con otras propiedades importantes las cuales son:

Longitud de camino y profundidad

Cuando se desea llegar de un nodo a otro dentro de un árbol se debe seguir un recorrido llamado **camino**. Formalmente un camino es una lista de ramas sucesivas que permiten llegar desde un nodo **n1** a otro nodo **n2** que van de **n1** a **n2**.

Estos recorridos se hacen a través de las ramas que conectan un **nodo** con otro, pero incluso si los **nodos** no están conectados directamente por una **rama**, se puede hacer el recorrido atravesando los **nodos** intermediarios.

La **longitud de camino** es la cantidad de **ramas** que hay que transitar para llegar de un **nodo** a otro. Por ejemplo, la longitud de camino entre los nodos **A** y **D** de la siguiente figura es de 2 ya que se deben transitar 2 ramas para llegar desde **A** hasta **D**.

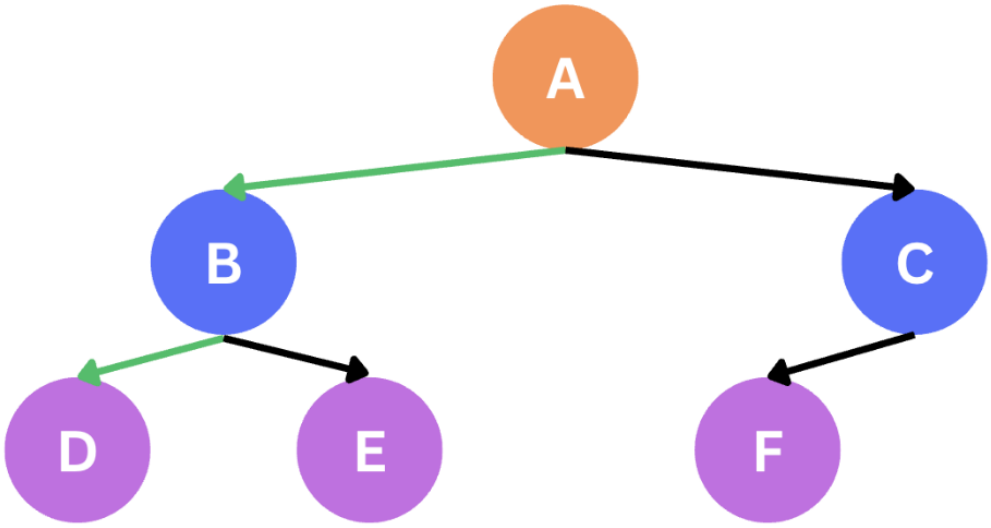

Figura 2. La longitud de camino entre los nodos A y D es igual a 2 ya que para llegar desde A hasta D hay que atravesar las dos ramas resaltadas en verde. **Fuente:** Apunte de cátedra "Árboles.pdf"

La **profundidad** de un **nodo** es la longitud de **camino** entre la **raíz** y él mismo.

Si imaginamos el **árbol** como un sistema de archivos en una computadora, podemos pensar en el **nodo raíz** como el directorio raíz del dispositivo de almacenamiento por ej. C: (profundidad = 0) y que el resto de **nodos** son las distintas carpetas que existan en su interior. Los **nodos hijos** de la raíz del árbol son las carpetas que se encuentran directamente en él y tendrían una profundidad = 1, las subcarpetas dentro de esas carpetas una profundidad = 2 y así sucesivamente.

Nivel y altura

El **nivel** de un **nodo** es la longitud de **camino** que lo conecta con la **raíz** más uno. Un nivel puede contener uno o más nodos.

Siguiendo con el ejemplo del sistema de archivos en una computadora, el **nodo raíz** sería el directorio principal (nivel 1), sus **hijos directos** serían las carpetas directamente dentro de él (nivel 2), y así sucesivamente. La diferencia entre el **nivel** y la **profundidad** es que el nivel incluye la **raíz** mientras que la **profundidad** no la tiene en cuenta.

La **altura** de un árbol es el **nivel** máximo del mismo.

