Trabajo Práctico Integrador Programación I

Datos Avanzados

Alumnos

Martinez Juan Pablo (**juan.martinez@tupad.utn.edu.ar**),

Balverdi Nicolás Leonel (**nicolas.balverdi@tupad.utn.edu.ar**)

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

Programación I

Docente Titular

Sebastián Bruselario

Docente Tutor

Verónica Carbonari

09 de junio de 2025

Índice

Introducción

El presente trabajo integrador pretende abordar el tema de investigación sobre “datos avanzados” del temario de la materia Programación I. Por definición “Un árbol es una estructura de datos no lineal en la que cada nodo puede apuntar a uno o varios nodos. Son muy similares a las listas doblemente enlazadas en el sentido que tienen punteros que apuntan a otros elementos. Sin embargo, a diferencia de las listas doblemente enlazadas, no tienen una estructura lógica de tipo lineal o secuencial, sino ramificada. Tienen aspecto de árbol, de ahí su nombre”(Apuntes de la cátedra de Programación I, 2025).

La elección de este tema es de importancia, ya que son estructuras flexibles para almacenar datos jerárquicamente y permite la búsqueda de datos de manera más fácil. Como futuros programadores es importante implementar estas estructuras ya que se encuentran en muchas aplicaciones tales como los sistemas de archivos, bases de datos, e inclusive en sistemas operativos.

Por último, el presente trabajo tiene como objetivo, implementar un caso práctico a modo de demostración sobre árboles de búsqueda binaria. Crear un código funcional sobre el caso práctico utilizando el lenguaje Python. Demostrar todos los conocimientos adquiridos y aprendidos durante la cursada de Programación I.

Marco Teórico

¿Qué es un árbol binario?: Un árbol binario es una estructura de datos no lineal y jerárquica donde cada nodo tiene como máximo dos hijos, denominados hijo izquierdo y hijo derecho. El nodo superior de un árbol binario se denomina raíz, los nodos en los extremos opuestos a la raíz se denominan hojas y los nodos entre ellos se denominan ramas del árbol. Se utiliza principalmente para optimizar procesos. Para este trabajo, en Python el concepto de árbol se implementará con las estructuras de listas anidadas.

¿Qué es un árbol de búsqueda binaria? Un árbol de búsqueda binaria (ABB) es un árbol binario con un conjunto de reglas para los datos que almacena. Por ejemplo, a partir de un nodo raíz o un nodo intermedio, sus hijos deben estar ubicados según su valor con respecto al del nodo padre. Si es menor, debe estar ubicado a la izquierda mientras que en el caso de ser mayor se debe ubicar a la derecha. El propósito de estos árboles es generar una estructura jerárquica y optimizar los tiempos de búsqueda.

¿Qué son los nodos? Un nodo es una parte fundamental de un árbol, son los puntos en donde que conecta cada nodo. Puede tener un nombre, y su principal función es la de contener información.

¿Qué es una raíz? Es el nodo inicial o principal con el que va a comenzar nuestro árbol ya que es el que tiene mayor jerarquía.

Hojas: Son los nodos finales de los árboles, es decir, que no tienen nodos hijos.

Ramas: Son los nodos que tienen al menos un nodo hijo y un nodo padre.

Listas: Las listas son estructuras de datos que permiten almacenar información de una forma ordenada, y del cual es posible agregar valores, editarlos o eliminar datos dentro de la propia lista. Los árboles en python se abordaron utilizando las listas.

Caso Práctico

Almacenaje de códigos de productos para un gigante comercial

Para nuestro caso en concreto, vamos a implementar un árbol de búsqueda binaria en donde un gigante comercial almacena los códigos de sus productos. El objetivo de nuestro programa es reducir los tiempos de búsqueda y facilitar la localización de productos en el sistema.

