Algoritmos de Búsqueda y Ordenamiento

Alumnos:
Sofía González – sofiagonzalez@gmail.com
Marcos Benítez – marcosbenitez@gmail.com

Materia: Programación I

Profesor: Ing. Laura Fernández

Fecha de Entrega: 2 de mayo de 2025

Índice

Introducción

En la programación, los algoritmos de búsqueda y ordenamiento son fundamentales para la gestión eficiente de la información.
Comprender cómo se implementan y en qué situaciones se aplican es clave para el desarrollo de software efectivo.
Este trabajo explora estos algoritmos, incluyendo una explicación especial sobre la búsqueda binaria, por su relevancia
en estructuras de datos ordenadas.

Marco Teórico

Los algoritmos de ordenamiento permiten reorganizar una colección de elementos de acuerdo a un criterio específico
(mayor a menor, alfabéticamente, etc.). Entre los más comunes se encuentran:
- Bubble Sort: compara pares de elementos adyacentes y los intercambia si están en el orden incorrecto.
- Insertion Sort: construye una lista ordenada tomando elementos uno a uno e

insertándolos en la posición adecuada.
- Selection Sort: selecciona el elemento más pequeño del arreglo y lo coloca en su posición final.

Por otro lado, los algoritmos de búsqueda tienen como objetivo encontrar un valor dentro de una colección.
Destacamos:
- Búsqueda lineal: recorre secuencialmente todos los elementos hasta encontrar el deseado.
- Búsqueda binaria: eficiente en listas ordenadas, divide el espacio de búsqueda en mitades sucesivas.

La búsqueda binaria funciona de la siguiente manera:
1. Se compara el elemento central con el valor buscado.
2. Si son iguales, se retorna la posición.
3. Si el valor es menor, se repite la búsqueda en la mitad izquierda.
4. Si es mayor, en la mitad derecha.

Caso Práctico

Se desarrolló un programa en Python para ordenar una lista de números utilizando los algoritmos Bubble Sort
y para buscar un número específico mediante búsqueda binaria.

Código de ejemplo:

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]

def binary_search(arr, target):
    low = 0
    high = len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
```

```
    return -1

datos = [34, 7, 23, 32, 5, 62]
bubble_sort(datos)
print("Lista ordenada:", datos)
posicion = binary_search(datos, 23)
print("Elemento encontrado en la posición:", posicion)
```

Metodología Utilizada

La elaboración del trabajo se realizó en las siguientes etapas:
- Recolección de información teórica en documentación confiable.
- Implementación en Python de los algoritmos estudiados.
- Pruebas con diferentes conjuntos de datos.
- Registro de resultados y validación de funcionalidad.
- Elaboración de este informe y preparación de anexos.

Resultados Obtenidos

- El programa ordenó correctamente la lista de números ingresada.
- La búsqueda binaria localizó de forma eficiente el número especificado.
- Se comprendieron las diferencias en complejidad entre los algoritmos estudiados.
- Se valoró la importancia de tener los datos ordenados para aplicar búsqueda binaria.

Conclusiones

Los algoritmos de búsqueda y ordenamiento son pilares esenciales de la informática. Su estudio no sólo
contribuye al desarrollo técnico del estudiante, sino que permite entender cómo optimizar procesos de búsqueda
y organización de datos. La búsqueda binaria destacó por su eficiencia, aunque requiere una condición previa:
tener la lista previamente ordenada. Este trabajo reforzó la importancia de seleccionar el algoritmo adecuado
según el contexto y el tipo de datos a procesar.

Bibliografía

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms.
- Python Oficial: https://docs.python.org/3/library/
- Khan Academy: https://es.khanacademy.org/computing/computer-science/algorithms

Anexos

- Captura de pantalla del programa funcionando (insertar imagen).
- Repositorio en GitHub: https://github.com/grupo-python-busqueda-ordenamiento
- Video explicativo: (insertar enlace al video en YouTube)