OBSERVACIONES DE LA PRACTICA

María Paula Alméciga Moreno 202023369

Andrés Felipe Vargas Cuadros 202013817

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | AMD A10-8700P Radeon R6 | AMD Ryzen 5 3500U |
| Memoria RAM (GB) | 16 GB | 8 GB |
| Sistema Operativo | Windows 8.1 Pro | Windows 10 |

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 | 2000.00 | 1942.71 | 109375.00 | 78.13 | 93.75 |
| 2000 | 8609375.00 | 8593.75 | 296875.00 | 182.29 | 197.91 |
| 4000 | 34468.75 | 35437.50 | 609375.00 | 354.17 | 427.08 |
| 8000 | 133156.25 | 137406.25 | 1312.50 | 895.83 | 880.21 |
| 16000 | 535484.38 | 555296.88 | 3125.00 | 1895.04 | 1880.21 |
| 32000 | Excede tiempo | Excede tiempo | 7750.00 | 3817.71 | 4000.00 |
| 64000 | Excede tiempo | Excede tiempo | 17953.13 | 7958.33 | 8536.46 |
| 128000 | Excede tiempo | Excede tiempo | 41812.50 | 15473.96 | 18000.00 |
| 256000 | Excede tiempo | Excede tiempo | Excede tiempo | 36984.38 | 38307.29 |
| 512000 | Excede tiempo | Excede tiempo | Excede tiempo | 60619.79 | 57057.29 |

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 | 144296.88 | 131500.00 | 7265.63 | 5057.29 | 796.88 |
| 2000 | 1169312.50 | 1007343.75 | 35937.50 | 23963.54 | 3390.33 |
| 4000 | Excede tiempo | Excede tiempo | 169437.50 | 105401.03 | 12046.88 |
| 8000 | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo | 48869.79 |
| 16000 | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo | 198343.75 |
| 32000 | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo |
| 64000 | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo |
| 128000 | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo |
| 256000 | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo |
| 512000 | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo |

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

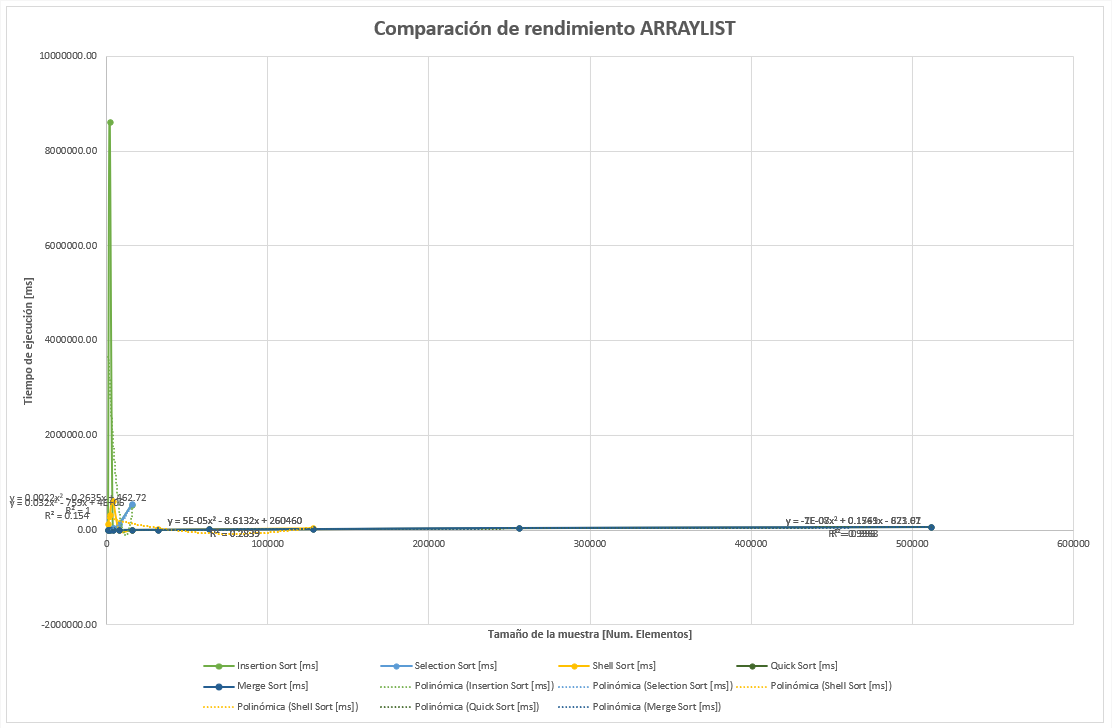
|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| Merge sort |  | x |
| Quick sort | x |  |

