TRABAJO PRACTICO INTEGRADOR PROGRAMACION I

IMPLEMENTACION Y ANALISIS COMPARATIVO DE LOS ALGORITMOS DE BUBBLE SORT

¿QUÉ ES UN ALGORITMO DE ORDENAMIENTO?

- Es un conjunto de pasos para organizar una lista según un criterio (numérico, alfabético, etc.)

Facilita búsquedas, análisis y la visualizacion de datos.

¿CÓMO FUNCIONA BUBBLE SORT?

COMPARA PARES DE ELEMENTOS ADYACENTES.

INTERCAMBIA SI SE ENCUENTRAN DESORDENADOS.

REPITE EL PROCESO HASTA ORDENAR TODA LA LISTA

VENTAJAS Y DESVENTAJAS

VENTAJAS:

Fácil de entender y de implementar

Ideal para listas pequeñas

DESVENTAJAS

Ineficiente para listas extensas de datos

Se producen muchas comparaciones e intercambios

CASO PRACTICO:

```python
import time
# Ordenamiento por burbuja según prioridad (Bubble Sort)
def bubble_sort(productos):
    n = len(productos)
    for i in range(n - 1):
        intercambio_realizado = False
        for x in range(n - 1 - i):
            # Compara la prioridad (primer elemento de la sublista)
            if productos[x][0] > productos[x + 1][0]:
                productos[x], productos[x + 1] = productos[x + 1], productos[x]
                intercambio_realizado = True
        if not intercambio_realizado:
            break
    return productos

# Ordenamiento rápido según prioridad (Quick Sort)
def quick_sort(productos):
    if len(productos) <= 1:
        return productos
    else:
        pivote = productos[0]
        menores = [x for x in productos[1:] if x[0] <= pivote[0]]
        mayores = [x for x in productos[1:] if x[0] > pivote[0]]
        return quick_sort(menores) + [pivote] + quick_sort(mayores)

productos = [[5, "Manteca"], [6, "Fiambre"], [2, "Fideos"], [1, "Leche"], [3, "Arroz"], [4, "Pan"]]


ordenados_burbuja = bubble_sort(productos)
ordenados_rapido  = quick_sort(productos)

#Bubble Sort
print("Lista Bubble Sort segun prioridad:")
for producto in ordenados_burbuja:
    print(f"{producto[0]}: {producto[1]}")

# Quick Sort
print("\nLista Quick Sort segun prioridad:")
ordenados_rapido = quick_sort(productos.copy())
for prioridad, nombre in ordenados_rapido:
    print(f"{prioridad}: {nombre}")
```

```python
print("\n--- Prueba con lista de ejemplo original (N=6) ---")

# Comparación de tiempos
# Bubble Sort
productos_burbuja = productos.copy()
inicio_bubble = time.time()
bubble_sort(productos_burbuja)
fin_bubble = time.time()
tiempo_bubble = fin_bubble - inicio_bubble

# Quick Sort
productos_quick = productos.copy()
inicio_quick = time.time()
quick_sort(productos_quick)
fin_quick = time.time()
tiempo_quick = fin_quick - inicio_quick

# Resultados
print(f"Tiempo Bubble Sort: {tiempo_bubble:.6f} segundos")
print(f"Tiempo Quick Sort : {tiempo_quick:.6f} segundos")
```

GRAFICA COMPARATIVA

```python
import random
import time

# Generador de productos aleatorios
def generar_productos(cantidad):
    productos = []
    nombres_ejemplo = ["Leche", "Pan", "Fideos", "Arroz", "Manteca", "Fiambre", "Jugo", "Cereal", "Frutas", "Verduras"]
    for _ in range(cantidad):
        prioridad = random.randint(1, 1000)
        nombre = random.choice(nombres_ejemplo)
        productos.append([prioridad, nombre])
    return productos

# Ordenamiento por burbuja según prioridad (Bubble Sort)
def bubble_sort(lista):
    n = len(lista)
    for i in range(n - 1):
        intercambio_realizado = False
        for x in range(n - 1 - i):
            # Compara la prioridad (primer elemento de la sublista)
            if lista[x][0] > lista [x + 1][0]:
                lista[x], lista[x + 1] = lista[x + 1], lista[x]
                intercambio_realizado = True
        if not intercambio_realizado:
            break
    return lista

# Ordenamiento rápido según prioridad (Quick Sort)
def quick_sort(lista):
    if len(lista) <= 1:
        return lista
    else:
        pivote = lista[0]
        menores = [x for x in lista[1:] if x[0] <= pivote[0]]
        mayores = [x for x in lista[1:] if x[0] > pivote[0]]
        return quick_sort(menores) + [pivote] + quick_sort(mayores)

```

```python

# Generar lista de productos
cantidad_productos = 1000  # Puedes aumentar o reducir este número para comparar
productos = generar_productos(cantidad_productos)

# Comparación de tiempos
# Bubble Sort
productos_burbuja = productos.copy()
inicio_bubble = time.time()
bubble_sort(productos_burbuja)
fin_bubble = time.time()
tiempo_bubble = fin_bubble - inicio_bubble

# Quick Sort
productos_quick = productos.copy()
inicio_quick = time.time()
quick_sort(productos_quick)
fin_quick = time.time()
tiempo_quick = fin_quick - inicio_quick

# Resultados
print(f"Cantidad de productos: {cantidad_productos}")
print(f"Tiempo Bubble Sort: {tiempo_bubble:.6f} segundos")
print(f"Tiempo Quick Sort : {tiempo_quick:.6f} segundos")

```

COMPARACION CON QUICK SORT

- Utiliza la técnica "divide y vencerás": selecciona un "pivote" y organiza los elementos menores a un lado y los mayores al otro.

- Es mucho más rápido que el Bubble Sort en la mayoría de los casos, destacándose en ordenar grandes volumenes de datos.

CONCLUSION

- Bubble Sort es más útil para aprender algoritmos.
- No es recomendable para grandes volumenes de datos
- Quick Sort es más eficiente en la practica