

TRABAJO PRÁCTICO INTEGRADOR

TÍTULO DEL TRABAJO: Árboles con Python

ALUMNOS:

Azcuy Nicolás - nico.azcuy@gmail.com

Anchorena Tomás - tomas_2806@hotmail.com

MATERIA: Programación 1

PROFESOR: AUS Bruselario, Sebastián

TUTORA: Gubiotti, Flor

FECHA DE ENTREGA: 09 de Junio de 2025

ÍNDICE

1. Introducción	Pág: 3
2. Marco Teórico	Pág:
3. Caso Práctico	Pág:
4. Metodología Utilizada	Pág:
5. Resultados Obtenidos	Pág:
6. Conclusiones	Pág:
7. Bibliografía	Pág:
8. Anexos	Pág:

1. Introducción

El presente trabajo, desarrollado en el marco de la Tecnicatura Universitaria en Programación de la UTN para la materia Programación 1, aborda el estudio e implementación de árboles binarios utilizando listas como estructura. La elección de este tema responde a la necesidad de comprender en profundidad el funcionamiento de los árboles binarios, una estructura presente en los sistemas de archivos o en las bases de datos, que como futuros técnicos en programación deberemos implementar, optimizar y adaptar según diferentes situaciones o problemas a resolver.

Los árboles binarios permiten representar relaciones jerárquicas claras, haciéndolos especialmente útiles en tareas como la búsqueda y organización de datos, la gestión de archivos y el desarrollo de algoritmos de decisión. A diferencia de estructuras lineales como las listas o arreglos, los árboles ofrecen una organización no secuencial, no lineal, que facilita el acceso y procesamiento de la información.

Como estudiantes de la Tecnicatura, el conocimiento de estructuras como los árboles binarios es esencial, ya que se aplican en áreas clave del desarrollo de software, como la programación de bases de datos, el diseño de algoritmos, el desarrollo de aplicaciones complejas y la comprensión del funcionamiento interno de muchos lenguajes y herramientas de programación.

El objetivo de este trabajo es comprender y aplicar el funcionamiento de los árboles binarios a través de su implementación con listas, analizando sus principales operaciones (como inserción, modificación y búsqueda) y su aplicación en contextos reales. A través de este desarrollo, se busca reforzar los conocimientos adquiridos en estructuras de datos, contribuyendo a la formación integral en el área de la programación y el pensamiento algorítmico.


2. Marco Teórico

🔗 Definición de Árbol Binario

Un **árbol binario** es una estructura de datos jerárquica compuesta por nodos, donde cada nodo puede tener como máximo dos hijos: uno izquierdo y uno derecho. Esta estructura permite representar relaciones padre-hijo y facilita operaciones como búsqueda, inserción y recorrido de datos.


🔗 Representación con Listas en Python

Una forma didáctica de implementar árboles en Python es usando listas. Cada nodo puede representarse como una lista de tres elementos:



```
[nodo, subarbol_izquierdo, subarbol_derecho]
```

Por ejemplo:



```
[10, [5, None, None], [15, None, None]]
```

Representa un nodo con:

- Nodo raíz = 10
- Hijo izquierdo = 5
- Hijo derecho = 15

🔍 Operaciones comunes sobre árboles

Operación	Descripción
Insertar	Agregar un nuevo valor manteniendo el orden del ABB
Buscar	Recorre el árbol comparando valores hasta encontrar el nodo deseado
Eliminar	Quita un nodo ajustando los enlaces entre los nodos restantes
Recorridos	Visita los nodos en un orden específico: Preorden, Inorden, Postorden

🔍 Tipos de Recorridos

Tipos de Recorrido	Orden de Visita	Uso Común
Preorden	Nodo → Izquierdo → Derecho	Serialización del Árbol
Inorden	Izquierdo → Nodo → Derecho	Obtener elementos en orden ascendente en un ABB
Postorden	Izquierdo → Derecho → Nodo	Eliminar o liberar memoria del árbol

📌 Ventajas de usar árboles binarios

- Eficiencia en búsqueda e inserciones
 - Estructura ordenada sin necesidad de reordenar manualmente
 - Buena base para estructuras más complejas
-

📌 Implementación en Python

En Python, un árbol binario con listas permite practicar operaciones básicas sin necesidad de clases. Es una aproximación útil en trabajos introductorios o educativos.

