**OBSERVACIONES DE LA PRACTICA**

*Jose Luis Tavera Ruiz 201821999*

*Juan Diego Yepes 202022391*

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | Apple M1, 8 núcleos | Inter Core i5 1,3 Ghz 2 núcleos |
| Memoria RAM (GB) | 16GB | 4 GB |
| Sistema Operativo | MacOS Big Sur Version 11.2.1 | MacOS Catalina Version 10.15.7 |

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 | 482,97 | 552,92 | 58,35 | 39,75 | 45,75 |
| 2000 | 1909,56 | 2108,04 | 91,66 | 78,33 | 76,21 |
| 4000 | 7457,27 | 8335,29 | 161,93 | 133,84 | 125,29 |
| 8000 | 30092,41 | 33205,51 | 341,75 | 213,17 | 223,07 |
| 16000 | 116987,04 | 134928,78 | 760,47 | 408,99 | 435,87 |
| 32000 | 509467,51 | 582390,13 | 1682,26 | 887,87 | 917,61 |

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 | 44474,67 | 37912,94 | 2281,54 | 1475,36 | 203,39 |
| 2000 | 371011,09 | 315280,58 | 9398,07 | 7975,06 | 782,22 |
| 4000 | 2544793,50 | 2131290,10 | 46541,70 | 34505,94 | 3133,75 |

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 | Más eficiente | Menos eficiente |
| Quick sort | Más eficiente | Menos eficiente |

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 para Insertion Sort.
  + Comparación de rendimiento para MergeSort.

# **Maquina 2**

# **Maquina 2**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 1037,56 | 1156,03 | 70,11 | 44,16 | 49,66 |
| 2000 | 4227,37 | 4668,74 | 136,26 | 102,59 | 96,59 |
| 4000 | 16679,56 | 19015,44 | 304,33 | 221,55 | 197,93 |
| 8000 | 74030,62 | 79680,59 | 743,50 | 420,51 | 441,98 |
| 16000 | 274967,61 | 328389,68 | 1762,50 | 842,67 | 897,05 |
| 32000 | 1154768,01 | 1322892,25 | 3716,17 | 1896,45 | 1972,53 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **T**amaño de la muestra (LINKED\_LIST) | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 70278,60 | 64469,86 | 3784,99 | 2447,03 | 345,04 |
| 2000 | 584936,57 | 528002,93 | 16012,09 | 13362,83 | 1368,25 |
| 4000 | 3264895,99 | 2827192,01 | 79173,29 | 58171,52 | 5342,23 |

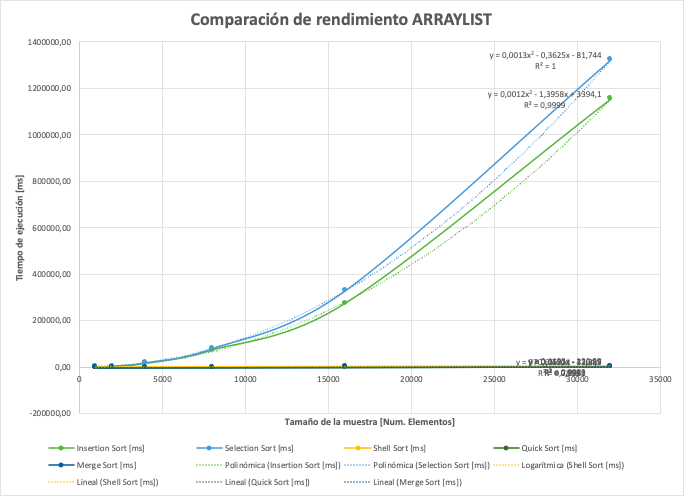
Tabla 3. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación lista enlazada

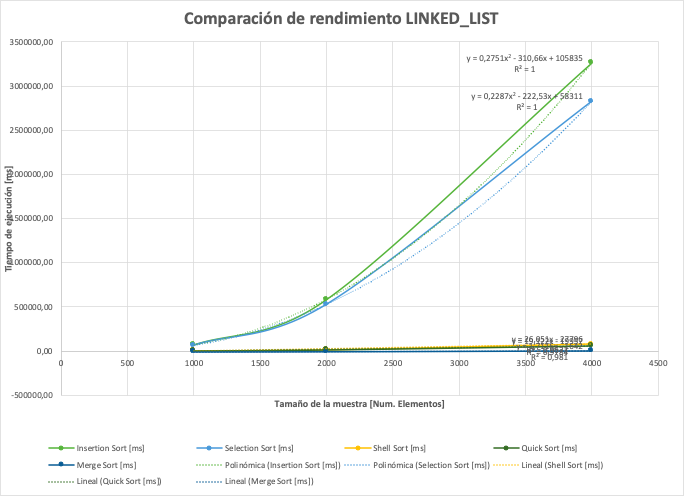
|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| Merge sort | Más eficiente | Menos eficiente |
| Quick sort | Más eficiente | Menos eficiente |