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

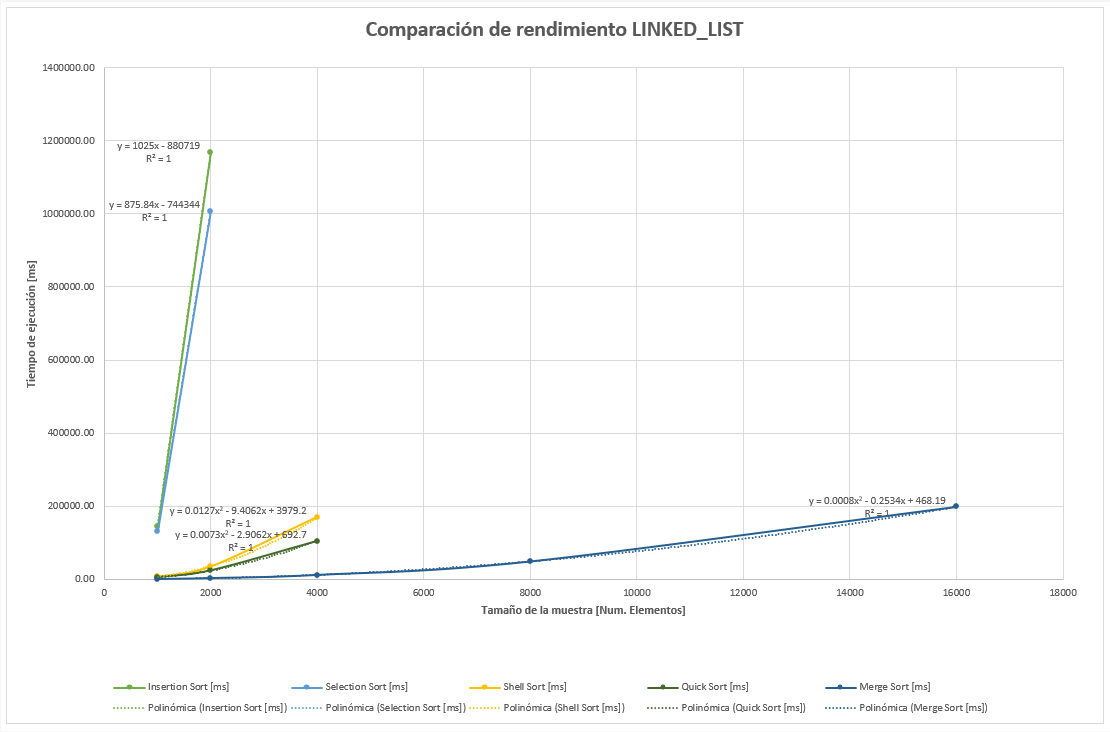
Se ha marcado en la anterior tabla, para cada tipo de estructura de datos, qué tipo de ordenamiento recursivo fue más eficiente en las pruebas. Según los datos y los tiempos exhibidos, se encontró que, para el arreglo, fue más eficiente el quick sort, y para la lista enlazada, el merge sort. Además, se ha determinado que el arreglo (array\_list) es el tipo de estructura de datos que exhibió tiempos de ejecución menores, resultando ser también la mejor opción entre las dos presentadas.

## **Graficas**

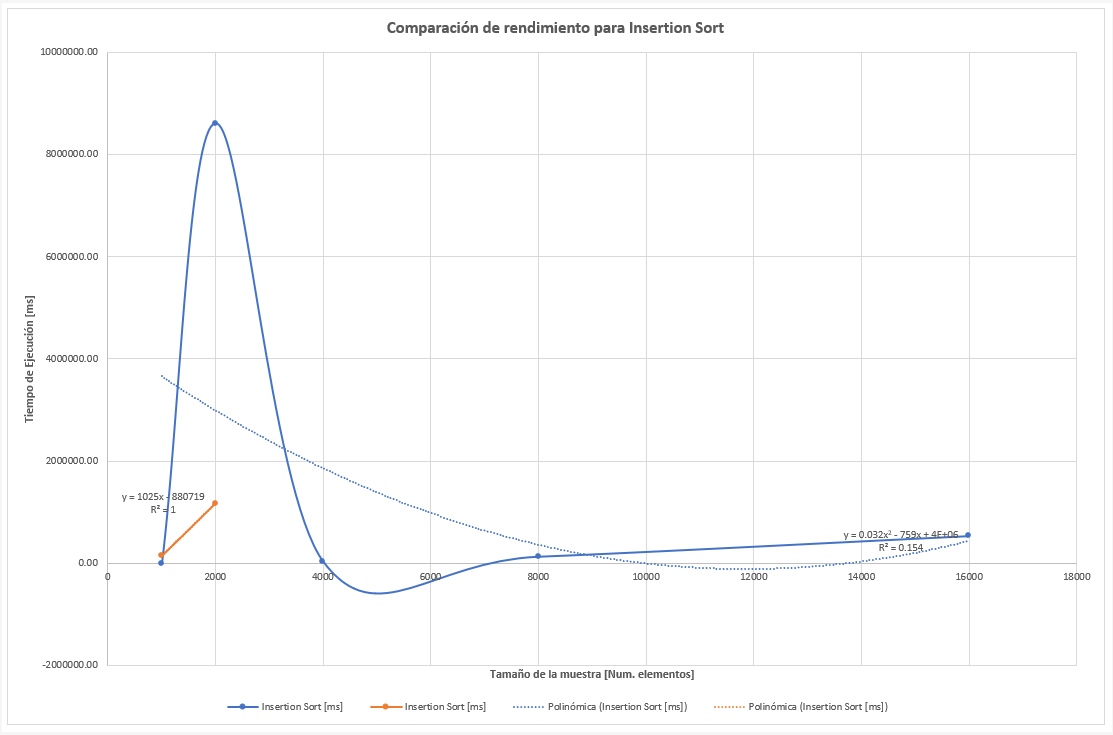
* Cinco gráficas generadas por los resultados de las pruebas de rendimiento en la **Maquina 1.**
  + Comparación de rendimiento ARRAYLIST.