Grado y orden

El **grado** de un **nodo** es la cantidad de hijos que tiene. El **grado** de un árbol es el grado máximo de los **nodos** del árbol.

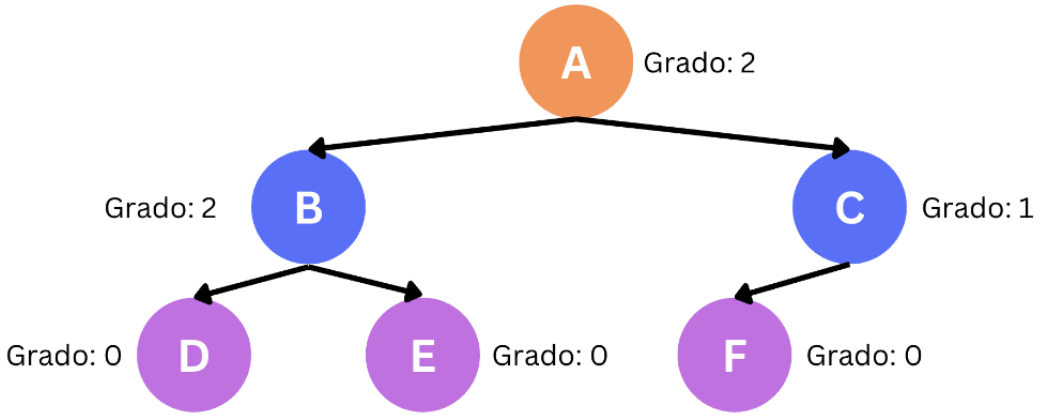

Figura 3. El grado de los nodos A y B es 2 ya que cada uno tiene dos nodos hijos. El grado del nodo C es 1 ya que tiene un solo hijo. El grado de los nodos D, E y F es 0 ya que no tienen hijos. El grado del árbol es 2 ya que el máximo grado de sus nodos es 2. **Fuente:** Apunte de cátedra "Árboles.pdf"

El **orden** de un árbol es una restricción que se establece antes de construirlo, la misma indica cual es la máxima cantidad de hijos que puede tener un **nodo** de ese árbol. Por ejemplo: si se quiere construir un árbol con un **orden** preestablecido de 3, esto significa que sus **nodos** no pueden tener más de 3 **hijos.**

Peso

El **peso** es la cantidad **total** de **nodos** que tiene un árbol. En programación es importante conocer este dato, ya que nos da una idea del tamaño del árbol y, por lo tanto, de cuanta memoria puede llegar a ocupar ejecutar un programa en él. Mientras más alto sea el **peso** de un árbol, mayor es la **memoria** que puede ocupar, por lo tanto, resulta crucial evaluar este aspecto en aplicaciones donde la memoria es limitada, como es el caso de dispositivos móviles o sistemas embebidos.

