Trabajo Integrador: Algoritmos de Búsqueda y Ordenamiento

Alumnos: Rivas Alfredo Guillermo — a.guille.rivas@gmail.com
Re Julián — contacto.julian.re@gmail.com
Materia: Programación I
Profesor: Nicolás Quirós
Fecha de Entrega: 09/06/2025

Índice

---

1. Introducción

Este trabajo trata sobre dos temas clave en programación: la búsqueda y el ordenamiento de datos. Estos conceptos son esenciales para mejorar la eficiencia de los programas. El objetivo es aprender cómo funcionan estos algoritmos y compararlos mediante un caso práctico hecho en Python.

---

2. Marco Teórico

Un algoritmo es un conjunto de pasos para resolver un problema. Los algoritmos de búsqueda permiten encontrar datos específicos, y los de ordenamiento reorganizan los datos en un orden lógico.

Búsqueda:

- **Lineal:** recorre uno por uno hasta encontrar el dato.
- **Binaria:** mucho más rápida, pero solo funciona si la lista está ordenada.

Ordenamiento:

- **Bubble Sort:** compara e intercambia elementos adyacentes.
- **Quick Sort:** divide la lista en partes menores y mayores a un “pivote” y ordena por separado. Es más rápido.

---

3. Caso Práctico

Para aplicar los conceptos estudiados, desarrollamos un programa en Python que permite ordenar listas de números y buscar elementos específicos. Se implementaron dos algoritmos de ordenamiento (Bubble Sort y QuickSort) y dos algoritmos de búsqueda (Búsqueda Lineal y Búsqueda Binaria).

El código fue probado con una lista de 10.000 elementos generados aleatoriamente para medir la eficiencia de cada algoritmo. Se realizaron múltiples repeticiones para comparar los tiempos de ejecución.

El código fuente completo, con comentarios y pruebas realizadas, se encuentra en el **Anexo 3**.

---

4. Metodología Utilizada

- Revisión de apuntes y páginas confiables sobre el tema.
- Programación en Python.
- Generación de listas grandes usando la librería random.
- Medición de tiempos de ejecución con time.
- Pruebas comparando resultados.
- Grabación de un video explicativo.

---

5. Resultados Obtenidos

- **Bubble Sort** funcionó correctamente, pero fue muy lento con muchos datos.
- **Quick Sort** ordenó mucho más rápido.
- La búsqueda binaria fue muy rápida, pero solo funcionó bien con listas ordenadas.
- Aprendimos a medir y comparar tiempos de ejecución.

---

6. Conclusiones

Estudiar estos algoritmos ayudó a entender cómo mejorar la eficiencia de un programa. Aprendimos que elegir bien el algoritmo depende del tipo de datos y del problema. El trabajo en equipo también fue importante para repartir tareas y organizar el desarrollo.

---

7. Bibliografía

- Python Software Foundation. (2024). *Python 3 Documentation*. https://docs.python.org/3/
- Cormen, T. et al. (2009). *Introduction to Algorithms*.
- Programación ATS. YouTube. https://www.youtube.com/@ProgramacionATS

- Geeks for Geeks. https://www.geeksforgeeks.org

---

8. Anexos

- Capturas del programa funcionando.
- Enlace al video: https://youtu.be/8Y3-NPNkHiE
- Repositorio del código: github.com/GuilleRivas4/Trabajo-_Integrador-_Re-_Rivas