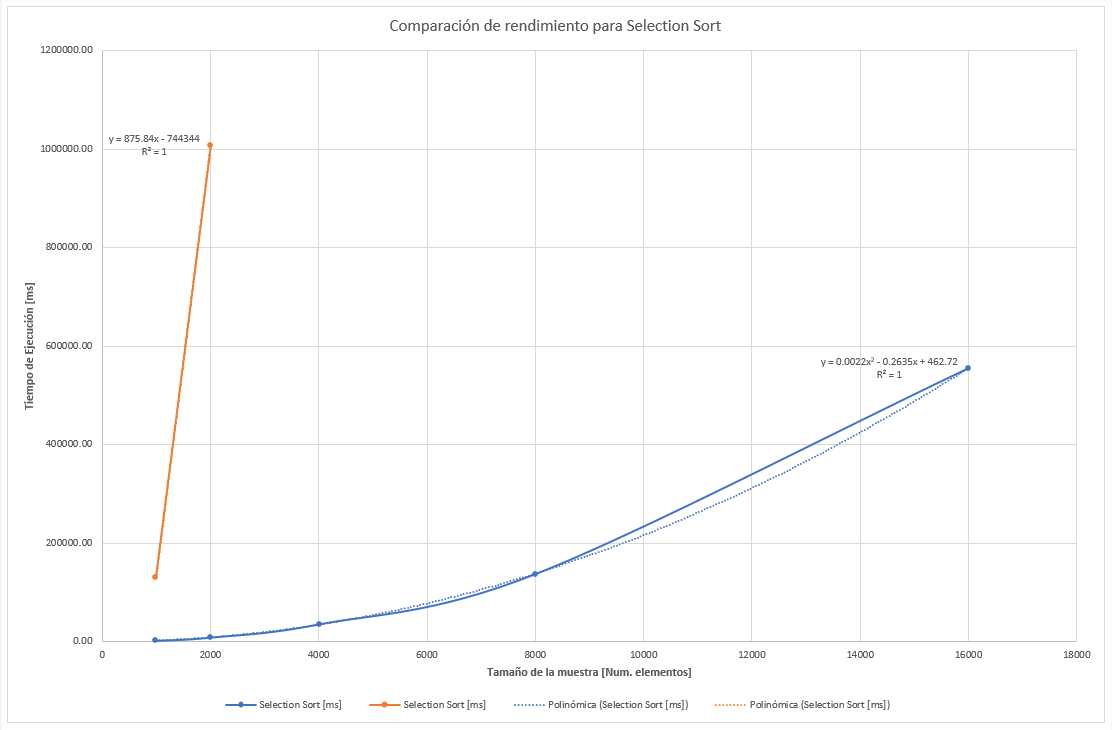
* + Comparación de rendimiento LINKED\_LIST.