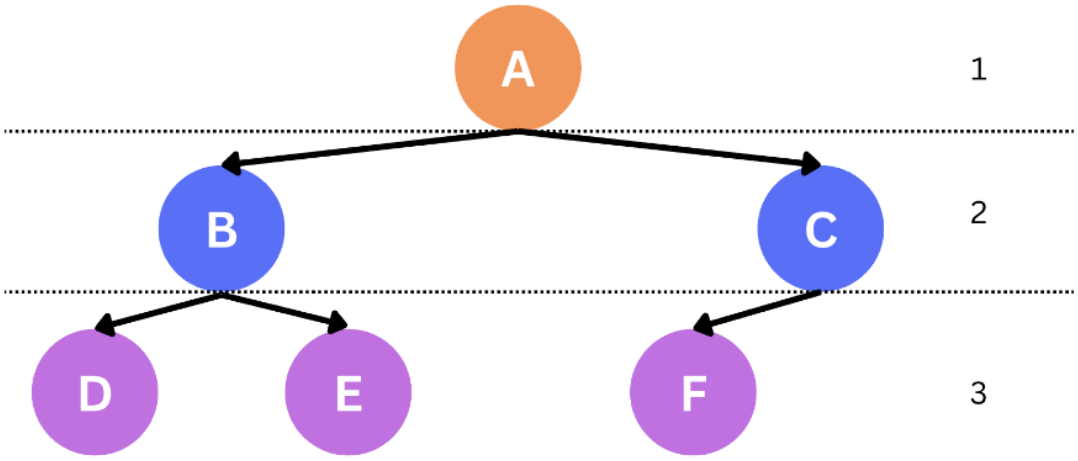

Figura 4. El peso de la línea superior es 1 ya que contiene un solo nodo, el de la línea del medio es 2 ya que contiene 2 nodos y el de la inferior es 3 ya que contiene 3 nodos. Por lo tanto el peso total del árbol es 1+2+3 = 6. **Fuente:** Apunte de cátedra "Árboles.pdf"

Tipos de árboles

Existen distintos tipos de árboles, en este trabajo se abordarán tres de ellos:

1. **Árbol binario:** es un árbol que como máximo cada **nodo** puede tener **2 hijos,** un **hijo** izquierdo y uno derecho.
2. **Árbol de búsqueda binaria (ABB):** este es un tipo de árbol binario que tiene una propiedad adicional. Los **nodos hijos** de la **izquierda** son **menores** que su **nodo padre** y los **nodos hijos** del lado **derecho** son **mayores** que su **nodo padre.**
3. **Árbol genérico o n-ario:** es un tipo de árbol en dónde cada **nodo** puede tener **n hijos**, donde **n** puede ser cualquier número entero positivo.

3. Caso Práctico

Cómo caso práctico, el grupo implementó un **árbol genérico** en la forma de un menú con submenús de un cajero automático.

A continuación, se adjunta el código y capturas de su funcionamiento.

Código:

```
import time, random

arbol = ["Menu Principal", ["1_ Consulta de saldo", ["1_ Consultar
saldo cuenta corriente"], ["2_ Consultar saldo cuenta ahorro"]], ["2_
Retiro de efectivo", ["1_ Retirar 20 mil pesos"], ["2_ Retirar un monto
especifico"]], ["3_ Depositos",["1_ Depositar en efectivo",], ["2_
Depositar en sobre"]], ["4_ Transferencias", ["1_ Transferir a un
alias"], ["2_ Transferir a un CBU/CVU"]], ["5_ Cambiar de clave",
["1_Cambiar clave del cajero"], ["2_Cambiar clave del home banking"]]]

activo = True

while activo:

    print("Bienvenido al banco")
    print(arbol[0])
    for submenu in arbol[1:]:
        print(submenu[0])

    eleccion = input("Por favor, seleccione una opcion: ")

    if(eleccion == "1"):
        print(f"{arbol[1][1][0]}\n{arbol[1][2][0]}")
        eleccion = input("Por favor, seleccione una opcion: ")
        if (eleccion == "1"):
            print(f"Su saldo es de {random.randint(10000,100000)}
pesos")
        else:
            print(f"Su saldo es de {random.randint(5000,10000)} pesos")
    elif(eleccion == "2"):
```

```
        print(f"{arbol[2][1][0]}\n{arbol[2][2][0]}")
        eleccion = input("Por favor, seleccione una opcion: ")
        if (eleccion == "1"):
            print("Usted retiro 20 mil pesos.")
        else:
            monto = input("Ingrese el monto a retirar: ")
            print(f"Usted retiro {monto} pesos de manera exitosa.")
    elif(eleccion == "3"):
        print(f"{arbol[3][1][0]}\n{arbol[3][2][0]}")
        eleccion = input("Por favor, seleccione una opcion: ")
        if (eleccion == "1"):
            print("Ingrese el dinero por la boquilla del cajero")
            time.sleep(3)
            print("Se ingreso el dinero de manera exitosa")
        else:
            print("Ingrese el sobre por la boquilla del cajero")
            time.sleep(3)
            print("Se ingreso el dinero de manera exitosa")
    elif(eleccion == "4"):
        print(f"{arbol[4][1][0]}\n{arbol[4][2][0]}")
        eleccion = input("Por favor, seleccione una opcion: ")
        if (eleccion == "1"):
            alias = input("Ingrese el alias: ")
            monto = input("Ahora ingrese el dinero a transferir: ")
            print(f"Se ha realizado la transferncia con exito al alias:
{alias} con un monto total de {monto} pesos")
        else:
            cbu = input("Ingrese el alias: ")
            monto = input("Ahora ingrese el dinero a transferir: ")
            print(f"Se ha realizado la transferncia con exito al cbu:
{cbu} con un monto total de {monto} pesos")
    elif(eleccion == "5"):
        print(f"{arbol[5][1][0]}\n{arbol[5][2][0]}")
        eleccion = input("Por favor, seleccione una opcion: ")
        if (eleccion == "1"):
            input("Ingrese la nueva contraseña para usar el cajero: ")
            print("La contraseña se cambio con exito")
```

```
        else:
            input("Ingrese la nueva contraseña para usar su home
banking: ")
            print("La contraseña se cambio con exito")
    else:
        print("Opcion invalida")

    continuar = input("Desea realizar otra operacion? (s/n)").lower()
    if continuar != "s":
        print("Gracias por usar el cajero, vuelva pronto.")
        activo = False
```

