Trabajo Práctico – Algoritmos de Búsqueda y Ordenamiento en Python

Alumnos:

Nelson, Cristhian Alejandro – nelsoncristian822@gmail.com

Ocete, Rocio Milagros– rocio.familioct@gmail.com

Comisión 18

Materia:

Programación I

Profesor titular:

Trapé, Julieta

Profesor adjunto:

Vega, Marcos

Fecha de Entrega:

9 de junio de 2025

Índice

Introducción

En el mundo de la programación, hay herramientas que usamos casi sin darnos cuenta, pero que son esenciales para que todo funcione de forma ágil. Entre ellas están los algoritmos de búsqueda y ordenamiento. Elegimos este tema porque nos parecía una excelente oportunidad para explorar algo que parece simple, pero que detrás tiene mucho más de lo que parece.

Los algoritmos de búsqueda y ordenamiento son herramientas fundamentales para la manipulación eficiente de grandes volúmenes de información. Permiten localizar datos específicos dentro de una estructura y organizarlos de manera que faciliten su análisis, visualización o procesamiento posterior.

La elección de un algoritmo de búsqueda u ordenamiento puede impactar significativamente en el rendimiento de un sistema, especialmente cuando se manejan conjuntos de datos extensos o cuando se requiere una respuesta en tiempo real.

A lo largo de nuestra investigación, se explorarán los principales tipos de algoritmos de búsqueda y ordenamiento, sus características, ventajas, desventajas y aplicaciones prácticas, con el objetivo de comprender su funcionamiento y saber cuándo aplicar cada uno según el contexto del problema.

Marco Teórico

¿Qué es un algoritmo?

Un algoritmo es básicamente un conjunto de pasos que nos permite resolver un problema. En programación, usamos algoritmos constantemente, aunque muchas veces no los llamemos así.

En este trabajo nos enfocamos en dos tipos: los de búsqueda, que sirven para encontrar algo dentro de un conjunto de datos, y los de ordenamiento, que reorganizan esos datos para que estén en un orden determinado.

Algoritmo de búsqueda:

- **Lineal**: revisa uno por uno los elementos de la lista. Es simple, pero puede ser lento si la lista es larga. Complejidad O(n).
- **Binaria**: solo funciona si la lista ya está ordenada. Divide la búsqueda a la mitad en cada paso. Mucho más rápida. Complejidad O(log n).

Algoritmo de ordenamiento:

- **Burbuja**: compara elementos adyacentes y los intercambia si están desordenados. Es fácil de entender, pero poco eficiente. $O(n^2)$.
- **Inserción**: construye la lista ordenada insertando elementos en su lugar correcto. $O(n^2)$.
- **Quicksort**: elige un pivote y divide la lista en dos partes para ordenar. Muy rápido en la práctica. O(n log n).

Todos estos conceptos fueron abordados en profundidad en los siguientes materiales:

- Introducción al Análisis de Algoritmos (Ariel Enferrel)
- Análisis de Algoritmo Teórico y Big-O
- Análisis Empírico de Algoritmos
- Notación Big-O

Caso Práctico

Creamos un programa en Python que permite:

- Ingresar una lista de números.
- Ordenarla con uno de tres métodos: burbuja, inserción o Quicksort.
- Buscar un número con búsqueda lineal o binaria.
- Por ultimo el programa tiene la función de finalizar el programa

Todo el código está comentado y disponible en el repositorio de GitHub.
Elegimos incluir Quicksort no solo por su eficiencia, sino también porque nos interesaba entender cómo funciona un algoritmo recursivo más avanzado.

Metodología Utilizada

El trabajo lo organizamos en varias etapas:

1. Investigamos teoría en fuentes como libros, presentaciones de clase y documentación oficial.
2. Fuimos probando el programa con distintos conjuntos de datos para asegurarnos de que funcionara correctamente.
3. Finalmente, escribimos este informe y grabamos el video explicativo para presentar la experiencia.

Resultados Obtenidos

- El programa funcionó correctamente en todos los casos de prueba.
- Quicksort demostró ser más rápido que los otros dos métodos, sobre todo en listas largas.
- La búsqueda binaria fue muy eficiente, siempre que la lista estuviera previamente ordenada.

