# Observaciones - Laboratorio 5

Curso: Estructuras de Datos (ISIS1225)

Integrantes: David, Gabriel y Santiago

Grupo 3

Fecha: 23/09/2025

## Introducción

En este laboratorio se evaluó el desempeño de diferentes algoritmos de ordenamiento implementados sobre dos estructuras de datos: ArrayList y SingleLinkedList. El objetivo principal fue comparar los tiempos de ejecución de cada algoritmo en ambas estructuras, relacionando los resultados empíricos con la complejidad teórica esperada. Este documento recoge los resultados obtenidos, las gráficas generadas en el archivo Excel, y las respuestas a las preguntas de análisis de las partes 1 y 2.

## Parte 1 - Análisis de Algoritmos de Ordenamiento

A continuación, se presentan las respuestas a las preguntas de análisis de la Parte 1:

1. ¿Qué diferencias se observan en los tiempos de ejecución de los algoritmos iterativos (Insertion, Selection, Shell) con respecto a los recursivos (Merge, Quick)?

Los algoritmos iterativos como Insertion y Selection muestran tiempos de ejecución significativamente más altos en listas grandes debido a su complejidad $O(n^2)$. ShellSort, aunque iterativo, logra un mejor desempeño cercano a O(n log n). Por otro lado, los algoritmos recursivos (Merge y Quick) se comportan mejor en entradas grandes gracias a su complejidad promedio O(n log n), lo cual se evidencia en los tiempos de ejecución observados.

2. ¿Cómo influye el tipo de estructura de datos (ArrayList vs SingleLinkedList) en el desempeño de los algoritmos?

En ArrayList, los accesos a posiciones específicas son más rápidos (O(1)), lo que favorece algoritmos que realizan muchas comparaciones. En cambio, en SingleLinkedList, el acceso es O(n), lo cual penaliza algoritmos como Insertion o Selection que requieren múltiples recorridos. Sin embargo, algoritmos recursivos como MergeSort muestran un comportamiento más equilibrado entre ambas estructuras.

3. ¿Se cumple la complejidad teórica esperada en los resultados experimentales?

Sí, los resultados confirman las complejidades teóricas: Selection e Insertion crecen cuadráticamente, ShellSort mejora notablemente, y Merge/Quick presentan curvas que se aproximan a n log n. Las pequeñas desviaciones se deben a factores de implementación y características del hardware, o temas del computador de prueba.

## Parte 2 - Comparación de los Mejores Algoritmos

1. ¿Cuál es el mejor algoritmo iterativo y por qué?

El mejor algoritmo iterativo identificado fue ShellSort. Aunque su complejidad exacta depende de la secuencia de incrementos, en la práctica su desempeño es cercano a O(n log

n), muy superior a Insertion o Selection. En las pruebas, ShellSort se mostró consistente y eficiente para listas grandes.

2. ¿Cuál es el mejor algoritmo recursivo y por qué?

El mejor algoritmo recursivo fue QuickSort. En promedio, QuickSort logra O(n log n) y fue el más rápido en la mayoría de los casos. Sin embargo, debe tenerse en cuenta que su peor caso es $O(n^2)$, por lo que MergeSort puede ser más estable en escenarios específicos.

3. Comparación final entre mejores algoritmos (iterativo vs recursivo)

Comparando ShellSort y QuickSort, QuickSort se posiciona como el más eficiente en entradas grandes, mientras que ShellSort es una excelente alternativa en escenarios donde se prefieren algoritmos iterativos. Ambos superan ampliamente a Insertion y Selection, y confirman la importancia de escoger algoritmos con mejor complejidad teórica.

## Conclusiones

Este laboratorio permitió evidenciar cómo la complejidad teórica de los algoritmos se refleja en la práctica. La estructura de datos influye en el desempeño, especialmente en algoritmos iterativos con listas enlazadas. Finalmente, se comprobó que los algoritmos con complejidad O(n log n), como QuickSort y ShellSort, son los más adecuados para manejar volúmenes grandes de datos.