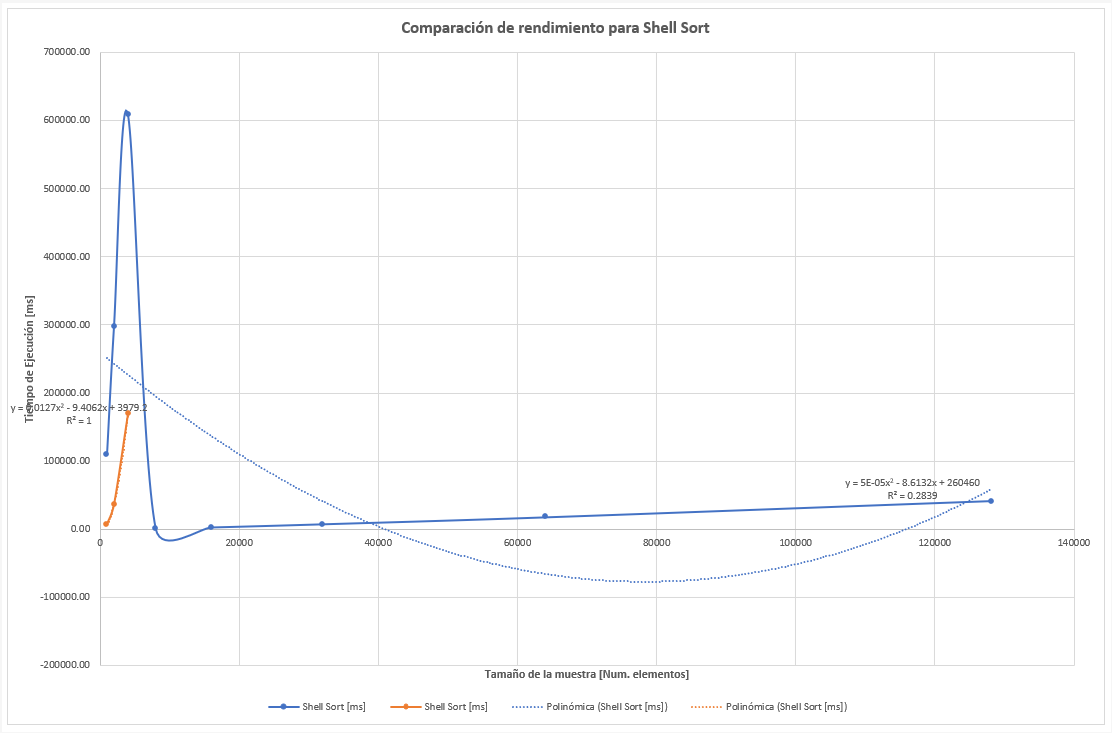
* + Comparación de rendimiento para Insertion Sort.



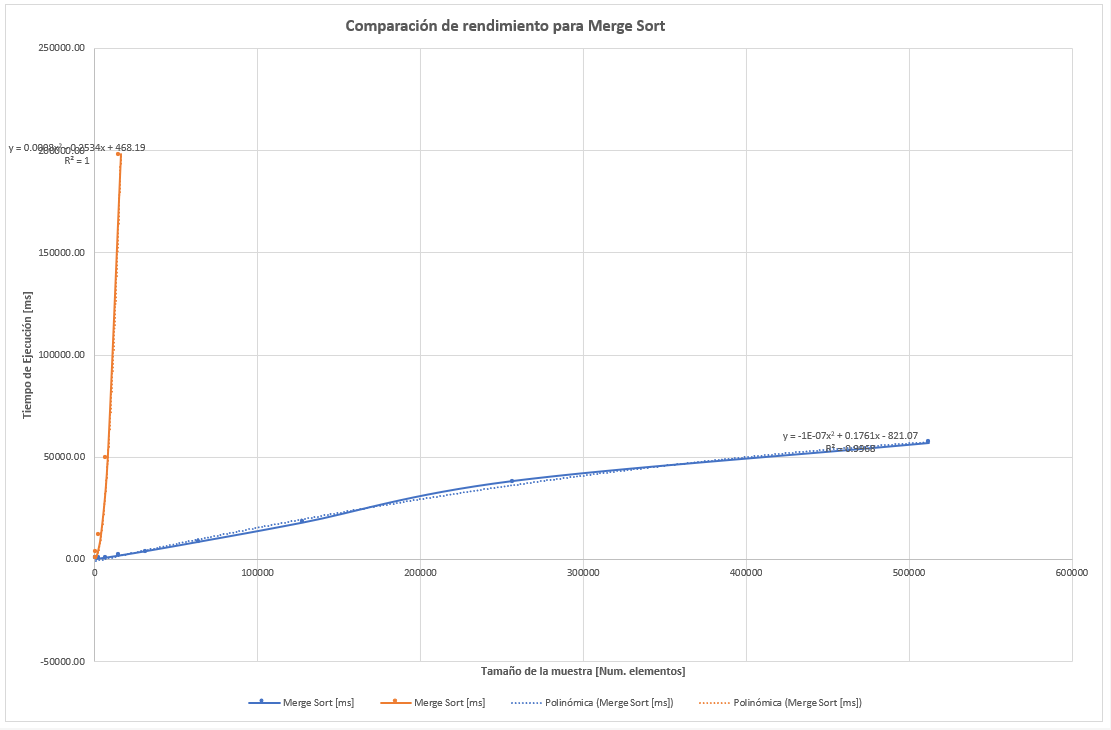
* + Comparación de rendimiento para Selection Sort.