Desarrollo del código

```
# Funciones

def insertar(arbol, valor):
    if arbol == []:
        return [valor, [], []]
    if valor < arbol[0]:
        arbol[1] = insertar(arbol[1], valor)
    elif valor > arbol[0]:
        arbol[2] = insertar(arbol[2], valor)
    return arbol

def crear_arbol():
    arbol = []
    continuar = 1
    while continuar == 1:
        valor = input("Ingrese un código: ")
        if valor.isdigit():
            codigo = int(valor)
            arbol = insertar(arbol, codigo)
        else:
            print("No se permiten códigos negativos o letras, el sistema trabaja con inputs
numéricos")
        continuar = int(input("Desea ingresar otro código? (1. Si / 2. No): "))
    return arbol

def mostrar_arbol(arbol, nivel=0):
    if arbol != []:
        mostrar_arbol(arbol[2], nivel + 1)
        print("    " * nivel + f"→ {arbol[0]}")
        mostrar_arbol(arbol[1], nivel + 1)

def encontrar_ruta(arbol, valor, ruta=[]):
    if arbol == []:
        return []

    ruta = ruta + [arbol[0]]

    if valor == arbol[0]:
        return ruta
    elif valor < arbol[0]:
        return encontrar_ruta(arbol[1], valor, ruta)
    else:
        return encontrar_ruta(arbol[2], valor, ruta)

def mostar_ruta(ruta = list):
```

```
    if ruta == []:
        return []
    else:
        for i in range(0, len(ruta)):
            if i == 0:
                print(ruta[i], end="")
            else:
                print(f" → {ruta[i]}", end="")


# Modificar un nodo

def minimo(arbol):
    actual = arbol
    while actual[1] != []:
        actual = actual[1]
    return actual[0]

def maximo(arbol):
    actual = arbol
    while actual[2] != []:
        actual = actual[2]
    return actual[0]

def modificar_nodo(arbol, valor_antiguo, nuevo_valor, padre=[], es_izquierdo=bool):
    if arbol == []:
        return []

    if valor_antiguo < arbol[0]:
        modificar_nodo(arbol[1], valor_antiguo, nuevo_valor, arbol, True)
    elif valor_antiguo > arbol[0]:
        modificar_nodo(arbol[2], valor_antiguo, nuevo_valor, arbol, False)
    else:
        # Validación con padre
        if padre != []:
            if es_izquierdo and nuevo_valor >= padre[0]:
                print(f"Un código a la izqierda de {padre[0]} no puede ser mayor. Si necesita
cargarlo, utilice insertar.")
                return
            elif not es_izquierdo and nuevo_valor <= padre[0]:
                print(f"Un código a la derecha de {padre[0]} no puede ser menor. Si necesita
cargarlo, utilice insertar.")
                return
        # Validación con hijos
        if arbol[1] != [] and nuevo_valor <= maximo(arbol[1]):
            print(f"{nuevo_valor} no puede ser menor o igual a {maximo(arbol[1])}, utilice la
operación insertar si desea cargar {nuevo_valor} en el sistema")
            return
```

```
        if arbol[2] != [] and nuevo_valor >= minimo(arbol[2]):
            print(f"{nuevo_valor} no puede ser mayor o igual a {minimo(arbol[2])}, utilice la
operación insertar si desea cargar {nuevo_valor} en el sistema.")
            return
        arbol[0] = nuevo_valor
        print(f"Se ha modificado correctamente {valor_antiguo} por {nuevo_valor}")
    return arbol
# Los 3 recorridos de los arboles binarios
def recorrer_preorden(arbol):
