# OBSERVACIONES DE LA PRACTICA

Hernán Felipe Buitrago Cod 201512807
Daniel Esteban Aguilera Figueroa Cod 202010592

| | Máquina 1 | Máquina 2 |
|---|---|---|
| **Procesadores** | Intel i5-7400 @3.00GHz | Intel Core i5 Dual-Core @2.60GHz |
| **Memoria RAM (GB)** | 8.0 GB | 8.0 GB |
| **Sistema Operativo** | Windows 10 (64-bits) | MacOS Big Sur |

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

## Maquina 1

### Resultados

| Tamaño de la muestra (ARRAYLIST) | Insertion Sort [ms] | Selection Sort [ms] | Shell Sort [ms] | Quick Sort [ms] | Merge Sort [ms] |
|---|---|---|---|---|---|
| 1000 | 812,50 | 937,50 | 46,88 | 31,25 | 31,25 |
| 2000 | 3375,00 | 3734,38 | 93,75 | 78,125 | 62,5 |
| 4000 | 13140,63 | 15234,38 | 234,38 | 156,25 | 156,25 |
| 8000 | 55140,63 | 61859,38 | 562,50 | 328,125 | 328,125 |
| 16000 | 219593,75 | 254937,50 | 1312,50 | 718,75 | 687,5 |
| 32000 | 918296,88 | 1041187,50 | 2921,88 | 1531,25 | 1515,625 |
| 64000 | 3047984,38 | 4187687,50 | 7234,38 | 3406,25 | 3234,375 |
| 128000 | 12926015,63 | Tiempo de espera exagerado | 17640,63 | 7468,75 | 6921,875 |
| 256000 | Tiempo de espera exagerado | Tiempo de espera exagerado | 43421,88 | 15953,125 | 14796,875 |
| 512000 | | | | | |

*Tabla 2. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación arreglo.*

| Tamaño de la muestra (LINKED_LIST) | Insertion Sort [ms] | Selection Sort [ms] | Shell Sort [ms] | Quick Sort [ms] | Merge Sort [ms] |
|---|---|---|---|---|---|
| 1000 | 51156,25 | 46843,75 | 2593,75 | 2015,625 | 234,375 |
| 2000 | 424796,88 | 382281,25 | 10718,75 | 9390,625 | 953,125 |
| 4000 | 3395343,75 | 429125,00 | 52296,88 | 37203,125 | 3765,625 |
| 8000 | Tiempo de espera exagerado | 3088515,63 | 266125,00 | 163453,125 | 15468,75 |
| 16000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | 705562,5 | 60734,375 |
| 32000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | 246281,25 |
| 64000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado |
| 128000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado |

| | | | | | |
|---|---|---|---|---|---|
| 256000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado |
| 512000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado |

*Tabla 3. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación lista enlazada.*

| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED_LIST) |
|---|---|---|
| **Merge sort** | 1 | 1 |
| **Quick sort** | 2 | 2 |

*Tabla 4. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.*

### Graficas

- Cinco gráficas generadas por los resultados de las pruebas de rendimiento en la **Maquina 1.**
  - Comparación de rendimiento ARRAYLIST.
  - Comparación de rendimiento LINKED_LIST.
  - Comparación de rendimiento para Insertion Sort.
  - Comparación de rendimiento para Selection Sort.
  - Comparación de rendimiento para Shell Sort.
  - Comparación de rendimiento para MergeSort.
  - Comparación de rendimiento para QuickSort.

## Maquina 2

### Resultados

| Tamaño de la muestra (ARRAYLIST) | Insertion Sort [ms] | Selection Sort [ms] | Shell Sort [ms] | Quick Sort [ms] | Merge Sort [ms] |
|---|---|---|---|---|---|
| 1000 | 919.93 | 1002.24 | 50.9 | 37.85 | 40.14 |
| 2000 | 4022.17 | 4711.29 | 126.8 | 75.93 | 81.48 |
| 4000 | 16223.40 | 16806.2 | 268.77 | 181.36 | 167.55 |
| 8000 | 63898.96 | 74768.9 | 646.81 | 341.69 | 363.32 |
| 16000 | 265147.7 | 298399.74 | 1591.8 | 727.16 | 796.66 |
| 32000 | 1087265.8 | 1156731.5 | 3386.85 | 1833.60 | 1713.0 |
| 64000 | Tiempo de espera exagerado | Tiempo de espera exagerado | 7723.45 | 3598.64 | 3596.33 |
| 128000 | Tiempo de espera exagerado | Tiempo de espera exagerado | 20395.3 | 7635.46 | 7646.150 |
| 256000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado |
| 512000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado |