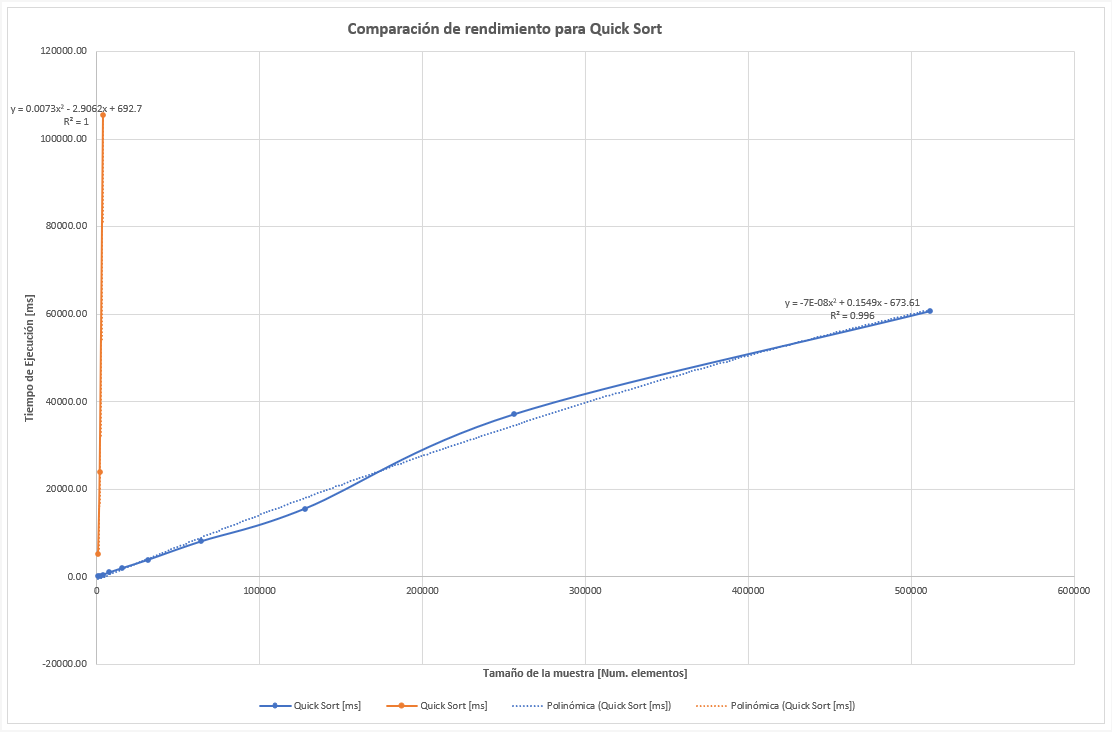
* + Comparación de rendimiento para Shell Sort.



* + 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 | 781.25 | 739.58 | 46.88 | 15.63 | 26.04 |
| 2000 | 3156.25 | 2968.75 | 93.75 | 41.67 | 52.08 |
| 4000 | 13078.12 | 12343.75 | 250.00 | 93.75 | 109.38 |
| 8000 | 53593.75 | 55734.38 | 531.25 | 239.58 | 244.79 |
| 16000 | 222937.50 | 226546.88 | 1250.00 | 520.83 | 520.83 |
| 32000 | Excede tiempo | Excede tiempo | 3046.88 | 1114.58 | 1114.58 |
| 64000 | Excede tiempo | Excede tiempo | Error\* | 2270.83 | 2468.75 |
| 128000 | Excede tiempo | Excede tiempo | Error\* | 5067.71 | 5213.54 |
| 256000 | Excede tiempo | Excede tiempo | Error\* | 10927.08 | 11177.08 |
| 512000 | Excede tiempo | Excede tiempo | Error\* | 17119.79 | 17578.13 |

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

\*Error descrito en la práctica anterior.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tamaño de la muestra (LINKED\_LIST)** | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 71671.88 | 65953.12 | 3359.38 | 1255.21 | 182.29 |
| 2000 | 560562.50 | 498562.50 | 16984.38 | 6296.88 | 697.92 |
| 4000 | Excede tiempo | Excede tiempo | 77875.00 | 25942.71 | 2812.50 |
| 8000 | Excede tiempo | Excede tiempo | 343109.38 | 129015.63 | 10921.88 |
| 16000 | Excede tiempo | Excede tiempo | Excede tiempo | 543494.79 | 45916.67 |
| 32000 | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo | 187562.50 |
| 64000 | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo | 759083.33 |
| 128000 | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo |
| 256000 | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo |
| 512000 | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo | Excede tiempo |

