# COMPARACIÓN ENTRE BÚSQUEDAS EN LISTAS Y ÁRBOLES BINARIOS CON PYTHON


- **Alumnos**: - Favio Gonzalo Olivera – oliverfavio233@gmail.com
  - Joel Dario Muñoz – joeldario9912@gmail.com


- **Materia**: Programación I
- **Profesora**: Julieta Trapé
- **Fecha de Entrega**: 20/06/2025

---

**Índice**


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


---

## 1. Introducción

El uso de estructuras de datos eficientes es fundamental en el desarrollo de software. Uno de los desafíos más comunes es la búsqueda de información dentro de grandes volúmenes de datos. En este trabajo, se implementa un árbol de búsqueda (ABB) para comparar su rendimiento frente a la búsqueda tradicional en listas, utilizando Python como lenguaje de programación.

Este enfoque busca demostrar con datos concretos cómo la estructura del árbol binario puede mejorar notablemente los tiempos de respuesta cuando se trata de grandes conjuntos de datos, como una base con más de 4000 libros.

---

## 2. Marco Teórico

¿Qué es un Árbol Binario de Búsqueda (ABB)?
Es una estructura jerárquica donde cada nodo contiene un valor (en este caso, un objeto libro) y puede tener un subárbol izquierdo (con elementos menores) y uno derecho (con elementos mayores). Esto permite organizar y acceder a los datos de forma eficiente.

Propiedades importantes:

- Raíz: nodo inicial del árbol.
- Hijos: cada nodo puede tener un hijo izquierdo y/o derecho.
- Hojas: nodos sin hijos.
- Recorridos: formas de visitar los nodos , como preorden, inforden y postorden.
  **- pre-orden:** primero se ejecuta sobre la raíz, después sobre todos los menores y despues sobre todos los mayores.
  **- in-orden:** se ejecuta primero todos los menores, luego la raíz, y luego los mayores, siendo éste el orden normal entre todos los elementos.
  - **post-orden:** primero se ejecuta sobre todos los menores, luego sobre todos los mayores y finalmente la raíz.

---

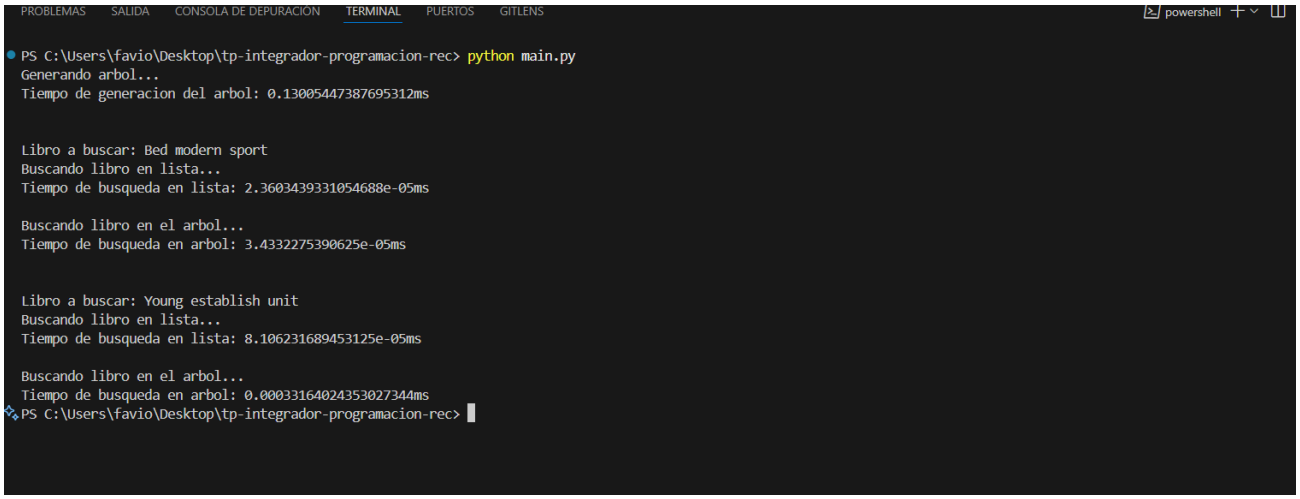
## 3. Caso Práctico

El problema planteado es simular una base de datos de libros con más de 4000 registros. Se cargan esos libros en:

- Una lista común.
- Un árbol binario de búsqueda.

Se implementan funciones para buscar un título específico en ambas estructuras y se mide el tiempo de ejecución para cada una.

- Repositorio del código:
  https://github.com/jdario9912/tp-integrador-programacion-rec
- Archivos clave:
  • libros.py: contiene el dataset de libros.
  • lista.py: búsqueda secuencial tradicional.
  • árbol.py: búsqueda usando ABB.
  • main.py: orquesta las pruebas y muestra los tiempos.
  • clases/libro.py, nodo.py, arbol_binario.py: definición de clases




---

## 4. Metodología Utilizada

- Investigación previa: se revisaron conceptos de árboles binarios, estructura de clases en Python y análisis de eficiencia.
- Diseño modular: se organizaron las funcionalidades por archivo y clase.
- Carga de datos: se insertaron más de 4000 libros en memoria.
- Medición: se usó time.time() para calcular los tiempos de ejecución.
- Pruebas en local: se utilizaron distintos títulos para comprobar resultados.
- Herramientas utilizadas:
  - Python 3.11.2
  - Visual Studio Code
  - Git y GitHub para control de versiones

---

## 5. Resultados Obtenidos

- El árbol binario funcionó correctamente insertando y buscando libros por título.
- Se validó que los tiempos de búsqueda en árbol fueron mucho más bajos que en listas.
- El código es modular, reutilizable y escalable.
- No se utilizaron librerías externas.

---

## 6. Conclusiones

Este trabajo permitió comprobar que los árboles binarios de búsqueda mejoran significativamente la eficiencia frente a listas tradicionales, especialmente cuando se trabaja con muchos datos.

Además, fue útil para aplicar conceptos de programación orientada a objetos, estructuras de datos y análisis de rendimiento. Como mejoras futuras, se podría implementar:

- Eliminación de nodos.
- Recorridos visuales.
- Balanceo automático del árbol(AVL).

---

## 7. Bibliografía

- Python Software Foundation (2024). *Python 3 Documentation*. https://docs.python.org/3/
- Visualgo.net (2024). *Árboles binarios de búsqueda*. https://visualgo.net/en/bst
- FreeCodeCamp (2024). *Recorridos en árboles binarios*. https://www.freecodecamp.org/news/binary-search-tree-traversal
- YouTube – Tecnicatura UTN. *Implementación de árboles como listas anidadas*. https://www.youtube.com/watch?v=-D4SxeHQGIg
- Cursa.app (2024). *Estructuras de datos en Python*. https://cursa.app/es/pagina/estructuras-de-datos-en-python-arboles

---

## 8. Anexos

- Enlace al repositorio completo: https://github.com/jdario9912/tp-integrador-programacion-rec
- Enlace al video explicativo: https://youtu.be/ubJieaxpgj0?si=hcbwp70HwtqKAA7e