*Tabla 5. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación arreglo.*

| Tamaño de la muestra (LINKED_LIST) | Insertion Sort [ms] | Selection Sort [ms] | Shell Sort [ms] | Quick Sort [ms] | Merge Sort [ms] |
|---|---|---|---|---|---|
| 1000 | 58082.5 | 52245.8 | 2650.8 | 2074 | 281.34 |
| 2000 | 510187.04 | 465607.7 | 15023.3 | 9018.3 | 1131.96 |
| 4000 | Tiempo de espera exagerado | Tiempo de espera exagerado | 71265.13 | 56455.8 | 4426.50 |
| 8000 | Tiempo de espera exagerado | Tiempo de espera exagerado | 356713.4 | 199467.8 | 18236 |

| | | | | | |
|---|---|---|---|---|---|
| 16000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | 931862.6 | 80909.40 |
| 32000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | 330618.3 |
| 64000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | 1313836.9 |
| 128000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado |
| 256000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado |
| 512000 | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado | Tiempo de espera exagerado |

*Tabla 6. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación lista enlazada.*

| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED_LIST) |
|---|---|---|
| **Merge sort** | 2 | 1 |
| **Quick sort** | 1 | 2 |

*Tabla 7. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.*

## Graficas

- Cinco gráficas generadas por los resultados de las pruebas de rendimiento en la **Maquina 2.**
  - Comparación de rendimiento ARRAYLIST.
  - Comparación de rendimiento LINKED_LIST.
  - Comparación de rendimiento para Insertion Sort.
  - Comparación de rendimiento para Selection Sort.
  - Comparación de rendimiento para Shell Sort.
  - Comparación de rendimiento para MergeSort.
  - Comparación de rendimiento para QuickSort.

# Preguntas de análisis

1) ¿El comportamiento de los algoritmos es acorde a lo enunciado teóricamente?

- Teniendo en cuenta las graficas de ambos algoritmos podemos observar que su comportamiento concuerda con lo descrito en el material del curso. Por un lado, en el caso del Merge Sort podemos evidenciar en la grafica su comportamiento como un O(nlogn). Por otro lado, en la grafica podemos observar una curva mas leve la cual representa O(logn).

2) ¿Existe alguna diferencia entre los resultados obtenidos al ejecutar las pruebas en diferentes máquinas?

- En este laboratorio podemos observar una pequeña diferencia comparando con el anterior. En el anterior laboratorio pudimos observar que para ambas maquinas los algoritmos óptimos eran los mismos. Sin embargo, en este caso para la maquina 2 en el arreglo ARRAY_LIST el Merge Sort fue mucho mas eficiente, y en la maquina uno esto fue al revés, el Quicksort presentó mejor rendimiento. Por otro lado, la diferencia mas notable fue el tiempo entre ambos computadores.

3) De existir diferencias, ¿A qué creen ustedes que se deben dichas diferencias?

- Aquellas diferencias en tiempos y en la mayor eficiencia de un tipo de algoritmo de ordenamiento sobre otro se deben a que, por un lado, algunos procesos de la maquina pudieron

disminuir la eficiencia del programa. Por otro lado, el estado de los componentes también juega un papel muy importante a la hora de ejecutar el programa de forma eficiente.

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

- En el caso de la maquina 1, el Merge Sort fue el mas eficiente para ambos tipos de arreglo. En el caso de la maquina 2, el Quicksort fue mucho mas eficiente en el ARRAY_LIST y el Merge Sort en el SINGLE_LINKED.

5) Para el caso analizado de ordenamiento de los videos, teniendo en cuenta los resultados de tiempo reportados por todos los algoritmos de ordenamiento estudiados (iterativos y recursivos), proponga un ranking de los algoritmos de ordenamiento (de mayor eficiencia a menor eficiencia en tiempo) para ordenar la mayor cantidad de videos.

- El ranking que propondríamos para definir mayor eficiencia y menor eficiencia en tiempo es:

1. Merge Sort
2. Quick Sort
3. Shell Sort
4. Selection Sort
5. Insertion Sort