3. Caso Práctico

Una biblioteca barrial decide dejar de llevar un inventario a mano e implementa un sistema de préstamo de libros basado en códigos numéricos que identifican de forma única cada ejemplar. Cada título de libro corresponde a un valor entero y se busca que:

- Registre el ingreso de un nuevo ejemplar mediante la carga de su código único.
- De de baja o elimine del registro un libro con determinado código que fue extraviado o destruido.
- Actualice o modifique el código de un ejemplar cuando se precise reenumerar el libro.
- Generar listados de códigos en diferentes órdenes para que el bibliotecario pueda tener un balance interno.

Para cubrir estas necesidades, se diseñó un árbol binario de búsqueda en Python usando listas, donde cada nodo contiene únicamente el código entero de un ejemplar y mediante operaciones de inserción, visualización, modificación y eliminación se realiza la gestión de los libros de la biblioteca. El programa presenta un menú interactivo en consola que permite al bibliotecario llevar a cabo todas las operaciones.

4. Metodología Utilizada

Durante el desarrollo del trabajo, se siguieron los siguientes pasos:

Investigación previa: Se consultaron materiales de la cátedra (videos y ejemplos prácticos), documentación oficial de Python y recursos adicionales como artículos y tutoriales sobre árboles binarios implementados con listas.

Diseño y desarrollo del código: Se planificó la estructura del programa, definiendo qué funciones se necesitaban (crear nodo, insertar, eliminar, recorrer, visualizar, etc.), y luego se implementaron de forma modular.

Pruebas y validaciones: Se realizaron pruebas manuales desde consola para verificar el correcto funcionamiento de las operaciones, y se observaron los resultados esperados en cada recorrido y visualización.

Herramientas utilizadas: Python 3.x, entorno de desarrollo VS Code y GitHub como sistema de control de versiones y trabajo colaborativo.

Trabajo en equipo: Uno de los integrantes creó el repositorio, ambos colaboramos mediante commits en la misma rama, resolviendo conflictos de merge durante el proceso de integración del código final.

5. Resultados Obtenidos

Se logró implementar correctamente un árbol binario con listas que permite la gestión completa de un catálogo de libros mediante códigos numéricos.

Todas las funcionalidades propuestas están operativas: inserción, eliminación, modificación, visualización y recorridos.

Se corrigieron errores relacionados con conflictos de Git en el proceso de trabajo colaborativo.

El programa muestra la información de manera clara, interactiva, y puede ser utilizado como base para ampliaciones futuras, como agregar títulos de libros.

El proyecto fue versionado y documentado en un repositorio público de GitHub, se adjunta el link al mismo:

https://github.com/nazcuy/TP_INTEGRADOR_ARBOL_BIN

6. Conclusiones

Realizar este trabajo nos permitió entender mejor el funcionamiento interno de los árboles binarios, especialmente cómo se recorren y manipulan usando listas y recursividad en Python.

Aunque el desarrollo fue progresivo, hubo momentos complejos, como el manejo de funciones recursivas. Además de presentarse dificultades técnicas al usar Git.