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

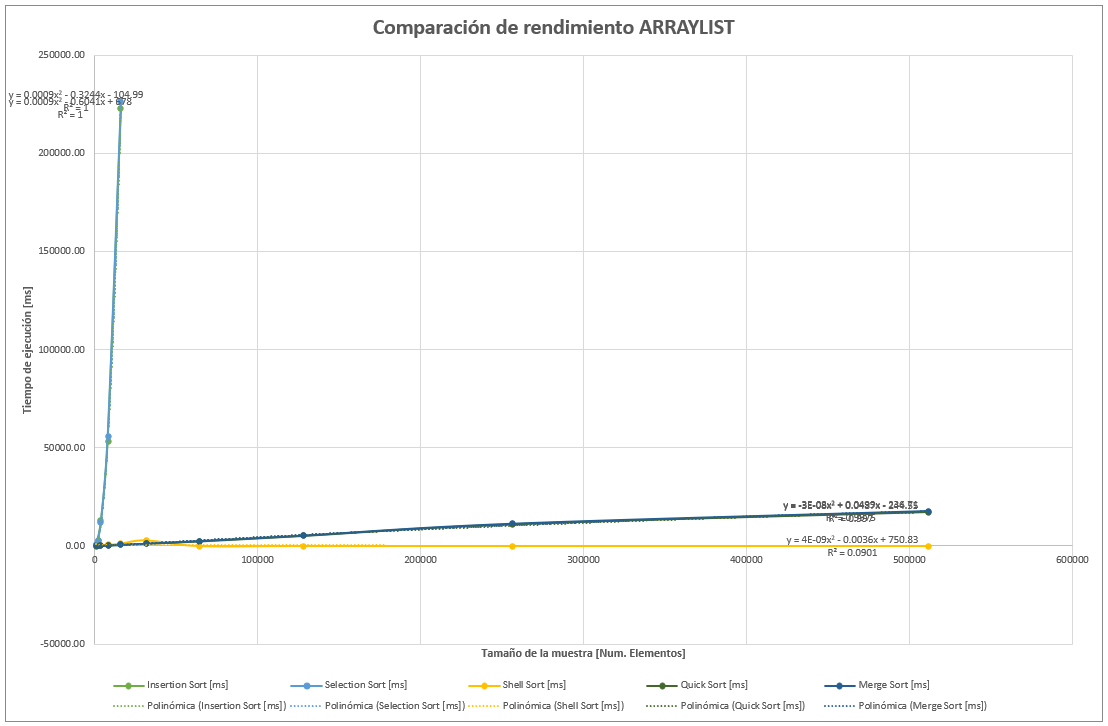
|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| Merge sort |  | x |
| Quick sort | x |  |

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

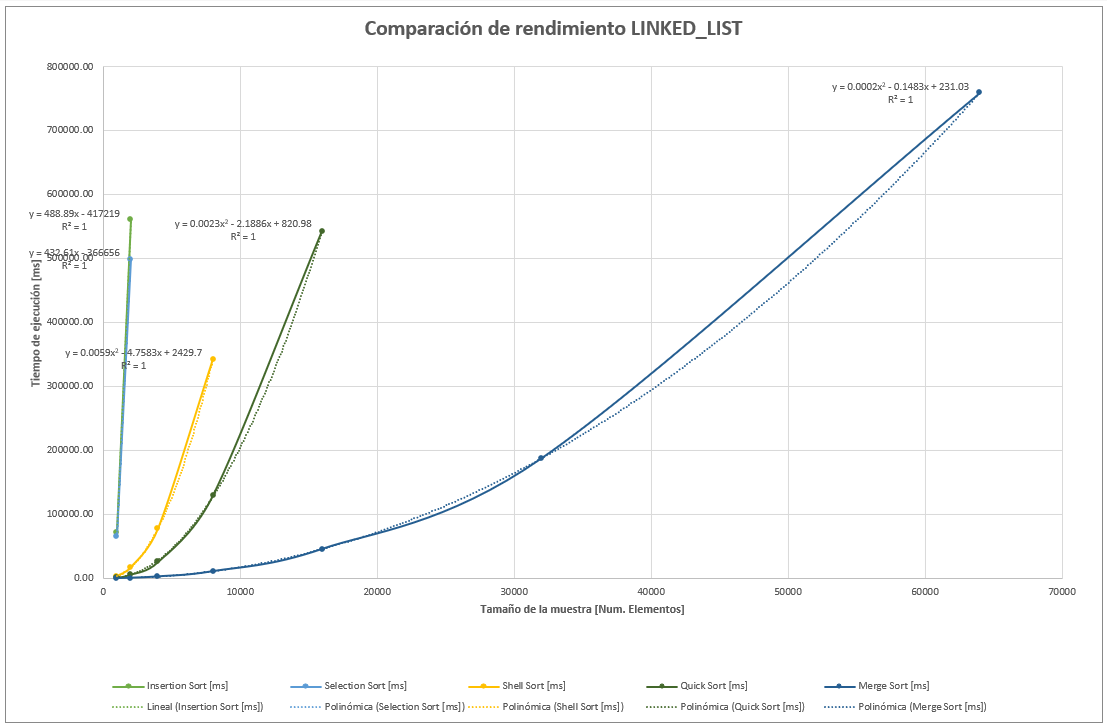
Se ha marcado en la anterior tabla, para cada tipo de estructura de datos, qué tipo de ordenamiento recursivo fue más eficiente en las pruebas. Según los datos y los tiempos exhibidos, se encontró que, para el arreglo, fue más eficiente el quick sort, y para la lista enlazada, el merge sort. Además, se ha determinado que el arreglo (array\_list) es el tipo de estructura de datos que exhibió tiempos de ejecución menores, resultando ser también la mejor opción entre las dos presentadas.