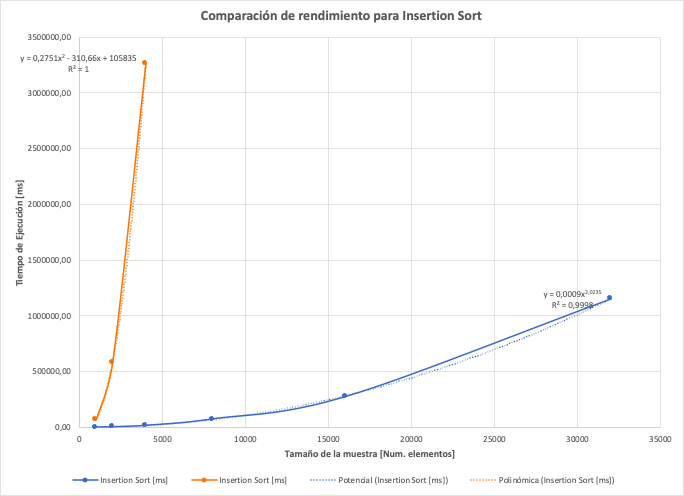
Tabla 4. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.

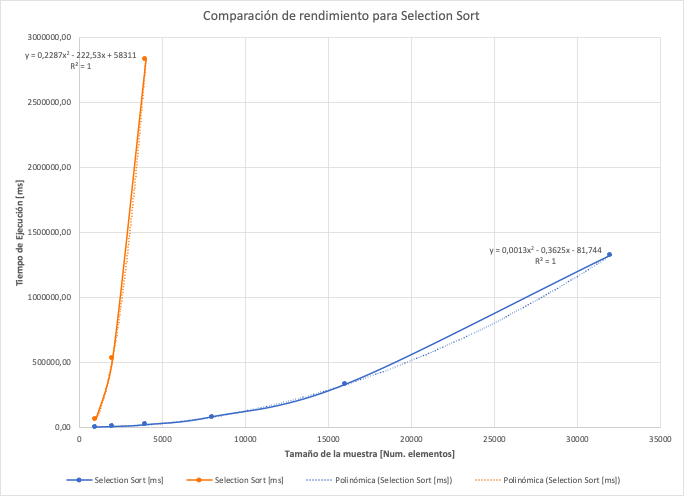
## **Graficas**

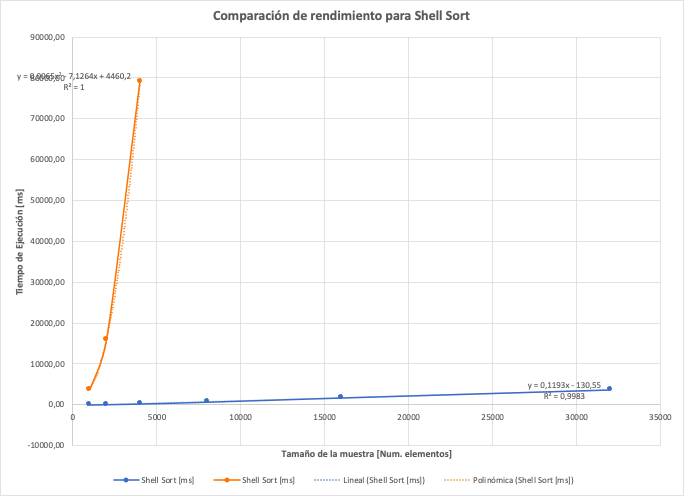
* Cinco gráficas generadas por los resultados de las pruebas de rendimiento en la **Maquina 2.**