Conclusiones

Los algoritmos de búsqueda y ordenamiento son fundamentales para la gestión eficiente de la información y su estudio permite comprender mejor cómo funcionan internamente los programas informáticos y cómo optimizar su rendimiento.

Hacer este trabajo nos ayudó a ver que entender bien los algoritmos no es solo teoría, sino algo que realmente cambia cómo programamos. No todos los algoritmos sirven para todos los casos, y muchas veces una buena elección puede ahorrar mucho tiempo y recursos. También aprendimos a trabajar en equipo y a documentar correctamente nuestro código y nuestras decisiones.

Nos quedamos con la idea de que seguir profundizando en temas como este es clave para crecer como programadores.

Bibliografía

- Introducción al Análisis de Algoritmos. Ariel Enferrel (2025).
- Análisis de Algoritmo Teórico y Big-O. Tecnicatura Universitaria en Programación a Distancia (2025).
- Análisis Empírico de Algoritmos. Tecnicatura Universitaria en Programación a Distancia (2025).
- Notación Big-O. Tecnicatura Universitaria en Programación a Distancia (2025).
- Presentación "Análisis Teórico de Algoritmos" (Gamma App, 2025).

Anexos

- Capturas de pantalla del programa funcionando.
- Enlace a repositorio Git
- Enlace al video explicativo

Capturas de Funcionamiento:

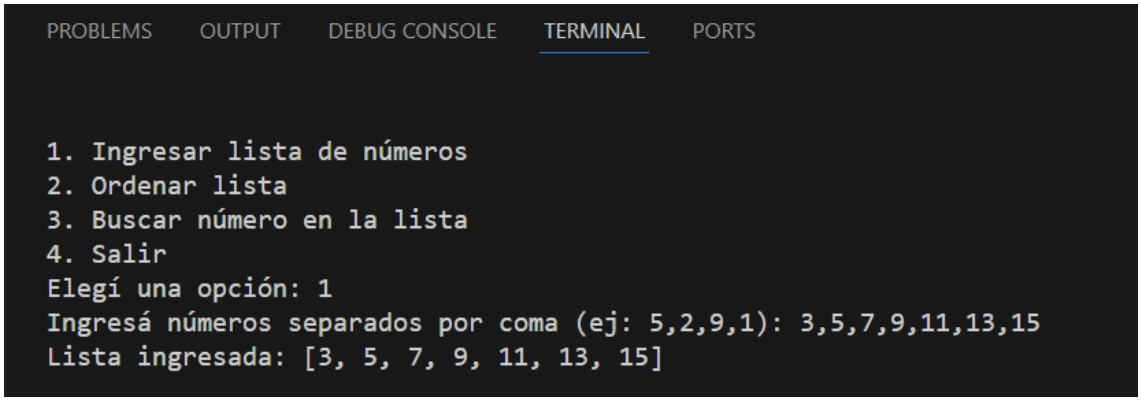

Imagen 1 – Funcionamiento del Ingreso de Lista de Números

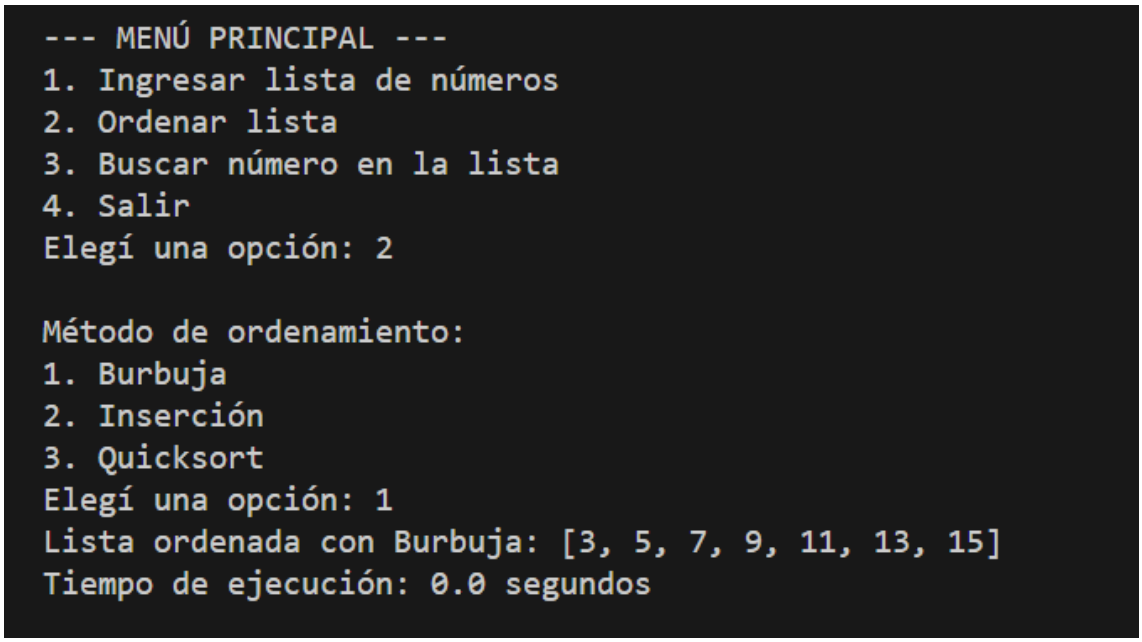

Imagen 2 – Funcionamiento del Ordenamiento de Burbuja

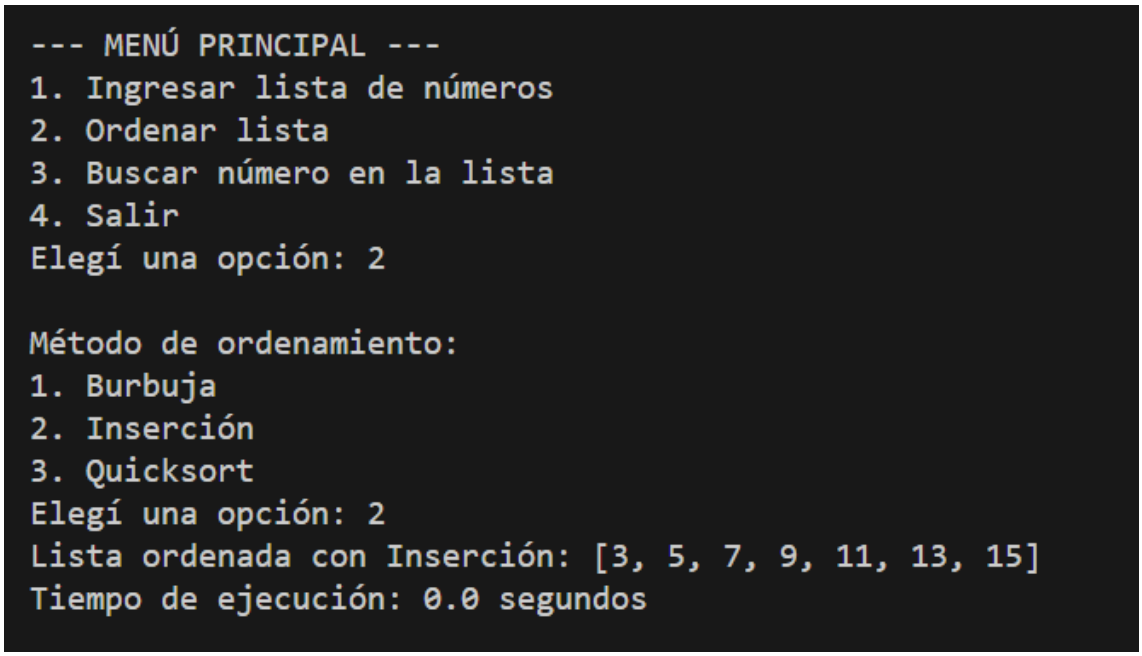