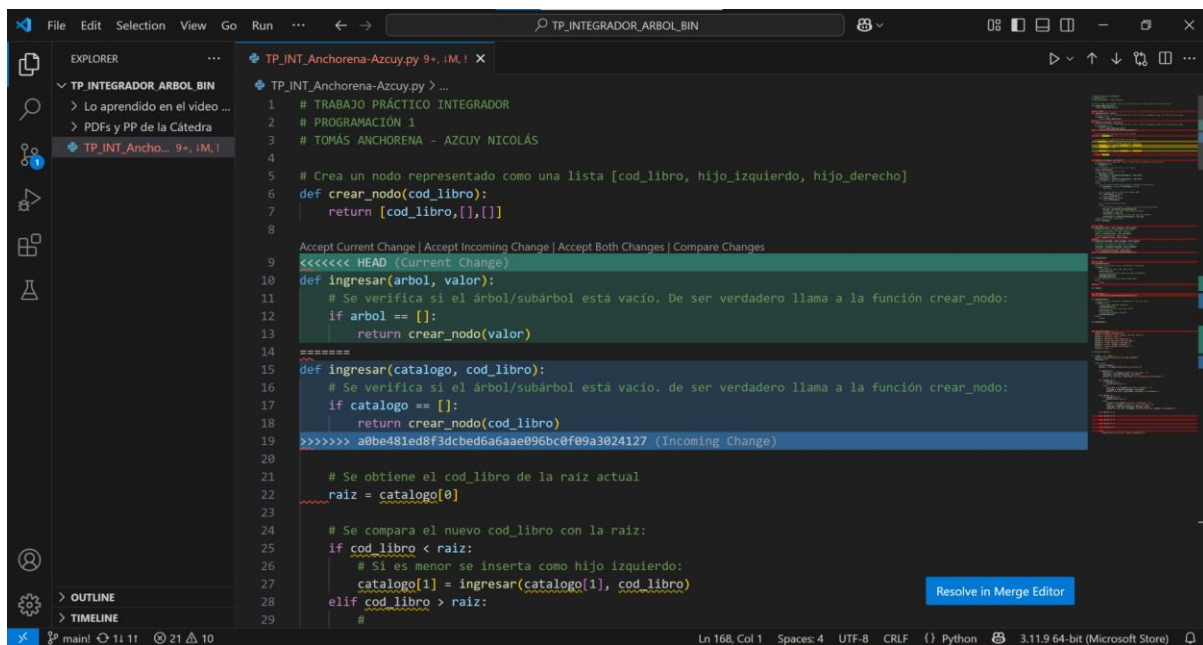
Como mejora futura, consideramos que sería útil implementar una visualización más clara del árbol. Por ejemplo, usar símbolos o una estructura jerárquica en consola para que los nodos se vean más “en forma de árbol”, y no solo como listas anidadas. Esto facilitaría la interpretación por parte del usuario y haría el programa más intuitivo.

7. Bibliografía

- Runestone Academy. (s.f.). *Representación de Árboles con Listas*.
<https://runestone.academy/ns/books/published/pythoned/Trees/RepresentacionDeListaDeListas.html>
 - YouTube – Robert Farfán. *Crear y Recorrer ABB*. [Cómo Crear y Recorrer Árboles Binarios en Python: Inorden, Preorden y Postorden](#)
 - Videos oficiales de la cátedra – Unidad: Datos Avanzados (UTN – Programación I)
-

8. Anexos

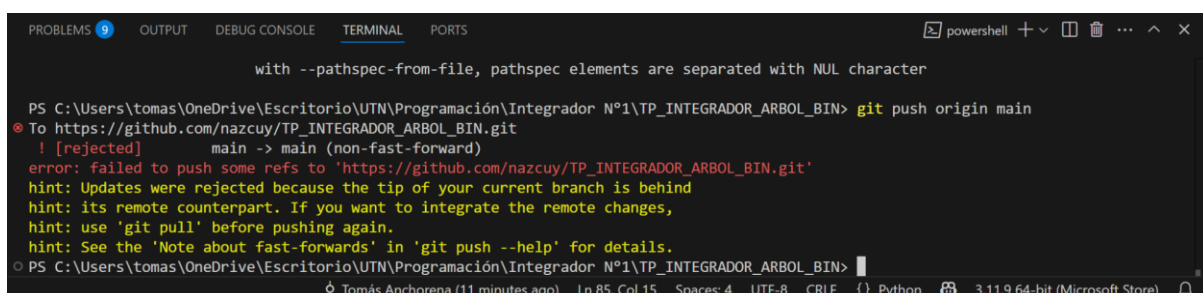
Aquí se adjuntan imágenes de un error de merge con GitHub:



```
1 # TRABAJO PRÁCTICO INTEGRADOR
2 # PROGRAMACIÓN 1
3 # TOMÁS ANCHORENA - AZCUY NICOLÁS
4
5 # Crea un nodo representado como una lista [cod_libro, hijo_izquierdo, hijo_derecho]
6 def crear_nodo(cod_libro):
7     return [cod_libro,[],[]]
8
9 <<<<<< HEAD (Current Change)
10 def ingresar(arbol, valor):
11     # Se verifica si el árbol/subárbol está vacío. De ser verdadero llama a la función crear_nodo:
12     if arbol == []:
13         return crear_nodo(valor)
14
15 >>>>>> a0be481ed8f3dcbed6a6aae096bc0f09a3024127 (Incoming Change)
16 def ingresar(catalogo, cod_libro):
17     # Se verifica si el árbol/subárbol está vacío. de ser verdadero llama a la función crear_nodo:
18     if catalogo == []:
19         return crear_nodo(cod_libro)
20
21 # Se obtiene el cod_libro de la raíz actual
22 raiz = catalogo[0]
23
24 # Se compara el nuevo cod_libro con la raíz:
25 if cod_libro < raiz:
26     # Si es menor se inserta como hijo izquierdo:
27     catalogo[1] = ingresar(catalogo[1], cod_libro)
28 elif cod_libro > raiz:
29     #
```

