TRABAJO INTEGRADOR PROGRAMACIÓN I

CAPUTO, FERNANDO
COLMAN, NICOLAS

Introducción

El presente trabajo aborda el estudio de los algoritmos de búsqueda y ordenamiento, un tema fundamental dentro de la programación.

La correcta gestión de los datos y su manipulación es una habilidad esencial a la hora de desarrollar software, optimizar algoritmos y brindar funcionalidades diversas al usuario con respecto a la gestión de uno de los recursos más importantes: la información.

¿Por qué elegimos algoritmos de búsqueda y ordenamiento?

Uso constante	• Algoritmos que utilizaremos durante toda nuestra carrera
Versatilidad	• Aplicables tanto en programas simples como complejos
Necesidad práctica	• Ordenar listas de datos es una tarea recurrente
Búsqueda eficiente	• Encontrar elementos específicos de manera rápida
Optimización	• Mejorar el rendimiento de nuestras aplicaciones

Objetivo del trabajo integrador

Demostrar la importancia de usar algoritmos de ordenamiento antes de realizar búsquedas para optimizar la eficiencia y velocidad del proceso.

- Mostrar cómo el ordenamiento previo acelera las búsquedas
- Comparar diferentes algoritmos según el tamaño de datos
- Entender que cada algoritmo tiene su contexto de aplicación óptimo
- Aplicar conceptos teóricos a problemas reales

MARCO TEÓRICO

Marco Teórico

En el desarrollo de programas informáticos, uno de los desafíos más frecuentes es la necesidad de organizar datos y acceder a ellos de forma eficiente. Para abordar estas tareas, existen diversos algoritmos de búsqueda y ordenamiento.

Algoritmos de búsqueda

Un algoritmo de búsqueda permite localizar un valor dentro de una estructura, como una lista.

Algoritmos de búsqueda

En este trabajo consideraremos dos tipos:

- Búsqueda Lineal: recorre la lista desde el primer elemento hasta el último, comparando uno por uno con el valor buscado.
- Búsqueda Binaria: solo puede aplicarse sobre listas ordenadas. Divide repetidamente la lista por la mitad y compara el valor buscado con el elemento central. Si son distintos, reduce el rango de búsqueda a la mitad correspondiente.

Algoritmos de ordenamiento

Organiza una colección de elementos según un criterio determinado.

Algoritmos de búsqueda

Hay distintos tipos:

- **Bubble Sort (Ordenamiento burbuja):** compara pares de elementos adyacentes y los intercambia si están en el orden incorrecto.
- **Inserción:** construye una lista ordenada insertando cada elemento en la posición correcta.
- **Selección:** encuentra el mínimo elemento de la lista y lo coloca en su posición definitiva. Luego repite el proceso con el resto de la lista.
- **QuickSort:** divide la lista en sublistas menores y mayores que un "pivote", ordena recursivamente cada sublista, y luego combina los resultados.

Diccionarios en Python

Los diccionarios son una estructura de datos nativa del lenguaje Python que permite almacenar información en pares clave-valor. Son útiles para representar objetos con múltiples atributos.

Ejemplo de Diccionario en Python

```
Persona1 = {
    "nombre": "Nahuel",
    "sexo": "masculino",
    "edad": 32,
    "color_piel": "negro"
}

print(Persona1["nombre"]) # Salida: Nahuel
```