Algoritmos de Búsqueda y Ordenamiento

Alumnos:

- Enzo Sebastian Medina, 41158049 @-enzomedina7498@gmail.com
- Santiago Pérez Aguirre, 47084625 @santiagoperezaguirre910@gmail.com

Materia: Programación I
Profesor: Julieta Trapé
Fecha de Entrega: 9 de junio de 2025
Índice

---

1. Introducción

En el campo de la programación, la gestión eficiente de la información es un pilar fundamental. Los algoritmos de búsqueda y ordenamiento son herramientas esenciales que permiten manipular grandes volúmenes de datos de manera rápida y efectiva. Comprender su funcionamiento e implementación no solo es clave para resolver problemas computacionales, sino también para optimizar el rendimiento de cualquier software.
Este trabajo tiene como objetivo principal investigar e implementar algunos de los algoritmos de ordenamiento y búsqueda más conocidos. Para ello, se ha desarrollado un caso práctico en Python que simula la gestión del catálogo de una tienda de videojuegos, "GameHub". A través de este proyecto, se busca demostrar la aplicación real de algoritmos como Bubble Sort, Selection Sort, Búsqueda Lineal y Búsqueda Binaria, y analizar sus diferencias y contextos de uso.

---

2. Marco Teórico

Algoritmos de Ordenamiento

Un algoritmo de ordenamiento es un procedimiento que permite reorganizar los elementos de una lista o colección según un criterio de orden específico (numérico, alfabético, etc.).

- Bubble Sort (Ordenamiento de Burbuja): Es uno de los algoritmos más sencillos. Funciona revisando repetidamente una lista, comparando elementos adyacentes e intercambiándolos si están en el orden incorrecto. Este proceso se repite hasta que la lista está completamente ordenada. Su principal desventaja es su ineficiencia en listas

grandes, con una complejidad temporal de O(n2).

- Selection Sort (Ordenamiento por Selección): Este algoritmo mejora ligeramente al de burbuja. En cada pasada, busca el elemento más pequeño (o más grande) de la porción no ordenada de la lista y lo coloca al principio. Aunque realiza menos intercambios que Bubble Sort, su complejidad temporal también es de O(n2), lo que lo hace poco práctico para grandes volúmenes de datos.

- Ordenamiento Nativo de Python (sorted()): Python incluye una función de ordenamiento sorted() altamente optimizada, que utiliza un algoritmo llamado Timsort. Timsort es un híbrido que combina las ventajas de Insertion Sort y Merge Sort, ofreciendo un rendimiento muy superior en la mayoría de los casos del mundo real, con una complejidad promedio de O(nlogn).

Algoritmos de Búsqueda

Los algoritmos de búsqueda se utilizan para encontrar uno o más elementos que cumplen con ciertas propiedades dentro de una estructura de datos.

- Búsqueda Lineal (Linear Search): Es el método más simple. Recorre la lista elemento por elemento desde el inicio hasta el final, comparando cada uno con el valor buscado. Si lo encuentra, retorna su posición; de lo contrario, llega al final de la lista. Su principal desventaja es que, en el peor de los casos, debe recorrer toda la lista, resultando en una complejidad de O(n).

- Búsqueda Binaria (Binary Search): Es un algoritmo mucho más eficiente, pero requiere una condición fundamental: la lista debe estar previamente ordenada. Funciona de la siguiente manera:

    1. Compara el elemento central de la lista con el valor buscado.
    2. Si son iguales, la búsqueda termina con éxito.
    3. Si el valor buscado es menor que el elemento central, la búsqueda se repite en la mitad izquierda de la lista.
    4. Si es mayor, se repite en la mitad derecha. Este proceso de "dividir y conquistar" reduce drásticamente el tiempo de búsqueda, dándole una complejidad de O(logn).

---

3. Caso Práctico

Para aplicar los conceptos teóricos, se desarrolló un programa en Python que gestiona el catálogo de una tienda de videojuegos ficticia llamada "GameHub". El sistema permite al usuario ordenar el catálogo por precio o por unidades vendidas, y buscar juegos específicos por su nombre.

Decisiones de Diseño:

- Se eligió una lista de diccionarios para representar los datos, ya que permite almacenar múltiples atributos (nombre, precio, unidades) para cada videojuego de forma clara.
- Los algoritmos bubble_sort y selection_sort fueron adaptados para recibir un parámetro clave, lo que los hace reutilizables para ordenar por cualquier atributo del diccionario (precio, unidades, etc.).
- Para la búsqueda binaria, se ordena primero la lista por nombre, cumpliendo con el prerrequisito del algoritmo.