Lo primero fue aplicar cambios en las porciones de código correspondientes, aquí en estos casos se aplicó el “Accept Incoming Change” para luego pushear los cambios.

Al hacer el push, salta este error

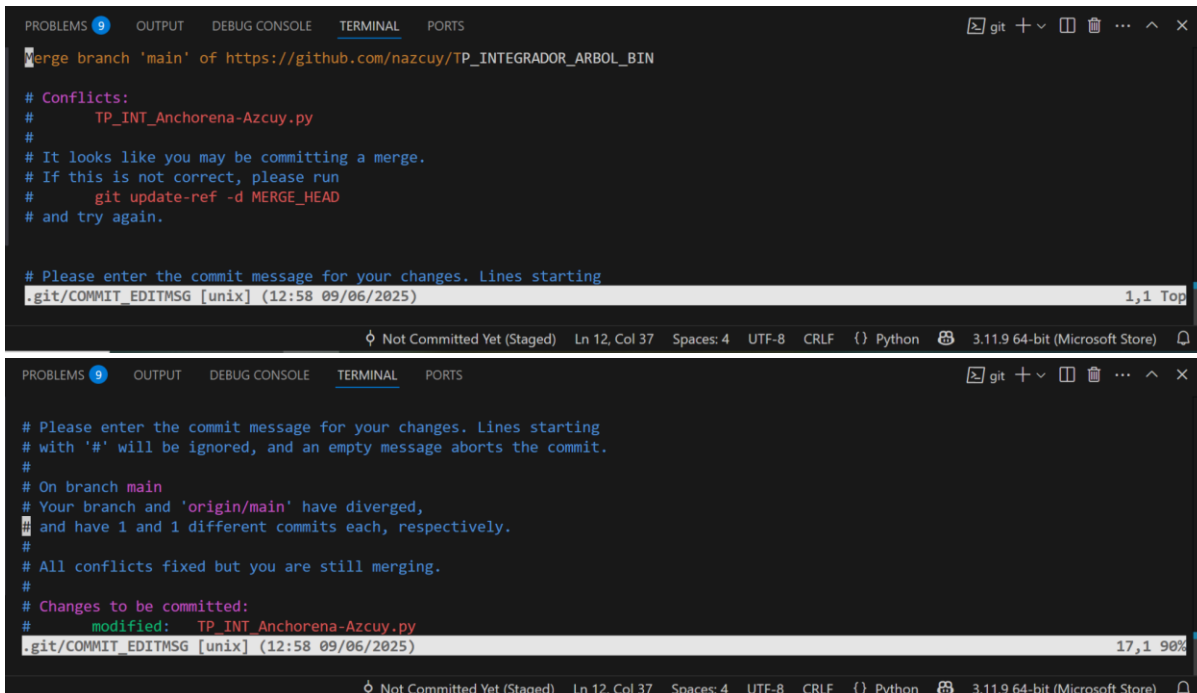


```
with --paths-from-file, pathspec elements are separated with NUL character

PS C:\Users\tomas\OneDrive\Escritorio\UTN\Programación\Integrador N°1\TP_INTEGRADOR_ARBOL_BIN> git push origin main
To https://github.com/nazcuy/TP_INTEGRADOR_ARBOL_BIN.git
! [rejected] main -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/nazcuy/TP_INTEGRADOR_ARBOL_BIN.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. If you want to integrate the remote changes,
hint: use 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
PS C:\Users\tomas\OneDrive\Escritorio\UTN\Programación\Integrador N°1\TP_INTEGRADOR_ARBOL_BIN>
```

Habiendo ocurrido esto lo que se hizo fue ejecutar un ‘git pull’ como me señalaban las hints de GitHub.