Código ejecutado:

```
Bienvenido al banco
Menu Principal
1_ Consulta de saldo
2_ Retiro de efectivo
3_ Depositos
4_ Transferencias
5_ Cambiar de clave
Por favor, seleccione una opcion: 1
1_ Consultar saldo cuenta corriente
2_ Consultar saldo cuenta ahorro
Por favor, seleccione una opcion: 1
Su saldo es de 76138 pesos
Desea realizar otra operacion? (s/n)n
Gracias por usar el cajero, vuelva pronto.
```

4. Metodología Utilizada

Para el desarrollo de este informe se hizo un trabajo de investigación previa. Las fuentes utilizadas fueron las siguientes:

- Árboles (s.f). Apunte de cátedra. Universidad Tecnológica Nacional
- Oscar Blancarte. (2014, agosto 22). **Estructura de datos – Árboles**. Oscar Blancarte software architect. https://www.oscarblancarteblog.com/2014/08/22/estructura-de-datos-arboles/

Código:

- Se utilizó **Python 3.11.9**
- Cada **nodo** del árbol fue representado por una opción del menú.
- Se desarrollaron funciones que simulan el comportamiento de un cajero automático.

---

5. Resultados Obtenidos

En este trabajo se implementó una estructura de árbol genérico para representar el menú de un cajero automático.

El **nodo raíz** fue representado por un **Menú Principal**, sus **nodos hijos** directos, que son 5, por las opciones directamente en él, y a su vez esos **nodos** tienen **2 nodos hijos** cada uno.

De esta manera, se pudo observar, como la estructura permite al usuario navegar por las opciones del sistema. A medida que se seleccionan las diferentes opciones, el sistema profundiza en los **niveles del árbol** mostrando otras subopciones(**nodos**), permitiendo una navegación intuitiva entre las distintas opciones que ofrece el cajero automático.

---

6. Conclusiones

Durante el desarrollo de este trabajo, el grupo comprendió la importancia que tienen los **árboles como estructuras de datos** para ser implementados en distintos algoritmos o estructuras que necesiten un orden jerárquico de sus elementos.

En un futuro, y contando con más conocimientos, en lugar de usar **listas anidadas** se podrían implementar con estructuras más complejas como **objetos, diccionarios** o incluso **árboles**.

---

7. Bibliografía

- Árboles (s.f). Apunte de cátedra. Universidad Tecnológica Nacional
- Oscar Blancarte. (2014, agosto 22). **Estructura de datos – Árboles**. Oscar Blancarte software architect. https://www.oscarblancarteblog.com/2014/08/22/estructura-de-datos-arboles/

8.Anexos

- **Enlace a la presentación: https://www.youtube.com/watch?v=sTit_EGrwVI**
- **Enlace al repositorio de Github:** https://github.com/rodrigoGalvanDev/menu-interactivo