Imagen 3 – Funcionamiento de Ordenamiento de Inserción

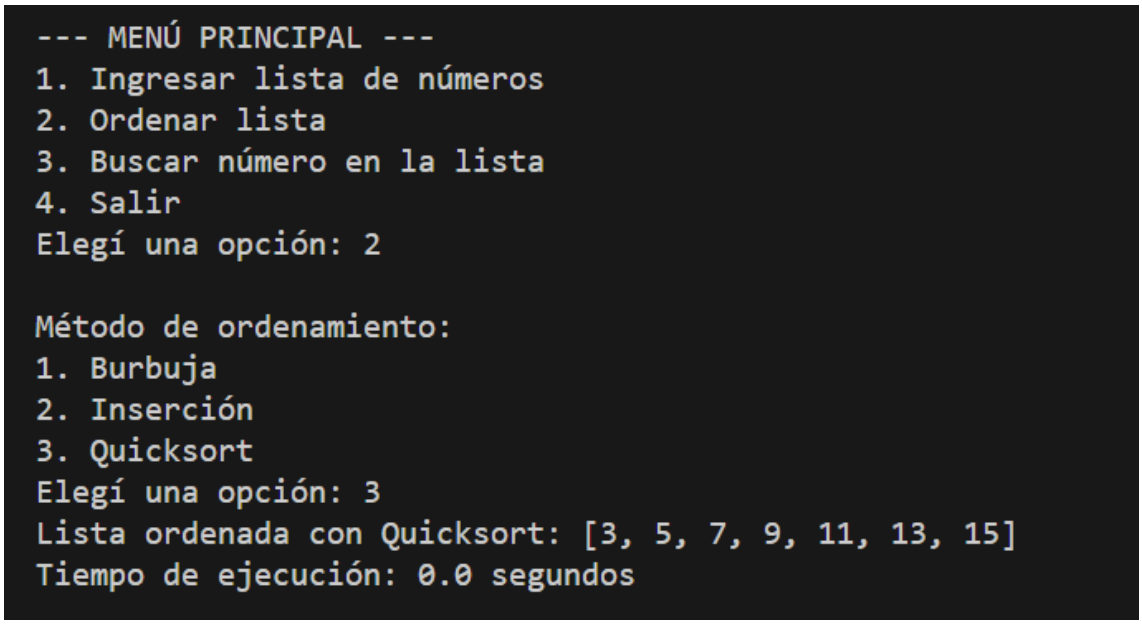

Imagen 4 – Funcionamiento de Ordenamiento de Quicksort

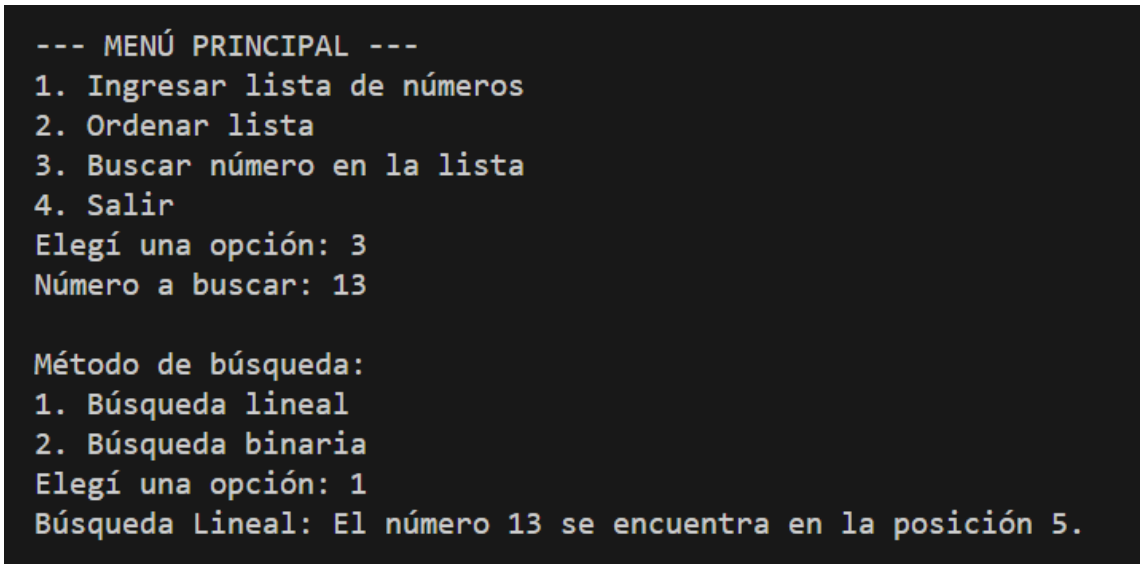

