Trabajo Integrador - Programación I

Título:

Exploración de Algoritmos de Búsqueda y Ordenamiento en Python

Integrantes:

Juan Galati – galatij545@gmail.com

Fecha de Entrega: 9/6/2025

Índice

1. Introducción

2. Marco Teórico

3. Desarrollo del Caso Práctico

4. Metodología de Trabajo

5. Resultados

6. Conclusiones

7. Fuentes Consultadas

8. Anexos

1. Introducción

En programación, el manejo eficiente de datos es esencial para el rendimiento de las aplicaciones. Los algoritmos de búsqueda y ordenamiento permiten encontrar información y organizarla, optimizando el uso de los recursos. Este trabajo busca analizar, implementar y reflexionar sobre dos de ellos usando el lenguaje Python.

2. Marco Teórico

Los algoritmos de ordenamiento reorganizan colecciones para facilitar tareas posteriores, como búsquedas, comparaciones o visualización.

Selection Sort es un algoritmo sencillo que recorre la lista para encontrar el elemento mínimo y lo coloca en su posición correcta. Su complejidad es O(n²), lo que lo hace menos eficiente para grandes volúmenes de datos, pero útil para aprendizaje.

Búsqueda Lineal verifica uno por uno los elementos hasta encontrar el deseado. Es ideal cuando los datos no están ordenados o el conjunto es pequeño.

| Algoritmo | Tipo | Complejidad | Uso recomendado |
| --- | --- | --- | --- |
| Selection Sort | Ordenamiento | O(n²) | Listas pequeñas, enseñanza básica |
| Búsqueda Lineal | Búsqueda | O(n) | Datos desordenados o pocos |

3. Desarrollo del Caso Práctico

A continuación, se presenta un código original en Python con una lista de datos aleatorios, aplicada al algoritmo de Selection Sort para ordenamiento, seguido de búsqueda lineal.

selection sort código:

def selection\_sort(arr):

n = len(arr)

for i in range(n):

min\_idx = i

for j in range(i + 1, n):

if arr[j] < arr[min\_idx]:

min\_idx = j

arr[i], arr[min\_idx] = arr[min\_idx], arr[i]

entrada = input("Ingresá los números separados por coma: ") lista = [int(x) for x in entrada.split(",")]

selection\_sort(lista)

print("Lista ordenada:", lista)

Búsqueda Lineal con ingreso de lista y valor a buscar:

def busqueda\_lineal(arr, objetivo):

for i in range(len(arr)):

if arr[i] == objetivo:

return i

return -1

entrada = input("Ingresá los números separados por coma: ") # ejemplo: 10,30,50,70,80

lista = [int(x) for x in entrada.split(",")]

valor = int(input("¿Qué número querés buscar? "))

resultado = busqueda\_lineal(lista, valor)

if resultado != -1:

print(f"El número {valor} está en la posición {resultado}.")

else:

print(f"El número {valor} no se encuentra en la lista.")

4. Metodología de Trabajo

1. Selección de algoritmos distintos al modelo para garantizar originalidad.

2. Programación en Python con pruebas.

3. Documentación de resultados.

4. Preparación de repositorio y video (pendientes).

5. Resultados

- El algoritmo Selection Sort ordenó correctamente los elementos.

- La búsqueda lineal encontró el valor indicado.

- Se comprobó que para pocos datos ambos algoritmos funcionan bien.

- El uso de funciones propias facilitó el control del proceso.

6. Conclusiones

La experiencia mostró que el algoritmo más adecuado depende del contexto. Selection Sort y búsqueda lineal funcionan bien para datos pequeños y ayudan a entender la lógica de procesamiento de datos.

7. Fuentes Consultadas

- Python Official Docs: https://docs.python.org/3/library/

- W3Schools Python: https://www.w3schools.com/python/

- Cormen, T. et al. (2009). Introduction to Algorithms.

8. Anexos

- Capturas del programa funcionando

- Repositorio GitHub: [colocar enlace aquí]

- Video explicativo: [colocar enlace aquí]