Trabajo Practico Integrador

Tema “Algoritmos de Búsqueda y Ordenamiento en Python”

Alumnos: Fernando Agustín Palacios, Lucas Martín Monsón

Materia: Programación I

Profesor/a: Julieta Trapé

Fecha de Entrega: 09/06/2025

Índice

1. Introducción

En informática el manejo de datos es una operación fundamental para el correcto funcionamiento de los sistemas, programas, bases de datos o servidores. En este contexto las operaciones de búsqueda y ordenamiento son de vital importancia ya que nos permiten acceder, organizar y procesar la información de manera rápida y precisa. Estas tareas son llevadas a cabo por algoritmos diseñados para este fin. Un algoritmo es una serie ordenada de pasos que nos permiten obtener un resultado especifico. En el caso de los algoritmos de búsqueda y ordenamiento, su función principal es localizar datos y organizarlos según determinados criterios.
El propósito de este trabajo es comprender el funcionamiento de estos algoritmos, sus características, ventajas, desventajas y su impacto en el desarrollo de soluciones informáticas eficaces.

2. Marco Teórico

Algoritmos de ordenamiento

Los algoritmos de ordenamiento son un conjunto de instrucciones que toman un arreglo o lista como entrada y organizan los elementos en un orden particular.

Importancia de algoritmos de ordenamiento

Estos algoritmos son importantes porque pueden reducir la complejidad de un problema, optimizando los procesos posteriores. Tienen implicación directa en algoritmos de búsqueda, algoritmos de estructura de datos y en base de datos.

Criterios de elección

Al momento de elegir cual algoritmo utilizar debemos considerar que tan grande es nuestra estructura de datos a ordenar, cuanta memoria tenemos disponible o si la colección de datos necesita crecer en un futuro. La respuesta a estas preguntas nos va a permitir elegir mas apropiado para cada situación, considerando las limitaciones de nuestro sistema.

Algoritmos de ordenamiento más comunes

Ordenamiento de burbuja (selection sort)

Este es un algoritmo simple que recorre una lista y compara valores adyacentes repetidamente, intercambiándolos si no están en el orden correcto.

Este método es bastante lento comparado con otros algoritmos. Es adecuado para ordenar arreglos pequeños o cuando se necesita ejecutar en un sistema con recursos de memoria limitados.

Ordenamiento por inserción

La ordenación por inserción funciona comparando un elemento con los elementos anteriores. Si los elementos anteriores son mayores que este, mueve el elemento anterior a la siguiente posición. Aunque fácil de implementar, es ineficiente para listas grandes.

Ordenamiento por selección

Es uno de los algoritmos de ordenamiento más simple. Funciona de la siguiente manera: Encuentra el elemento más pequeño del arreglo y lo intercambia con el primer elemento, encuentra el segundo elemento más pequeño y lo intercambia con el segundo elemento y repite este proceso con las demás posiciones hasta ordenar todo el arreglo.

Ordenamiento por fusión (Merge Sort)

Este algoritmo divide el arreglo en dos mitades, ordena la mitad izquierda y la mitad derecha llamándose a si mismo de manera recursiva y por último fusiona las dos mitades ordenadas en un solo arreglo.

Ordenamiento Rápido (Quicksort)

El algoritmo de ordenamiento rápido selecciona un elemento de la lista y lo utiliza como pivote. A continuación, ordena el arreglo de manera que los elementos menores al pivote se ubican a la izquierda y los mayores a la derecha. Es de los algoritmos mas eficientes en la práctica

Algoritmos de Búsqueda

Búsqueda lineal

Este es un algoritmo simple de búsqueda que recorre cada elemento del conjunto de datos hasta encontrar el elemento deseado. Es sencillo de implementar, pero se vuelve lento en conjunto de datos grandes.

Búsqueda binaria

Es un algoritmo de búsqueda eficiente que funciona en conjuntos de datos ordenados. Divide el conjunto de datos en dos mitades y busca el elemento deseado en la mitad correspondiente. Repite este proceso hasta encontrar el elemento o determinar que no está en el conjunto de datos.

Búsqueda de interpolación

La búsqueda por interpolación es una versión optimizada de la búsqueda binaria que, luego de dividir el conjunto de datos en dos mitades, estima la posición del elemento buscado según el valor del mismo. Este algoritmo requiere que los datos estén ordenados para funcionar correctamente.

Búsqueda de hash

Este algoritmo utiliza como método una función hash que asigna a cada elemento una posición única en una tabla, lo que le permite acceder de manera rápida al elemento deseado. Este algoritmo es eficiente para conjuntos de datos grandes.

4. Caso Práctico

Desarrollo de algoritmos de búsqueda binaria y ordenamiento por burbuja.

4. Metodología Utilizada

El trabajo integrador se desarrolló de la siguiente manera:

- Investigación y recolección de información
- Desarrollo de algoritmos en Python
- Prueba del código y validación de resultados
- Elaboración de este informe

5. Resultados Obtenidos

El código desarrollado se ejecutó correctamente demostrando el funcionamiento de los algoritmos desarrollados.

6. Conclusión

A partir del desarrollo de este trabajo de investigación hemos logrado conocer y comprender tanto los algoritmos de búsqueda y ordenamiento como la importancia de los mismos.

Mediante ejemplos prácticos, se ha demostrado el desarrollo y funcionamiento en situaciones reales, pudiendo aplicar con éxito el contenido teórico desarrollado.

Existen diferentes algoritmos, con distintos niveles de complejidad y eficiencia y es importante determinar el óptimo para cada situación. Esto nos permite un desarrollo eficiente, teniendo en cuenta la cantidad de datos y recursos disponibles, dando como resultado programas rápidos, organizados y confiables.

7. Bibliografía

- *Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). MIT Press.*
- *Khan Academy. (s.f.). Búsqueda binaria. Khan Academy. https://es.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search*
- *freeCodeCamp. (s.f.). Sorting algorithms explained with examples in Python, Java and C. https://www.freecodecamp.org/news/sorting-algorithms-explained-with-examples-in-python-java-and-c/*
- *Bolat, E. (s.f.). Sorting algorithms with animations. Emre Bolat. Recuperado de https://emre.me/algorithms/sorting-algorithms/*