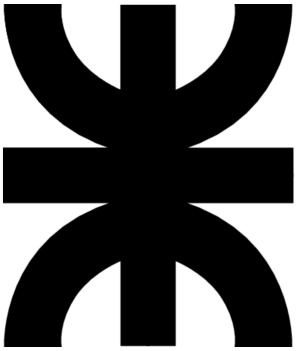
Universidad Tecnológica Nacional
Facultad Regional San Nicolás

PROGRAMACIÓN I



Trabajo Práctico Integrador

“Estructuras de datos avanzadas: árboles”

Integrantes:
Ríos Jesús Jonathan
Rodriguez Maximiliano Xavier

Profesor:
Nicolás Quirós

Comisión: 20

Fecha de entrega: 09/06/2025

Tecnicatura Universitaria en Programación a Distancia
2025

Índice

1. Introducción

El presente trabajo integrador tiene como objetivo aplicar el concepto de árboles en Python mediante el desarrollo de un juego interactivo llamado "Adivina el Animal". Este proyecto permite al usuario pensar en un animal y responder preguntas de "sí" o "no" hasta que el programa intente adivinarlo. Si no lo logra, aprende una nueva pregunta y animal, incorporándose al árbol de decisión, lo que le permite mejorar en futuras partidas.

El uso de árboles en este contexto resulta particularmente interesante debido a su aplicabilidad en campos como la inteligencia artificial, los sistemas expertos y el aprendizaje automático. Además, permite a los estudiantes experimentar con estructuras dinámicas que evolucionan según la interacción del usuario, fortaleciendo la comprensión de conceptos fundamentales como la recursión, las estructuras jerárquicas y la persistencia de datos.

2. Marco teórico

Un árbol es una estructura de datos no lineal que se utiliza para representar relaciones jerárquicas entre elementos. Se compone de nodos, donde cada nodo puede tener uno o más hijos. En el caso del árbol binario, cada nodo puede tener como máximo dos hijos: uno izquierdo y uno derecho.

Las principales características de los árboles binarios son:

- **Raíz:** nodo inicial del árbol.
- **Nodos internos:** nodos que tienen al menos un hijo.
- **Hojas:** nodos que no tienen hijos.
- **Altura:** cantidad de niveles desde la raíz hasta la hoja más profunda.

Los árboles de decisión son una aplicación específica que permiten tomar decisiones mediante una serie de preguntas binarias. Cada nodo representa una pregunta, y sus hijos representan las respuestas "sí" y "no". En este proyecto, los nodos se representan mediante una clase que contiene el texto de la pregunta o del animal, y referencias a los nodos hijos correspondientes.

La representación y manipulación del árbol se realiza de forma recursiva, y se implementa un sistema de almacenamiento en archivos de texto mediante recorrido preorden, lo que permite guardar y recuperar el estado del árbol entre ejecuciones del programa.

3. Caso práctico

Se desarrolló el juego "Adivina el Animal" con las siguientes funcionalidades:

- El programa inicia con un nodo raíz con un animal por defecto.
- A través de preguntas de "sí" o "no", intenta adivinar el animal que el usuario está pensando.
- Si no acierta, solicita al usuario el nombre del animal correcto y una pregunta que lo diferencie del animal propuesto.
- La nueva pregunta y el nuevo animal se incorporan al árbol de forma dinámica, ampliando el conocimiento del sistema.
- El árbol se guarda en un archivo de texto para mantener el aprendizaje entre partidas.
- Se incluye una opción para visualizar el árbol de decisiones de manera estructurada en la consola.

Este caso práctico permite simular un sistema básico de aprendizaje automático, donde el árbol se actualiza en tiempo real según la experiencia previa.

Fragmento de código - Nodo y Juego:

```
class NodoDecision:
    def __init__(self, texto):
        self.texto = texto
        self.si = None
        self.no = None

# Representación binaria de decisiones
# Ejemplo: ["Tiene plumas?", Nodo('Pájaro'), Nodo('Perro')]
```

4. Metodología utilizada

El desarrollo del trabajo siguió los siguientes pasos:

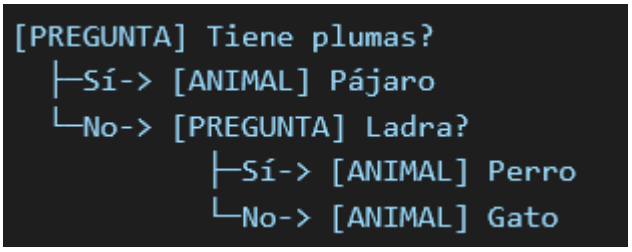
- **Selección del tema:** se optó por árboles de decisión con aprendizaje debido a su valor didáctico y su aplicabilidad en IA.
- **Investigación teórica:** se consultaron materiales bibliográficos y documentación oficial de Python sobre estructuras de datos y árboles.
- **Diseño del sistema:** se definió la estructura de las clases y funciones necesarias para representar, recorrer y modificar el árbol.
- **Implementación:** se desarrollaron módulos independientes para cada funcionalidad (juego, aprendizaje, guardado, visualización).
- **Pruebas:** se ejecutaron múltiples sesiones del juego para validar la persistencia y el correcto aprendizaje de nuevos animales.
- **Trabajo colaborativo:** ambos integrantes participaron activamente en todas las etapas, dividiendo tareas y revisando el código en conjunto.

5. Resultados obtenidos

El proyecto demostró ser exitoso en varios aspectos:

- El árbol de decisión se construye de manera correcta y se expande dinámicamente a partir de la interacción con el usuario.
- La información se guarda y recupera correctamente desde un archivo, permitiendo continuidad entre partidas.
- El sistema muestra un comportamiento creciente en su "inteligencia", aprendiendo nuevos animales en cada fallo.
- La visualización textual del árbol permite comprender la evolución de la estructura a lo largo del tiempo.
- Se validó el funcionamiento mediante diferentes pruebas con usuarios, obteniendo retroalimentación positiva sobre la dinámica del juego.

Ejemplo de visualización del árbol:



6. Conclusiones

El desarrollo de este trabajo permitió aplicar de manera concreta los conceptos de estructuras de datos avanzadas, específicamente árboles binarios y árboles de decisión. La implementación de un sistema que aprende progresivamente a través de la interacción con el usuario ofreció un ejemplo claro de cómo estos conceptos pueden ser utilizados para construir soluciones inteligentes.

Además de consolidar conocimientos sobre recursividad, estructuras jerárquicas y persistencia de datos, el proyecto fortaleció el trabajo en equipo, la planificación del desarrollo y el uso de buenas prácticas en programación. La experiencia resultó enriquecedora tanto en lo técnico como en lo formativo, permitiendo a los integrantes profundizar en el uso de Python aplicado a problemas reales.

El proyecto puede ser fácilmente ampliado en el futuro, por ejemplo, añadiendo una interfaz gráfica, un sistema de puntuación, o integración con bases de datos para mejorar el rendimiento y la escalabilidad.

7. Bibliografía

- Python Software Foundation. (2024). *Python 3 Documentation*. https://docs.python.org/3/
- Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). *Introduction to Algorithms*.
- Miller, B., & Ranum, D. *Problem Solving with Algorithms and Data Structures using Python*.
- Sweigart, A. (2019). *Automate the Boring Stuff with Python*. No Starch Press.

8. Anexos

- Capturas de pantalla del árbol en ejecución.
- Archivo “`arbol animales.txt`” con ejemplo de estructura guardada.
- Enlace al repositorio GitHub: https://github.com/jonathanriosok/TI-PROGRAMACION-I
- Enlace al video explicativo: https://youtu.be/2SAgacWTAso