    if arbol == []:
        return []
    return [arbol[0]] + recorrer_preorden(arbol[1]) + recorrer_preorden(arbol[2])

def recorrer_inorden(arbol):
    if arbol == []:
        return []
    return recorrer_inorden(arbol[1]) + [arbol[0]] + recorrer_inorden(arbol[2])

def recorrer_postorden(arbol):
    if arbol == []:
        return []
    return recorrer_postorden(arbol[1]) + recorrer_postorden(arbol[2]) + [arbol[0]]


# Programa principal

print("== Trabajo Práctico Integrador ==")
print("A continuación debe cargar los códigos de sus productos al sistema")
arbol = crear_arbol()
print("Sus códigos han sido cargados correctamente en un árbol de búsqueda binaria")
mostrar_arbol(arbol)

print("Desea realizar otra operación?")
print("1. Buscar un código")
print("2. Insertar un código")
print("3. Modificar un código")
print("4. Recorrer el sistema")
print("5. Salir")

valor = input("Ingrese el número de la operación: ")
while not valor.isdigit():
    print("El sistema sólo permite inputs numéricos positivos.")
    valor = input("Ingrese el número de la operación: ")

operacion = int(valor)

while operacion != 5 and (operacion > 0 and operacion < 6):
    if operacion == 1:
```

```
    valor = input("Indique el código que esta buscando: ")
    if valor.isdigit():
        codigo = int(valor)
        ruta = encontrar_ruta(arbol, codigo)
        mostar_ruta(ruta)
        print()
    else:
        print("El sistema sólo trabaja con inputs numéricos positivos")
elif operacion == 2:
    valor = input("Indique el código: ")
    if valor.isdigit():
        codigo = int(valor)
        if codigo > 0:
            insertar(arbol, codigo)
            mostrar_arbol(arbol)
    else:
        print("El sistema sólo trabaja con inputs numéricos positivos")
elif operacion == 3:
    valor = input("Ingrese el código que desea modificar: ")
    valor2 = input("Ingrese el nuevo código: ")
    if valor.isdigit() and valor2.isdigit():
        codigo = int(valor)
        nuevoCodigo = int(valor2)
        modificar_nodo(arbol, codigo, nuevoCodigo)
        mostrar_arbol(arbol)
    else:
        print("El sistema sólo trabaja con inputs numéricos positivos")
elif operacion == 4:
    print("En qué orden desea visualizar el sistema?")
    print("1. Preorden / 2. Inorden / 3. Postorden")
    valor = input("Ingrese el número de la opción: ")
    if valor.isdigit():
        opcion = int(valor)
        if opcion == 1:
            print(f"Recorrido en preorden: {recorrer_preorden(arbol)}")
        elif opcion == 2:
            print(f"Recorrido Inorden: {recorrer_inorden(arbol)}")
        elif opcion == 3:
            print(f"Recorrido Postorden: {recorrer_postorden(arbol)}")
    else:
        print("Opción incorrecta. El sistema sólo trabaja con inputs numéricos positivos")
else:
    print("Ha cerrado el sistema")

print("Desea realizar otra operación?")
print("1. Buscar un código")
print("2. Insertar un código")
print("3. Modificar un código")
```

```
    print("4. Recorrer el sistema")
    print("5. Salir")

    valor = input("Ingrese el número de la operación: ")
    while not valor.isdigit():
        print("El sistema sólo permite inputs numéricos positivos.")
        valor = input("Ingrese el número de la operación: ")

    operacion = int(valor)
```