Al hacer el pull, aparece un mensaje en la terminal diciendo que debía ejecutar únicamente ‘git commit’ acá aprendimos / descubrimos que ejecutando solamente ‘git commit’ se abre una terminal tipo Vim en la propia terminal de VSCode y ahí es donde teníamos que ejecutar los cambios.



```
PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Merge branch 'main' of https://github.com/nazcuy/TP_INTEGRADOR_ARBOL_BIN

# Conflicts:
#   TP_INT_Anchorena-Azcuy.py
#
# It looks like you may be committing a merge.
# If this is not correct, please run
#   git update-ref -d MERGE_HEAD
# and try again.

# Please enter the commit message for your changes. Lines starting
.git/COMMIT_EDITMSG [unix] (12:58 09/06/2025) 1,1 Top

Not Committed Yet (Staged) Ln 12, Col 37 Spaces: 4 UTF-8 CRLF {} Python 3.11.9 64-bit (Microsoft Store)

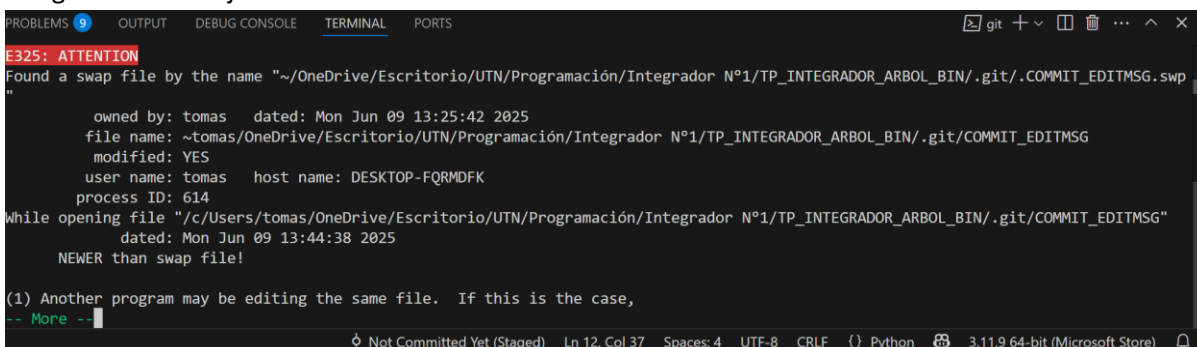
PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Your branch and 'origin/main' have diverged,
# and have 1 and 1 different commits each, respectively.
#
# All conflicts fixed but you are still merging.
#
# Changes to be committed:
#   modified:   TP_INT_Anchorena-Azcuy.py
.git/COMMIT_EDITMSG [unix] (12:58 09/06/2025) 17,1 90%

Not Committed Yet (Staged) Ln 12, Col 37 Spaces: 4 UTF-8 CRLF {} Python 3.11.9 64-bit (Microsoft Store)
```

En esta terminal no logré entender que debía hacer, consulté a ChatGPT y me dijo qué tenía que hacer, pero al intentar escribir en esta terminal no lograba mostrar nada, no podía escribir. Decido a clickear en Kill Terminal para volver a arrancar de 0 y buscar otra alternativa, pero terminé volviendo a lo mismo.