## **Graficas**

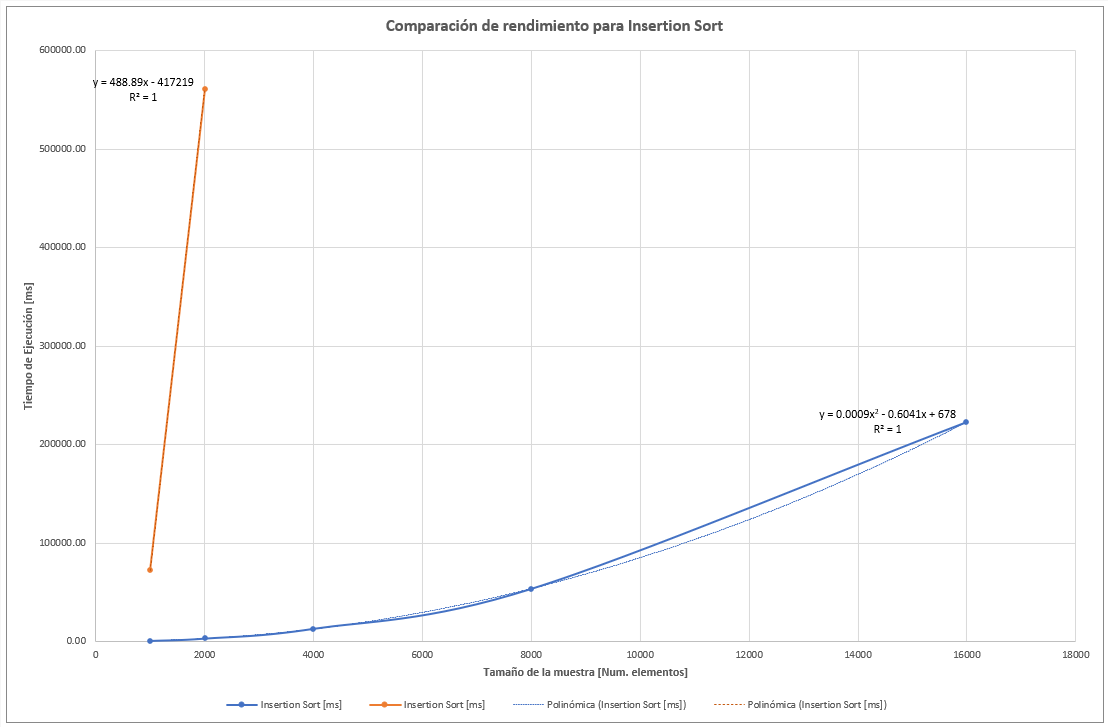
* Cinco gráficas generadas por los resultados de las pruebas de rendimiento en la **Maquina 2.**
  + Comparación de rendimiento ARRAYLIST.