Resultado esperado (salida por consola)

```
== Trabajo Práctico Integrador ==
A continuación debe cargar los códigos de sus productos al sistema
Ingrese un código: 50
Desea ingresar otro código? (1. Si / 2. No): 1
Ingrese un código: 25
Desea ingresar otro código? (1. Si / 2. No): 1
Ingrese un código: 20
Desea ingresar otro código? (1. Si / 2. No): 1
Ingrese un código: 30
Desea ingresar otro código? (1. Si / 2. No): 1
Ingrese un código: 70
Desea ingresar otro código? (1. Si / 2. No): 1
Ingrese un código: 65
Desea ingresar otro código? (1. Si / 2. No): 1
Ingrese un código: 80
Desea ingresar otro código? (1. Si / 2. No): 2
Sus códigos han sido cargados correctamente en un árbol de búsqueda binaria
        → 80
    → 70
        → 65
→ 50
        → 30
    → 25
        → 20
Desea realizar otra operación?
1. Buscar un código
2. Insertar un código
3. Modificar un código
4. Recorrer el sistema
5. Salir

Ingrese el número de la operación: 1
Indique el código que esta buscando: 20
50 → 25 → 20
Desea realizar otra operación?
1. Buscar un código
```

2. Insertar un código
3. Modificar un código
4. Recorrer el sistema
5. Salir
Ingrese el número de la operación: 2
Indique el código: 120
→ 120
→ 80
→ 70
→ 65
→ 50
→ 30
→ 25
→ 20
Desea realizar otra operación?
1. Buscar un código
2. Insertar un código
3. Modificar un código
4. Recorrer el sistema
5. Salir
Ingrese el número de la operación: 3
Ingrese el código que desea modificar: 120
Ingrese el nuevo código: 90
Se ha modificado correctamente 120 por 90
→ 90
→ 80
→ 70
→ 65
→ 50
→ 30
→ 25
→ 20
Desea realizar otra operación?
1. Buscar un código
2. Insertar un código
3. Modificar un código
4. Recorrer el sistema
5. Salir
Ingrese el número de la operación: 4
En qué orden desea visualizar el sistema?
1. Preorden / 2. Inorden / 3. Postorden
Ingrese el número de la opción: 1
Recorrido en preorden: [50, 25, 20, 30, 70, 65, 80, 90]
Desea realizar otra operación?
1. Buscar un código
2. Insertar un código
3. Modificar un código
4. Recorrer el sistema

5. Salir
Ingrese el número de la operación:

Metodología

Desde la metodología, utilizamos diferentes fuentes, herramientas y prácticas para llevar a cabo el proyecto, entre ellos están:

Busqueda de informacion

- Se realizó la búsqueda de fuentes de material bibliográfico como documentos, libros y videos dispuesto por la cátedra de Programación I en relación al tema. Se utilizaron páginas web de python sobre árboles y sus definiciones, terminologías y sintaxis.

Herramientas utilizadas:

- Visual studio code: Es una herramienta para trabajo colaborativo, en donde se creó, desarrolló y editó el código fuente del trabajo integrador.
- Python: Se utilizó el lenguaje de programación Python en su versión 3.11.9, así como se estipula su utilización por la cátedra.
- Creación de repositorio en GitHub para almacenar el código y compartirlo.
- Se hizo uso de archivo .doc para documentar, escribir y desarrollar el trabajo integrador.
- Uso de aplicaciones para mantenernos comunicados, por ejemplo, whatsapp, y discord

Trabajo colaborativo:

- Ambos realizamos la tarea en conjunta de la búsqueda de información que pudiera resultar útil e interesante para la creación del proyecto integrador.
- Desarrollo del trabajo escrito: Se desarrolló el trabajo escrito de manera conjunta y a la par de la creación del código fuente.
- El diseño del código se realizó dividiendo las tareas, pero de manera sincrónica, por un lado se trabajó las líneas de las funciones y por el otro, el código principal.
- Constante comunicación por diferentes vías.

Resultados Obtenidos

Como puede ver al momento de ejecutar el código (ver Anexo 1) y en el video explicativo (ver Anexo 2), la terminal muestra el mensaje de ingresar un dato numérico, este será el nodo raíz del árbol binario (ver Anexo 3), una vez que se ingresa el nodo raíz, se le da la opción al usuario, ingresando un 1 si decide continuar y un 2 para no continuar (ver Anexo 4). Una vez que se ingrese otro nodo, el árbol va a ir alineando los nodos entrantes al comparar con el nodo padre si es mayor o menor que él y con los nodos ramas.
Una vez que el usuario decida finalizar, la terminal mostrará el árbol armado con todos los valores ingresados (ver Anexo 5). A su vez, se le dará la opción al usuario si decide buscar, insertar o modificar un código. Podrá recorrer el sistema dependiendo si lo quiere recorrer en preorden, inorden o postorden (ver Anexo 6), y para finalizar, si quiere terminar con la ejecución, puede indicar salir.

Como podemos ver en el desarrollo del caso práctico, hemos podido realizar el código funcional de un árbol de búsqueda binaria para datos de manera exitosa. Si bien, la ejecución del código dio el resultado esperado, el trabajo no estuvo libre de complicaciones y lo que a primera vista parecía ser algo simple de realizar probó ser todo un reto que puso a prueba las habilidades adquiridas hasta el momento. Por ejemplo, inicialmente se pensaba imprimir en pantalla el árbol de forma vertical dado que sería la mejor forma para que el usuario comprendiera la estructura del árbol y sus reglas pero se hizo demasiado complejo su implementación. Decantamos por el método actual que si bien no es el óptimo cumple con la función de graficar el árbol.

También se intentó implementar la eliminación de un nodo pero, al eliminar un nodo padre que tenía nodos hijos el reemplazo del padre se realizaba de forma incorrecta. Creemos que con un poco más de tiempo hubiéramos logrado corregirlo pero de momento quedó fuera del programa.