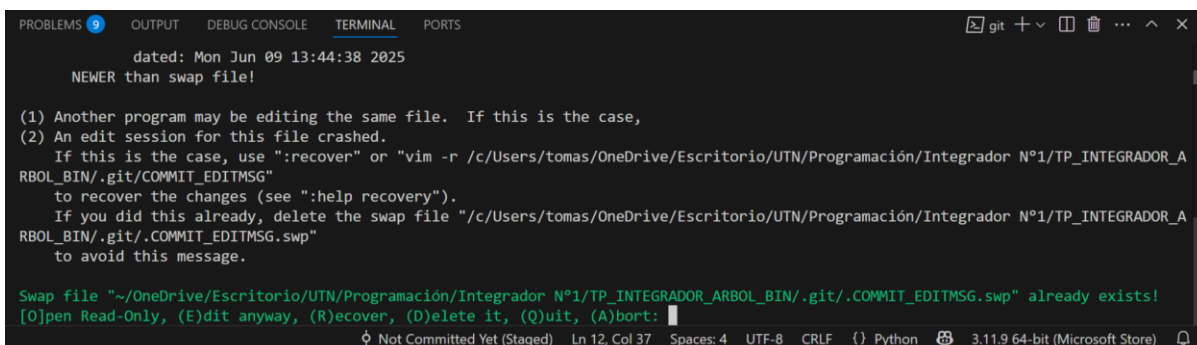
Luego de esto ejecuto otra vez 'git commit' para ver si ahora podía hacer modificaciones y apareció el siguiente mensaje



```
PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS
E325: ATTENTION
Found a swap file by the name "~/OneDrive/Escritorio/UTN/Programación/Integrador N°1/TP_INTEGRADOR_ARBOL_BIN/.git/.COMMIT_EDITMSG.swp"
owned by: tomas   dated: Mon Jun 09 13:25:42 2025
file name: ~/tomas/OneDrive/Escritorio/UTN/Programación/Integrador N°1/TP_INTEGRADOR_ARBOL_BIN/.git/COMMIT_EDITMSG
modified: YES
user name: tomas   host name: DESKTOP-FQRMDFK
process ID: 614
While opening file "/c/Users/tomas/OneDrive/Escritorio/UTN/Programación/Integrador N°1/TP_INTEGRADOR_ARBOL_BIN/.git/COMMIT_EDITMSG"
dated: Mon Jun 09 13:44:38 2025
NEWER than swap file!

(1) Another program may be editing the same file. If this is the case,
-- More --

Not Committed Yet (Staged) Ln 12, Col 37 Spaces: 4 UTF-8 CRLF {} Python 3.11.9 64-bit (Microsoft Store)
```



```
PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS
dated: Mon Jun 09 13:44:38 2025
NEWER than swap file!

(1) Another program may be editing the same file. If this is the case,
(2) An edit session for this file crashed.
If this is the case, use ":recover" or "vim -r /c/Users/tomas/OneDrive/Escritorio/UTN/Programación/Integrador N°1/TP_INTEGRADOR_A
RBOL_BIN/.git/COMMIT_EDITMSG"
to recover the changes (see ":help recovery").
If you did this already, delete the swap file "/c/Users/tomas/OneDrive/Escritorio/UTN/Programación/Integrador N°1/TP_INTEGRADOR_A
RBOL_BIN/.git/.COMMIT_EDITMSG.swp"
to avoid this message.

Swap file "~/OneDrive/Escritorio/UTN/Programación/Integrador N°1/TP_INTEGRADOR_ARBOL_BIN/.git/.COMMIT_EDITMSG.swp" already exists!
[O]pen Read-Only, [E]dit anyway, [R]ecover, [D]elete it, [Q]uit, [A]bort:

Not Committed Yet (Staged) Ln 12, Col 37 Spaces: 4 UTF-8 CRLF {} Python 3.11.9 64-bit (Microsoft Store)
```

Acá se presiona D para borrar este “swap”, volver a la terminal de Vim, ahora siguiendo (con un poco más de idea) a la IA presiono Esc, seguido de las teclas :wq y Enter. Esto se hizo para salir de esa terminal y supuestamente poder pushear mis cambios al repositorio, que terminó sin resolverse, me volvía a aparecer el error de que antes debo ejecutar el pull y volví al ciclo. Al final terminé desistiendo y cerré la carpeta, mis cambios los dejé anotados en otra carpeta, cloné el repositorio de nuevo y pegué mis cambios en este nuevo clon.

Seguido a esto, cuando tuve que commitear de vuelta, pude hacerlo sin problema y todo lo hecho pudo subirse al repositorio remoto (la segunda vez, la primera vez falló)

Link al código directo:

https://github.com/nazcuy/TP_INTEGRADOR_ARBOL_BIN/blob/main/TP_INT_Anchorena-Azcuy.py