Universidad Nacional Autónoma de México

Facultad de Ciencias

Análisis de Algoritmos

Práctica 3 - Ordenamientos

Jorge Angel Sánchez Sánchez

Fecha de entrega: 15 de Noviembre de 2024

Pasos para ejecutar el programa

1. Requisitos previos

- Asegurarse de tener instalado **Python 3** en el equipo.
  - Puede verificarse con el comando: `python3 --version`.
- Contar con un editor de texto (como Visual Studio Code) para visualizar o modificar el código.

2. Instrucciones de ejecución

1. Copiar el código al archivo `practica03.py`.
2. Abrir una terminal y navegar al directorio donde se guardó el archivo:

   ```
   cd /ruta/del/archivo
   ```

3. Ejecutar el programa con:

   ```
   python3 practica03.py
   ```

3. Salida esperada

El programa imprimirá en la terminal los resultados del número de operaciones realizadas por cada algoritmo según el tamaño del arreglo y el nivel de zigzag del patrón. Por ejemplo:

```
Resultados para tamaño = 1000, factor_zigzag = 1:
Merge Sort operaciones: 4932
Insertion Sort operaciones: 999
Inserción Local operaciones: 999
--------------------------------------------------------------------------------
Resultados para tamaño = 1000, factor_zigzag = 3:
Merge Sort operaciones: 5143
Insertion Sort operaciones: 1497
Inserción Local operaciones: 1497
```

Parámetros utilizados

Para evaluar los algoritmos, configuré los siguientes parámetros:

- **Tamaños del arreglo:** Probé arreglos de diferentes tamaños: `1000`, `2500`, `5000` y `10,000` elementos.
- **Factor zigzag (k):** Este parámetro controla el nivel de desorden en el arreglo inicial:
  - $k = 1$: Arreglo con pequeñas irregularidades.
  - $k = 3$: Arreglo con un mayor desorden.

Análisis de los resultados

Observaciones generales

- **Merge Sort** realizó más operaciones que **Insertion Sort** e **Inserción Local**, especialmente en arreglos más pequeños, debido a la sobrecarga asociada con su enfoque recursivo.
- **Insertion Sort** e **Inserción Local** mostraron el mismo desempeño, ya que sus implementaciones son prácticamente equivalentes.

Impacto del factor zigzag

- Cuando el factor zigzag es bajo ($k = 1$), los algoritmos requieren menos operaciones, ya que el arreglo está más ordenado inicialmente.
- Con un factor zigzag alto ($k = 3$), aumenta el número de operaciones para todos los algoritmos debido a la mayor cantidad de comparaciones necesarias para ordenar los datos.

Resultados representativos

Tamaño del arreglo	**Factor zigzag**	**Merge Sort**	**Insertion Sort**	**Inserción Local**
1000	1	4932	999	999
1000	3	5143	1497	1497
2500	1	13652	2499	2499
2500	3	14300	3747	3747
5000	1	29804	4999	4999
5000	3	31119	7498	7498

Cuadro 1: Resultados del número de operaciones según el tamaño y el factor zigzag.

Comportamiento según el tamaño del arreglo

- En arreglos pequeños, **Insertion Sort** e **Inserción Local** son más eficientes debido a su simplicidad.
- En arreglos grandes, **Merge Sort** se vuelve más efectivo porque su complejidad $O(n \log n)$ le permite manejar mejor el aumento del tamaño.