Laboratorio de Contenido: Insertion Sort en Python

Explicación básica

Insertion Sort toma un elemento y lo inserta en la posición correcta dentro de la parte ya ordenada de la lista. Es eficiente para listas pequeñas o casi ordenadas.
Complejidad promedio $O(n^2)$.

Objetivo

- Comprender el proceso de inserción progresiva.
- Implementar Insertion Sort.
- Analizar su eficiencia en listas casi ordenadas.

Caso práctico

Un administrador registra intentos de acceso fallidos:
intentos = [3, 1, 4, 2, 5]
Se requiere ordenarlos para análisis estadístico.

Problema

Implementar Insertion Sort mostrando el estado de la lista después de cada inserción.

Ejemplo paso a paso (Algoritmo básico)

Lista inicial:
3, 1, 4, 2, 5
Se toma 1 y se inserta antes de 3.
Resultado parcial:
1, 3, 4, 2, 5

Implementación en Python

```
# ----------------------------------------
# Método Insertion Sort en Python
# ----------------------------------------

def insertion_sort(lista):
    comparaciones = 0
    desplazamientos = 0

    for i in range(1, len(lista)):
        clave = lista[i]
        j = i - 1

        print(f"\nIteración {i}:")
```

```
        print(f"  Elemento a insertar: {clave}")
        print(f"  Parte ordenada: {lista[:i]}")

        # Mover elementos mayores que la clave
        while j >= 0 and lista[j] > clave:
            comparaciones += 1
            lista[j + 1] = lista[j]
            desplazamientos += 1
            print(f"  → Se desplaza {lista[j]} a la derecha")
            j -= 1

        # Cuenta la última comparación fallida (si aplica)
        if j >= 0:
            comparaciones += 1

        lista[j + 1] = clave
        print(f"  → Lista después de insertar: {lista}")

    print("\nLista ordenada:", lista)
    print("Total comparaciones:", comparaciones)
    print("Total desplazamientos:", desplazamientos)

    return lista

# ----------------------------------------
# Programa principal
# ----------------------------------------

if __name__ == "__main__":
    datos = [5, 8, 3, 7, 6]
    print("Lista original:", datos)
    insertion_sort(datos.copy())
```

Reflexión o Análisis

- ¿Qué ocurre si la lista está casi ordenada?
- ¿Por qué suele ser más eficiente que Bubble Sort en ese caso?
- ¿En qué tipo de aplicaciones reales podría utilizarse?