Imagen 5 – Funcionamiento de Búsqueda Lineal

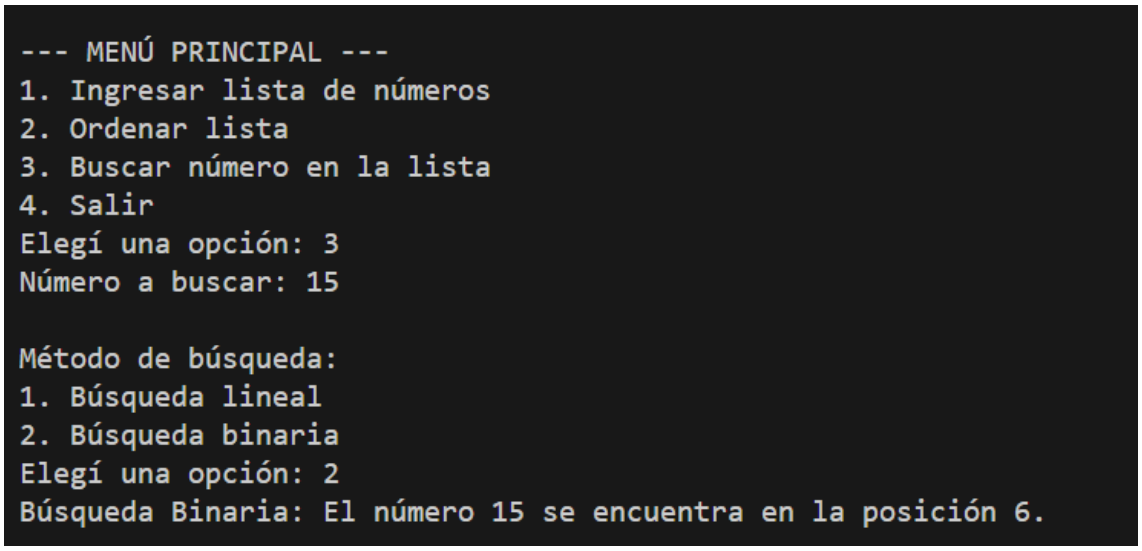

Imagen 6 – Funcionamiento de Búsqueda Binaria

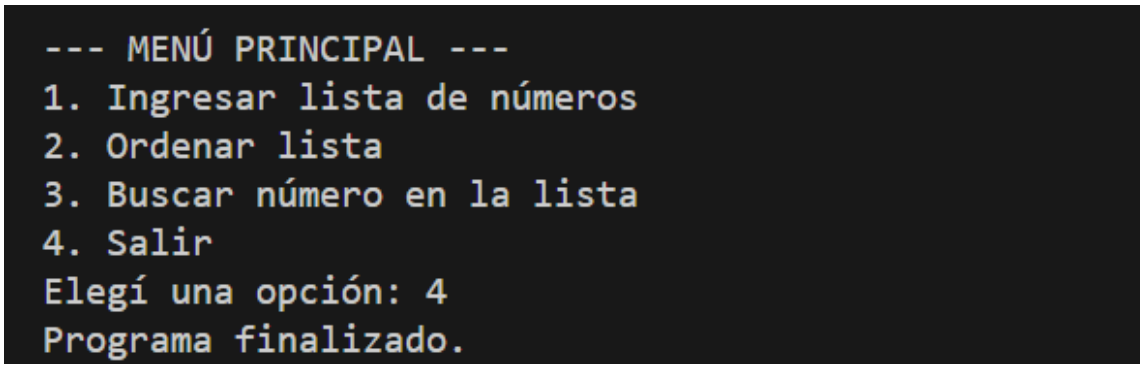

Imagen 7 – Funcionamiento de la Opción 4: Finalización de Programa