Datos Generales

Título del trabajo: Algoritmos de Búsqueda y Ordenamiento en Python aplicados a una lista de nombres
Alumno: Juan Manuel Gomez
Materia: Programación I
Profesor/a: Ariel Enferrel

LINK VIDEO: https://youtu.be/yxPQ5gDQAyc

Índice

---

1. Introducción

En este trabajo se abordan dos temas clave para cualquier programador principiante como lo son los algoritmos de búsqueda y de ordenamiento. Se eligió este tema porque son herramientas fundamentales para manejar información de forma eficiente. Aprender a ordenar y buscar datos correctamente no solo sirve para rendir bien la materia, sino que es la base para muchos programas más complejos. El objetivo del trabajo es entender cómo funcionan estos algoritmos y aplicarlos en Python, usando una lista de nombres.

---

2. Marco Teórico

Los algoritmos de ordenamiento son procedimientos que reacomodan elementos según el criterio que elijas. Uno de los más básicos es **Bubble Sort**, que va comparando de a pares y moviendo los elementos para que queden en orden.

Por otro lado, los algoritmos de búsqueda nos permiten encontrar un dato específico dentro de una lista. Acá usamos la **búsqueda lineal**, que va revisando uno por uno hasta encontrar lo que buscamos. Es lenta en listas largas, pero muy simple y no necesita que los datos estén ordenados.

Implementación en Python:

- bubble_sort(lista) → ordena la lista de manera alfabéticamente.
- busqueda_lineal(lista, nombre) → busca un nombre y devuelve a la posición.

---

3. Caso Práctico

Problema: En este caso práctico se plantea una situación muy frecuente en programación que es gestionar una lista de datos y realizar dos operaciones importantes en ella, ordenarla y buscar un elemento específico.

Imaginen que tenemos una agenda con numeros desordenados y necesitamos mostrarla de menor a mayor valor para que sea más fácil de leer. Despues queremos saber si un numero en particular está en esa lista y en qué posición. Para resolverlo implemente dos algoritmos simples en Python: **Bubble Sort** para ordenar y **búsqueda lineal** para localizar el numero buscado.

Codigo:

```
# Función que ordena una lista usando Bubble Sort def

bubble_sort(lista):

   n = len(lista)

for i in range(n):

      # En cada pasada se comparan los elementos de a pares

      for j in range(0, n - i - 1):

if lista[j] > lista[j + 1]:

            # Si están desordenados, se intercambian

            lista[j], lista[j + 1] = lista[j + 1], lista[j]


# Función que busca un número en la lista usando búsqueda lineal

def busqueda_lineal(lista, objetivo):     for i in range(len(lista)):

if lista[i] == objetivo:
```

```
        return i  # Devuelve la posición si lo encuentra
    return -1  # Devuelve -1 si no lo encuentra


# Lista original numeros = [12,

4, 7, 19, 3, 10]


# Se ordena la lista

bubble_sort(numeros) print("Lista

ordenada:", numeros)


# Se busca un número que quieras en la lista ordenada

posicion = busqueda_lineal(numeros, 10)


if posicion != -1:

    print("El número se encontró en la posición:", posicion) else:

    print("El número que buscas no está en la lista.")
```

Elegi **Bubble Sort** porque por mas que no sea el algoritmo más eficiente, es uno de los más fáciles de entender y de implementar para listas pequeñas, como la de este ejemplo. Para la búsqueda opte por **búsqueda lineal** ya que permite encontrar un dato sin necesidad de condiciones previas.

4. Metodología Utilizada

1. Se buscó información básica sobre los algoritmos en sitios confiables y libros introductorios.
2. Se diseñó el código primero en papel para entender la lógica paso a paso.
3. Se escribió el código en Python.
4. Se hicieron pruebas cambiando los numeros y buscando distintos valores.

5. Resultados Obtenidos

- La lista de números fue ordenada correctamente de menor a mayor.
- La búsqueda encontró los numeros buscados en la posición correcta.
- El código funcionó bien con otras listas de ejemplo también.
- No se presentaron errores al ejecutar el programa.

---

6. Conclusiones

Este trabajo me permitió entender bien cómo aplicar algoritmos básicos en Python. Tambien aprendi cómo ordenar y buscar datos, y quedó claro que elegir el algoritmo adecuado depende del tipo de datos y el problema a resolver. En el futuro podría probar con otros algoritmos más eficientes como la búsqueda binaria o QuickSort.

```python
# Función que ordena una lista usando Bubble Sort
def bubble_sort(lista):
    n = len(lista)
    for i in range(n):
        # En cada pasada se comparan los elementos de a pares
        for j in range(0, n - i - 1):
            if lista[j] > lista[j + 1]:
                # Si están desordenados, se intercambian
                lista[j], lista[j + 1] = lista[j + 1], lista[j]

# Función que busca un número en la lista usando búsqueda lineal
def busqueda_lineal(lista, objetivo):
    for i in range(len(lista)):
        if lista[i] == objetivo:
            return i  # Devuelve la posición si lo encuentra
    return -1  # Devuelve -1 si no lo encuentra

# Lista original
numeros = [12, 4, 7, 19, 3, 10]

# Se ordena la lista
bubble_sort(numeros)
print("Lista ordenada:", numeros)

# Se busca un número que quieras en la lista ordenada
posicion = busqueda_lineal(numeros, 10)

if posicion != -1:
    print("El número se encontró en la posición:", posicion)
else:
    print("El número que buscas no está en la lista.")
```

```
Lista ordenada: [3, 4, 7, 10, 12, 19]
El número se encontró en la posición: 3
```

---

7. Bibliografía

- Python Software Foundation. (2024). *Python 3 Documentation*. https://docs.python.org/3/
- Sweigart, A. (2019). *Automate the Boring Stuff with Python*. No Starch Press.
- Khan Academy. *Algoritmos de búsqueda y ordenamiento*. https://es.khanacademy.org/