Programación
Búsqueda y Ordenamiento

Alumnos: Fausto Gagliano (Comisión 2) faustogagliano05@gmail.com

Fernando Aguiar (Comisión 1) fercovichargento@gmail.com

Materia: Programación I
Profesor/a: Cinthia Rigoni
Fecha de Entrega: 09/06/25

Índice

Introducción

Los algoritmos de Búsqueda y Ordenamiento permiten localizar y organizar datos eficientemente, lo que es crucial para el desempeño de programas en ámbitos como bases de datos, sistemas de archivos y aplicaciones de inteligencia artificial.

El tema fue elegido debido a su capacidad de optimizar procesos computacionales que manejan grandes cantidades de información, mejorando la velocidad y precisión de las operaciones. El objetivo principal es comprender los conceptos teóricos, conocer diferentes algoritmos y aplicarlos en un lenguaje de programación actual como Python, para analizar su funcionamiento y eficiencia.

Marco Teórico

Búsqueda

La búsqueda es la acción de localizar un elemento específico dentro de un conjunto de datos. Se emplea en una amplia variedad de aplicaciones para extraer información relevante.

Tipos de Algoritmos de Búsqueda

Búsqueda lineal: Recorre secuencialmente cada elemento hasta encontrar el deseado o agotar la lista. Su complejidad es O(n), siendo simple pero ineficiente para grandes conjuntos.

Búsqueda binaria: Opera sobre listas ordenadas, dividiendo la lista en mitades sucesivas para localizar el elemento. Su complejidad O(log n) la hace muy eficiente para grandes volúmenes de datos.

Búsqueda por interpolación: Similar a la binaria, pero estima la posición probable del elemento según su valor.

Búsqueda por hash: Utiliza una función hash para asignar elementos a posiciones específicas en una tabla, permitiendo acceso en tiempo constante O(1).

Aplicaciones de la búsqueda incluyen:

Localización de palabras en documentos.

Búsqueda de archivos.

Recuperación de registros en bases de datos.

Rutas más cortas en grafos.

Solución de problemas de optimización.

Ordenamiento

El ordenamiento organiza los datos según un criterio, facilitando búsquedas más rápidas y operaciones eficientes.

Algoritmos de Ordenamiento comunes

Bubble Sort: Compara pares adyacentes e intercambia si están desordenados. Sencillo pero lento ($O(n^2)$).

Selection Sort: Selecciona el elemento mínimo y lo coloca en su posición correcta, repetidamente. Complejidad $O(n^2)$.

Insertion Sort: Inserta elementos uno a uno en su posición adecuada, eficiente para listas pequeñas o casi ordenadas ($O(n^2)$).

Quick Sort: Divide y conquista usando un pivote, con rendimiento promedio O(n log n).

Merge Sort: Divide la lista en mitades, ordena y fusiona, garantizando O(n log n).

Beneficios del ordenamiento:

Permite usar búsqueda binaria.

Facilita análisis de datos y detección de patrones.

Acelera operaciones como fusión y eliminación de duplicados.

Complejidad y Comparación

Tamaño de lista	Búsqueda Lineal (O(n))	Búsqueda Binaria (O
10	10	3
100	100	7
1000	1000	10
10000	10000	13
100000	10000	116

La búsqueda binaria crece mucho más lentamente, demostrando su eficiencia para listas grandes.

Caso Práctico

El caso práctico consiste en desarrollar un sistema sencillo para gestionar un inventario de productos, que permita agregar, eliminar, modificar y mostrar productos. Además, el sistema debe implementar algoritmos de búsqueda (lineal y binaria) y algoritmos de ordenamiento (Quicksort y Mergesort) para organizar y buscar productos eficientemente según diferentes atributos (ID, nombre, precio, cantidad).

(capturas, porque se eligio ese diseño y la validacion del funcionamiento)

Metodología Utilizada

Para realizar este proyecto utilizamos Quick sort y Merge sort para poder ordenar de forma lineal y binaria nuestro inventario. En las opciones te permite seleccionar cual de las 2 preferis utilizar.

Resultados Obtenidos

Los resultados que obtuvimos fueron los esperados, logrando un inventario sencillo pero muy útil, siendo capaz de buscar, ordenar, agregar y eliminar productos, con un menú numérico y fácil de entender

Conclusiones

La implementación del sistema de gestión de inventario permitió integrar y aplicar de manera práctica los algoritmos de búsqueda y ordenamiento más relevantes en programación, como la búsqueda lineal, búsqueda binaria, Quicksort y Mergesort. Se evidenció que la búsqueda binaria ofrece una mejora significativa en la eficiencia cuando el inventario está previamente ordenado, mientras que la búsqueda lineal, aunque más simple, resulta menos eficiente en listas extensas. En cuanto a los métodos de ordenamiento, Quicksort destacó por su rapidez promedio y bajo consumo de memoria, mientras que Mergesort aportó estabilidad y rendimiento predecible, aspectos cruciales en ciertos contextos. El desarrollo de un menú interactivo facilitó la interacción y comprensión del sistema, demostrando la importancia de combinar teoría y práctica en el aprendizaje de estructuras de datos y algoritmos. Este trabajo no solo permitió profundizar en conceptos fundamentales de programación, sino que también sentó las bases para futuras mejoras orientadas a optimizar y expandir la gestión de inventarios en entornos más complejos y reales.