---

4. Metodología Utilizada

El desarrollo de este trabajo siguió una serie de etapas bien definidas, de acuerdo con las pautas de la cátedra:

1. Investigación Teórica: Se consultaron fuentes como la documentación oficial de Python y bibliografía recomendada para afianzar los conceptos de algoritmos.
2. Diseño y Desarrollo: Se diseñó el caso práctico "GameHub" y se implementaron los algoritmos en Python, utilizando Visual Studio Code como entorno de desarrollo.
3. Pruebas y Validación: Se ejecutó el programa de forma interactiva, probando cada una de las opciones del menú con el conjunto de datos definido para verificar su correcto funcionamiento.
4. Documentación: Se documentó el código fuente, se elaboró este informe y se preparó el material para la presentación audiovisual.

---

5. Resultados Obtenidos

El programa tiendajuegos.py se ejecutó correctamente y arrojó los siguientes resultados, validando el funcionamiento de los algoritmos implementados.
Resultado 1: Ordenamiento por Precio (Bubble Sort) Al seleccionar la opción 1, el catálogo se ordenó de menor a mayor precio:

🔮 CATÁLOGO ORDENADO POR PRECIO:

🎮 Minecraft - $29.99 (Vendidos: 500)
🎮 Cyberpunk 2077 - $39.99 (Vendidos: 200)
🎮 Animal Crossing - $49.99 (Vendidos: 350)
🎮 The Legend of Zelda - $59.99 (Vendidos: 150)
🎮 FIFA 23 - $59.99 (Vendidos: 300)

Resultado 2: Ordenamiento por Ventas (Selection Sort) La opción 2 ordenó el catálogo según las unidades vendidas, de menor a mayor:

🔮 CATÁLOGO ORDENADO POR VENTAS:

🎮 The Legend of Zelda - $59.99 (Vendidos: 150)
🎮 Cyberpunk 2077 - $39.99 (Vendidos: 200)
🎮 FIFA 23 - $59.99 (Vendidos: 300)
🎮 Animal Crossing - $49.99 (Vendidos: 350)
🎮 Minecraft - $59.99 (Vendidos: 500)

Resultado 3: Búsqueda Binaria Al elegir la opción 5 y buscar "Cyberpunk 2077", el programa primero ordena la lista alfabéticamente (un paso interno necesario) y luego encuentra el juego eficientemente:
🔍 Ingrese nombre del juego a buscar: Cyberpunk 2077

🔍 RESULTADO BÚSQUEDA BINARIA:

🎮 Cyberpunk 2077 - $39.99 (Vendidos: 200)

Estos resultados prácticos permitieron comprender las diferencias de aplicación entre los algoritmos y la importancia de la estructura de datos para su correcto funcionamiento.

---

6. Conclusiones

La realización de este trabajo integrador nos ha permitido afirmar que los algoritmos de búsqueda y ordenamiento son pilares en la ciencia de la computación. El desarrollo del caso práctico "GameHub" fue fundamental para llevar la teoría a un escenario tangible y comprender cómo estas herramientas pueden optimizar tareas cotidianas como la gestión de un inventario.
Aprendimos que, si bien algoritmos como Bubble Sort o Selection Sort son valiosos a nivel académico para entender la lógica del ordenamiento, en aplicaciones reales es casi siempre preferible utilizar las funciones nativas y optimizadas que ofrecen los lenguajes modernos como Python (sorted()).
La mayor lección fue la clara diferencia de eficiencia entre la búsqueda lineal y la binaria, y cómo una simple condición —tener los datos ordenados— puede transformar una operación lenta en una casi instantánea. Este proyecto refuerza la idea de que la elección del algoritmo correcto, según el contexto y los requisitos del problema, es una de las habilidades más importantes para un programador.

---

7. Bibliografía

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms.
- Python Software Foundation. (2025). Python 3 Documentation. Recuperado de https://docs.python.org/3/

- Khan Academy. (s.f.). Algoritmos. Recuperado de https://es.khanacademy.org/computing/computer-science/algorithms

---

8. Anexos

- Repositorio en GitHub:
- Video Explicativo: