**UNIDAD 6**

**ALGORISMOS 2**

**DANCY DAYANA GOMEZ GALINDEZ**

**ANA SOFIA ANGEL GALAN**

**Código: 1060296429**

****

**Docente: CRISTIAN CAMILO IDALGO NIEVA**

**UNIVERSIDAD REMINGTON DE CALI**

**Facultad de Ingeniería**

**Ingeniería En Sistema**

**Cali, Agosto 2025**

**Documentación del Proyecto**

**• README: Los estudiantes deberán elaborar un archivo README.md que detalle:**

• **Análisis del problema: La necesidad de automatización y la importancia del ordenamiento estable.**

En una empresa de mensajería, cada pedido tiene un identificador único (id), una prioridad (alta, media, baja) y un orden de llegada (ordenOriginal). Para mejorar la eficiencia operativa, es necesario automatizar el proceso de ordenamiento de pedidos.

La automatización permite:

* Procesar grandes volúmenes de pedidos rápidamente.
* Priorizar entregas según criterios definidos.
* Mantener la estabilidad del ordenamiento, es decir, conservar el orden original entre elementos con claves iguales.

**Importancia del ordenamiento estable:**  
Cuando se ordenan los pedidos por múltiples criterios (por ejemplo, primero por prioridad y luego por ID), es fundamental que el algoritmo preserve el orden relativo de los elementos con claves iguales. Esto garantiza que los pedidos con la misma prioridad mantengan su orden de llegada.

**• Justificación de los algoritmos: Por qué se eligieron Bubble Sort, Selection Sort e Insertion Sort para el análisis**

|  |  |
| --- | --- |
| Insertion Sort | Estable y eficiente en listas pequeñas o parcialmente ordenadas. Muy útil en escenarios reales. |
| Bubble Sort | Algoritmo estable y fácil de implementar. Ideal para visualizar el proceso paso a paso. |

**• Resultados y Conclusiones: Una tabla comparativa de los tiempos de ejecución y la notación Big O de cada algoritmo, seguida de una conclusión sobre el algoritmo más adecuado para la empresa de mensajería**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tamaño de datos** | **Bubble Sort** | **Selection Sort** | **Insertion Sort** | **Complejidad Big O** |
| **100** | **4.48 ms** | **5.35 ms** | **1.78 ms** | **O(n²)** |
| **1.000** | **15.18 ms** | **40.95 ms** | **14.72 ms** | **O(n²)** |
| **10.000** | **1698.20 ms** | **668.06 ms** | **287.33 ms** | **O(n²)** |

Insertion Sort: Más eficiente en listas parcialmente ordenadas. Su rendimiento mejora en casos casi ordenados.