OBSERVACIONES DE LA PRÁCTICA

Ashlee Yin 202421132

**Ambientes de pruebas**

Máquina 1

Máquina 2

Máquina 3

Procesadores

Procesador 1,6 GHz Dual-Core Intel Core i5

Memoria RAM (GB)

Memoria 8 GB 1600 MHz DDR3

Sistema Operativo

MacOS

Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

**Preguntas de análisis**

**• ¿El comportamiento de los algoritmos es acorde a lo enunciado teóricamente?**

Sí, en general, los resultados concuerdan con la complejidad teórica de los algoritmos:

**⁃ Quick Sort** es más rápido en la mayoría de los casos, especialmente con **ArrayList**.

**⁃ Insertion Sort y Selection Sort** son significativamente más lentos, especialmente para listas grandes.

**⁃ Merge Sort y Shell Sort** muestran un desempeño más estable en comparación con los métodos cuadráticos.

**• ¿Cuál Estructura de Datos es mejor utilizar si solo se tiene en cuenta los tiempos de ejecución de los algoritmos?**

**⁃ Array List es superior en todos los casos.**

⁃ Los tiempos de ejecución con **Linked List** son considerablemente mayores o incluso llegan al punto de no completar la ejecución en algunos algoritmos.

⁃ Esto se debe a que las listas enlazadas tienen un mayor costo en acceso a elementos dispersos en memoria y en ciertas operaciones de ordenamiento que dependen de acceso aleatorio.

**• ¿Cómo afecta el tamaño de los datos a la eficiencia de cada algoritmo en las diferentes estructuras de datos?**

A medida que aumenta el tamaño de la muestra, **los algoritmos con peor complejidad (O(n²)) como Insertion y Selection Sort tardan mucho más**.

**Quick Sort y Merge Sort escalan mejor**, aunque Quick Sort tiende a ser más rápido en la práctica debido a la optimización de particionamiento en caché.

**Las listas enlazadas sufren más con listas grandes**, ya que la falta de acceso aleatorio ralentiza las comparaciones y movimientos de datos.

**• ¿Cuál consideran que fue el mejor algoritmo iterativo y por qué?**

**Quick Sort fue el mejor en Array List**, ya que mostró tiempos considerablemente menores en comparación con los demás.

**Shell Sort también tuvo un buen desempeño** y mostró escalabilidad en listas más grandes.

**Selection e Insertion Sort fueron los peores**, con tiempos que se disparan para tamaños grandes.

**• Si quisieras ordenar una lista muy grande de datos, ¿qué algoritmo escogerías? ¿Por qué?**

**⁃ Quick Sort** es la mejor opción para listas grandes debido a su complejidad **O(n log n)** y su buen rendimiento en la práctica.

**• Si quisieras ordenar una lista parcialmente ordenada, ¿cuál sería el mejor algoritmo?**

**⁃ Insertion Sort sería una buena opción en este caso**, ya que si la lista está casi ordenada, su rendimiento puede acercarse a **O(n)** en lugar de **O(n²)**.