Trabajo Practico integral programacion1

- Busqueda y Ordenamiento

- Alumnas: Ahumada,Alicia Yasmin
- Cabñas, Araceli, Jazmin

Algoritmos de Búsqueda y Ordenamiento en Python

Implementación de un Menú Interactivo para Restaurante

- **Introducción**
- Las operaciones de **búsqueda y ordenamiento** son esenciales para manejar y organizar datos eficientemente.
- Permiten localizar y reordenar información en estructuras como listas y diccionarios.
- Este proyecto implementa estos algoritmos en **Python**, aplicados a un menú de restaurante.
- Objetivo: fortalecer pensamiento lógico y aplicar conocimientos de programación en un caso práctico.

- **Marco Teórico - Generalidades**
- Los algoritmos de búsqueda y ordenamiento son fundamentales en sistemas informáticos.
- Su correcta selección mejora el **rendimiento** de aplicaciones, especialmente con **grandes volúmenes** de datos.
- Se analizan y aplican en el desarrollo de un sistema práctico y educativo en Python.

- **Algoritmos de Búsqueda - Definición**
- **Búsqueda:** proceso para localizar elementos en una estructura de datos.
- Depende de cómo están organizados los datos (listas, árboles, etc.).
- Existen varias técnicas con diferentes niveles de eficiencia.

- **Búsqueda Lineal**
- Recorre cada elemento hasta encontrar el deseado.
- **Ventajas:** simple y útil en listas no ordenadas.
- **Desventajas:** ineficiente con muchos datos.
- **Complejidad:** O(n)

- **Búsqueda Binaria**
- Requiere datos **ordenados**.
- Divide la lista sucesivamente y compara el valor central.
- **Ventajas:** muy eficiente en listas grandes.
- **Desventajas:** no funciona si la lista no está ordenada.
- **Complejidad:** O(log n)

- **Algoritmos de Ordenamiento - Definición**
- Reorganiza elementos según un criterio (precio, nombre, etc.).
- Mejora la eficiencia de operaciones como búsqueda.
- Evaluamos su **complejidad computacional**.

Complejidad Computacional

Notación	Significado
O(1)	Tiempo constante
O(n)	Tiempo lineal
O(n log n)	Tiempo logarítmico
$O(n^2)$	Tiempo cuadrático

- **Ordenamiento Burbuja (Bubble Sort)**
- Compara elementos adyacentes e intercambia si están mal ordenados.
- **Ventajas:** simple y didáctico.
- **Desventajas:** lento en listas grandes.
- **Complejidad:** $O(n^2)$
- Usado en el proyecto por su valor pedagógico.

- **Otros Algoritmos de Ordenamiento**
- **Insertion Sort:**
- Inserta cada elemento en la posición correcta.
- Bueno en listas pequeñas.
- **Selection Sort:**
- Encuentra el menor y lo mueve a su lugar final.
- Fácil de entender, pero lento.
- **Quick Sort:**
- Usa pivote para dividir y ordenar recursivamente.
- Muy eficiente en promedio.
- **Merge Sort:**
- Divide y fusiona listas ordenadas.
- Estable y predecible.

Comparación de Algoritmos

Algoritmo	Ventajas	Desventajas
Bubble Sort	Simple, didáctico	Ineficiente para listas grandes
Insertion Sort	Bueno con listas casi ordenadas	No escala bien
Selection Sort	Fácil de implementar	Siempre hace muchas comparaciones
Quick Sort	Rápido en la práctica	No estable, depende del pivote
Merge Sort	Rendimiento constante	Usa más memoria

- **¿Por qué elegir Bubble Sort?**
- Ideal para **aprendizaje**: muestra claramente el proceso.
- Muy fácil de **implementar**: poco código, sin estructuras complejas.
- Bueno para **listas pequeñas**: rendimiento aceptable.
- **Estable**: mantiene orden relativo de datos similares.
- Se puede **optimizar** para casos casi ordenados.

- Caso Practico

- **Conclusión**
- Comprender algoritmos clásicos fortalece la lógica y el análisis.
- La implementación práctica facilita la asimilación de conceptos.
- Elegir el algoritmo correcto depende del contexto y los datos.
- Python permite integrar teoría y práctica de forma accesible.