Conclusiones

A modo de conclusión, se pudo llevar a cabo la realización y la implementación de un código ejecutable y funcional de un árbol de búsqueda binaria para el trabajo integrador de Programación I, teniendo en cuenta muchos de los conocimientos que fuimos adquiriendo durante la cursada y algunos nuevos como a implementación del árbol con listas anidadas, pudiendo verse reflejado en nuestro desempeño, hemos podido lograr la comprensión de conocimientos, comandos y estrategias que nos ayudaran en nuestros próximos proyectos.

En cuanto a la importancia del tema elegido a la programación, como ya mencionamos anteriormente, los árboles son de utilidad debido a su eficiencia, la flexibilidad y su capacidad para poder representar gráficamente datos almacenados, y es fundamental para el desarrollo de códigos con el que podamos encontrarnos en algun futuro ya como programadores más experimentados o en casos más avanzados.

Bibliografía

Árboles. (2025) Apuntes de la cátedra de Programación I. Universidad Tecnológica Nacional. https://docs.google.com/document/d/10k16oL15EeyOaq92aoi4qwK3t_22X29-FSV2iV-8N1U/edit?tab=t.0

Miller, B., & Ranum, D. (2011, agosto 22) *Solución de problemas con algoritmos y estructuras de datos usando Python*. Editorial Franklin, Beedle & Associates. https://runestone.academy/ns/books/published/pythoned/index.html

Miller, B., & Ranum, D. (2011, agosto 22). Comencemos con los datos. En *Solución de problemas con algoritmos y estructuras de datos usando Python*. Editorial Franklin, Beedle & Associates. https://runestone.academy/ns/books/published/pythoned/Introduction/ComencemosConLosDatos.html

Miller, B., & Ranum, D. (2011, agosto 22) Árboles y algoritmos de árboles. En *Solución de problemas con algoritmos y estructuras de datos usando Python*. Editorial Franklin, Beedle & Associates. https://runestone.academy/ns/books/published/pythoned/Trees/toctree.html

Miller, B., & Ranum, D. (2011, agosto 22). Listas. En *Solución de problemas con algoritmos y estructuras de datos usando Python*. Editorial Franklin, Beedle & Associates. https://runestone.academy/ns/books/published/pythoned/index.html

Downey, A. (2012, septiembre 12). Think Python. Editorial O'Reilly Media. https://greenteapress.com/wp/think-python/

Tecnicatura (2025, febrero 5) Introducción a árboles [Video]. Youtube. https://www.youtube.com/watch?v=cVm51pO35qA&list=PLy5wpwhsM-2lIY-qe_fALJ4K_XAhLZ2l-&index=4

Tecnicatura (2025, febrero 5) Propiedades de árboles [Video]. Youtube. https://www.youtube.com/watch?v=kqmn7AYYdSA

Tecnicatura (2025, febrero 5) Formas de recorrer un árbol [Vídeo]. Youtube. https://www.youtube.com/watch?v=I4IKGp0PsO0&list=PLy5wpwhsM-2lIY-qe_fALJ4K_XAhLZ2l-

Tecnicatura (2025, febrero 5) Árboles de búsqueda binaria [Vídeo]. Youtube. https://www.youtube.com/watch?v=O5gvB-LWyhY&list=PLy5wpwhsM-2lIY-qe_fALJ4K_XAhLZ2l-&index=2

Wikipedia contributors. (2025, junio 8). *Árbol (informática)*. Wikipedia. https://es.wikipedia.org/wiki/%C3%81rbol_(inform%C3%A1tica)

