Conclusiones

A lo largo de esta investigación, pudimos profundizar en el análisis comparativo de distintos algoritmos de ordenamiento y búsqueda, así como en el uso de estructuras avanzadas como los árboles binarios y AVL, implementándolos en el lenguaje Python.

Al ejecutar los algoritmos con conjuntos de datos de diferente tamaño, observamos que:

- BubbleSort, aunque fácil de implementar, se vuelve ineficiente con listas grandes debido a su complejidad cuadrática.

- MergeSort y QuickSort demostraron mejor rendimiento, siendo QuickSort especialmente eficiente en promedio.

- La búsqueda binaria y los recorridos en árboles (DFS y BFS) resultaron herramientas poderosas al aplicarse sobre estructuras ordenadas o jerárquicas.

Además, el uso de mediciones prácticas con time y timeit nos permitió validar los análisis teóricos con resultados empíricos.

Implementar árboles AVL nos permitió entender cómo las estructuras balanceadas ayudan a mantener la eficiencia incluso cuando se insertan muchos datos.

Como aprendizaje, destacamos la importancia de:

- Elegir el algoritmo correcto según el contexto del problema.

- Utilizar estructuras adecuadas para optimizar búsquedas.

- Medir y comparar el rendimiento, más allá de la teoría.

Este proyecto también nos permitió mejorar nuestra organización de código, usar herramientas como Git, y aplicar buenas prácticas de programación.