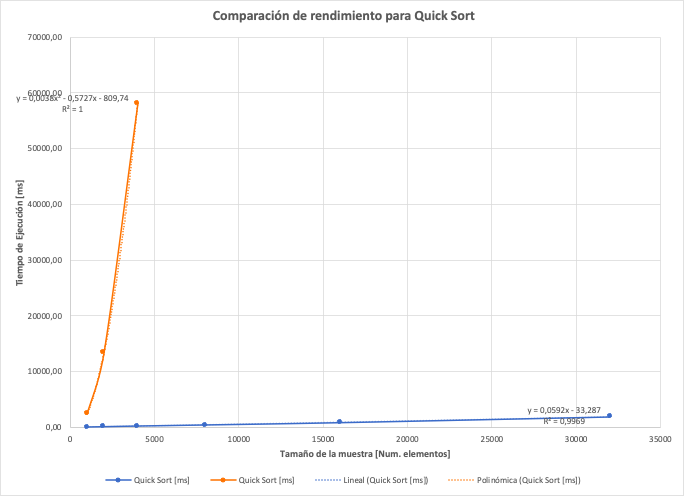


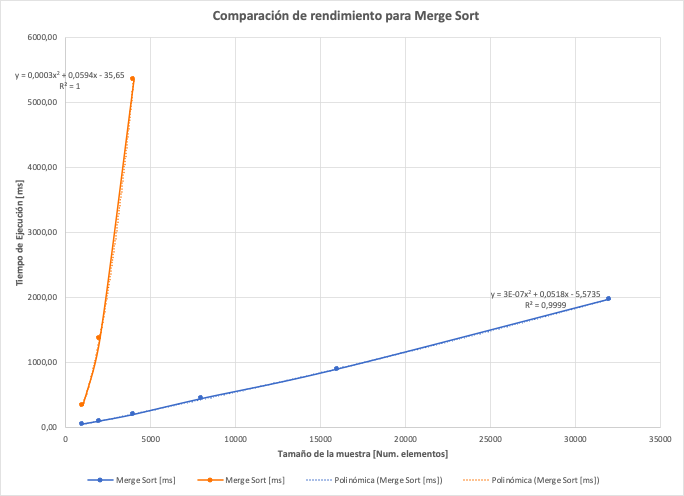












# **Preguntas de análisis**

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

En efecto, ya que según la teoría que manejamos en clase, para las estructuras de dats tipo SINGLE\_LINKED, encontrar un elemento, comparar un elemento y demás operaciones que requieren los ordenamientos están determinadas por el orden de crecimiento O(n). Por el contrario, en el ARRAY\_LIST, estas operaciones solo toman 0(1). Esto se explica por la manera en como funcionan estos tipos abstractos de datos, ya que aunque ambos están ordenados secuencialmente, en el ARRAY\_LIST se encuentran los elementos seguidos en memoria mientras que en SINGLE\_LINKED se encuentran unidos por nodos. Finalmente, contrastando los algoritmos iterativos y los recursivos, el hecho de que los ordenamientos recursivos sean mucho más rápidos se debe a que estos se llaman a sí mismos, bajo la premisa de ‘dividir y conquistar’. Esto es efectivo porque muestra ordenamientos de carácter O(log(n)) en sus mejores casos. Mientras que los iterativos son mucho más lentos porque se ejecutan mediante ciclos y en sus mejores casos (insertion y selection) su orden es de O(nˆ2) y O(n). El caso de shell es distinto, mostrando resultados similares a los de los algoritmos recursivos.

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

Al ejecutar las pruebas en ambos computadores encontramos una brecha en todos los tiempos. Sin embargo, esto se mantuvo “relativamente” constante en cada uno de los algoritmos y estructuras de datos.

1. De existir diferencias, ¿A qué creen ustedes que se deben dichas diferencias?

Estas diferencias se deben a los aspectos técnicos de cada una de las maquinas. En primera instancia, la diferencia de 12GB de ram afecta el espacio de la memoria caché que tiene el dispositivo para el almacenamiento de datos y ejecucuión de tareas. A menor espacio aumentará el tiempo de ejecución. Esto debido a que la Memoria de Acceso Aleatorio (RAM) es la encargada de las instrucciones ejecutadas por el CPU y otras unidades del computador, además de contener los datos que manipulan los distintos programas. Por lo tanto, también influye la diferencia de procesamiento del CPU y su número de nucleos que limitan o aumentan según el caso la memoria disponible de caché y su velocidad de acceso para la ejecucuión de tareas.

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

Teniendo en cuenta los tiempos de ejecución para ambos casos, encontramos que es más adecuado utilizar para todos los casos la estructura de datos tipo ARRAY, puesto que esta demostró ser mucho más eficiente en términos de tiempo para los algoritmos analizados (insertion, selection, shell, quick y merge).

1. 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 quedaría de la siguiente manera: En primer lugar, según las pruebas que hicimos, el algoritmo más eficaz es quicksort, ya que en ambas máquinas mostró ser el más eficiente. Después le sigue mergesort, posteriormente shellsort, después hay un empate entre selectionsort e insertionsort, porque mostraron resultados muy similares y uno fue mejor que otro dependiendo del tipo de Estructura de Datos.