Anexos

Anexo 1

Enlace del repositorio de GitHub:

https://github.com/NicoBalverdi/UTN_TUPaD_TP_Integrador_Programacion_I

Anexo 2

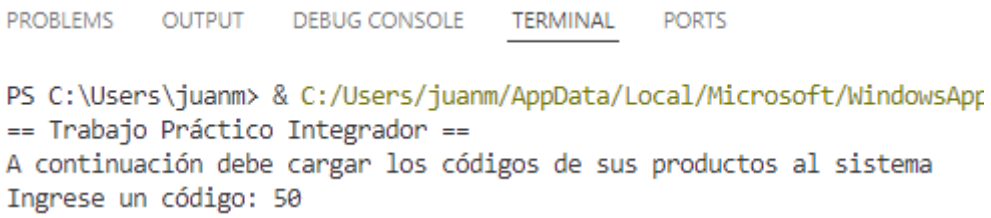
Enlaces del video explicativo:

https://www.youtube.com/watch?v=uBsS1K37f9Y

https://drive.google.com/drive/folders/1uo4rMla_sCI84k14EYFSVZ-nlQUUTXAZ

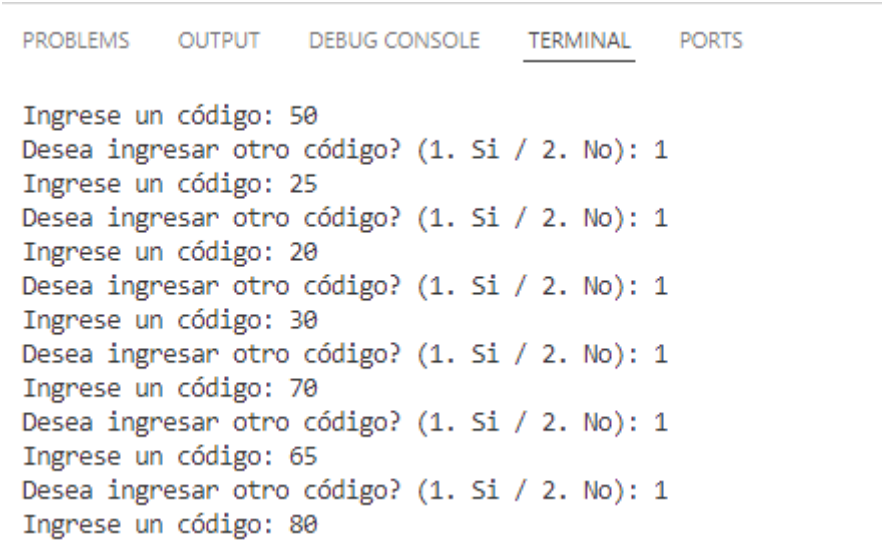
Anexo 3

Captura de pantalla: Ingreso del nodo raíz



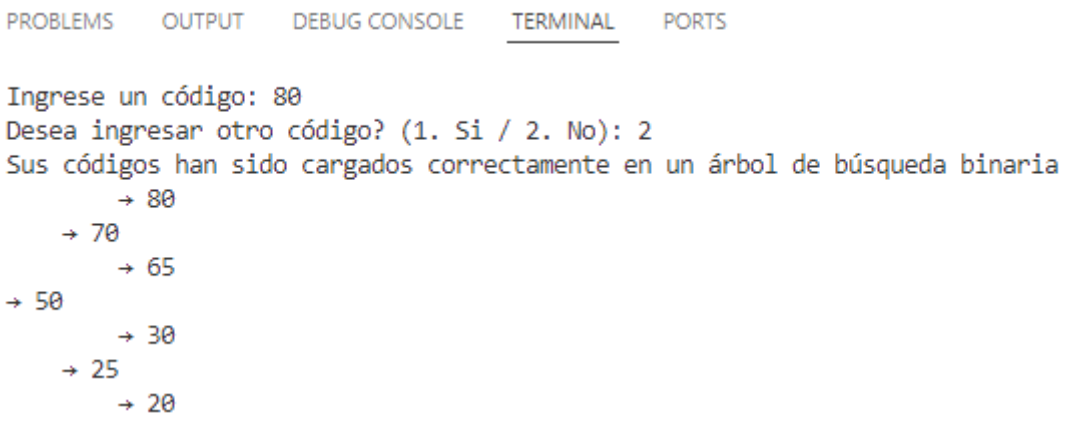
Anexo 4

Captura de pantalla: ingreso de los nodos ramas e hojas



Anexo 5

Captura de pantalla: Salida de consola del árbol armado



Anexo 6

Captura de pantalla: Se le solicita al usuario elegir mas opciones de operaciones