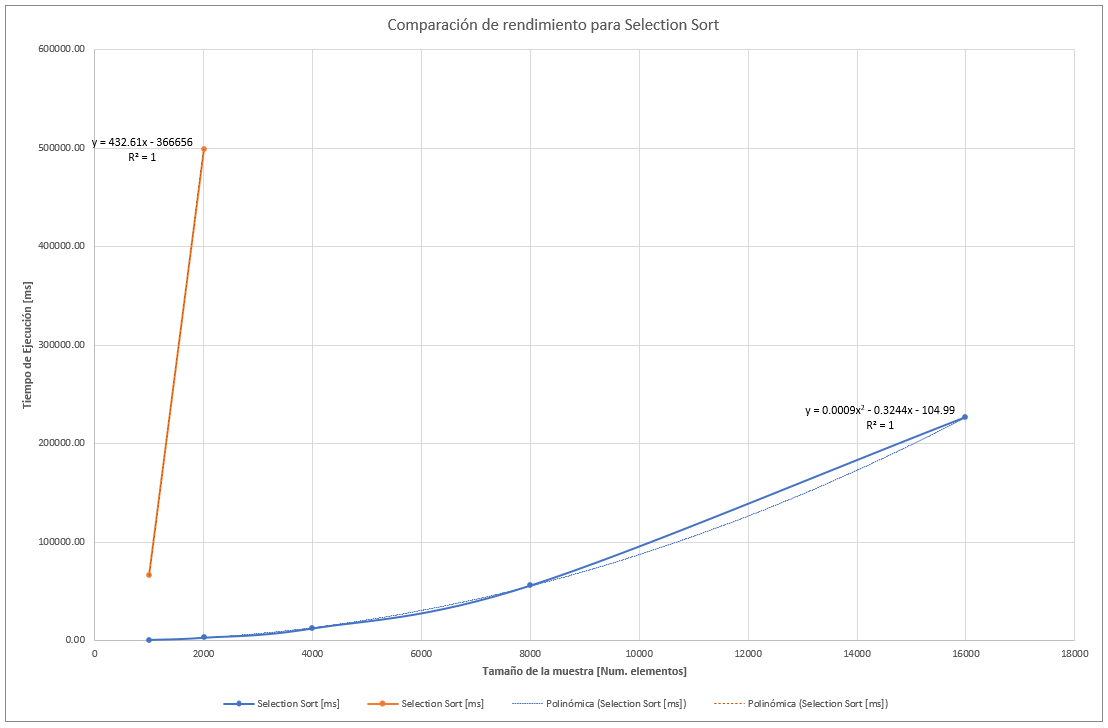
* + Comparación de rendimiento LINKED\_LIST.



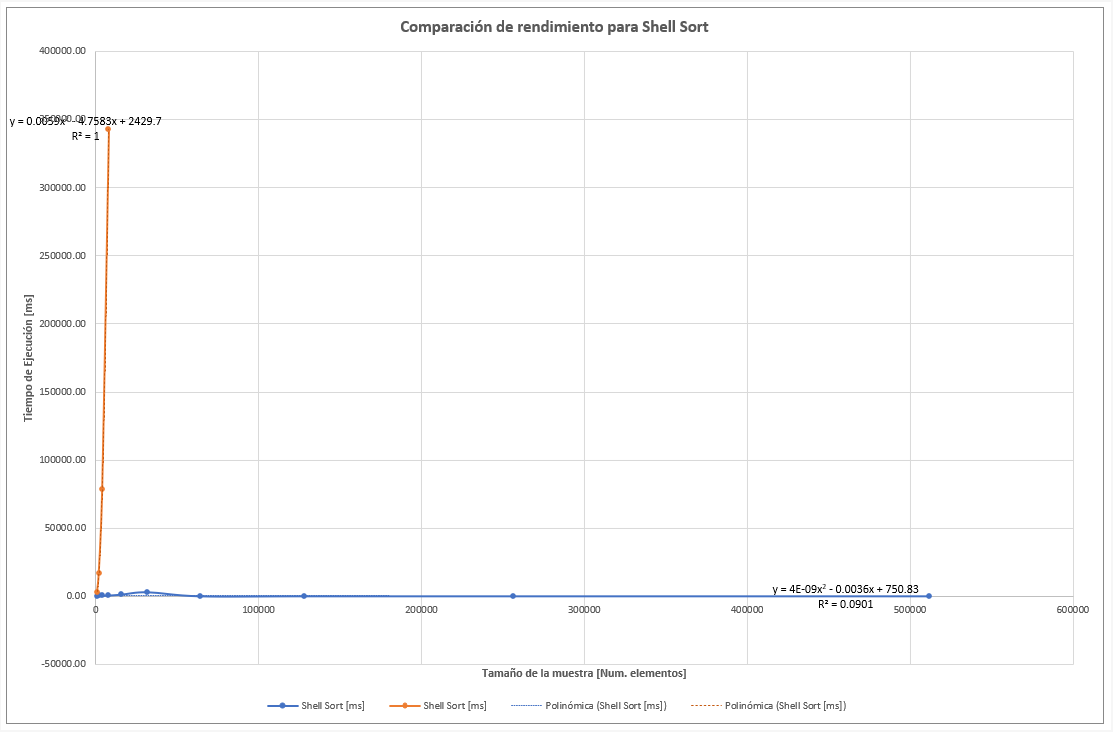
* + Comparación de rendimiento para Insertion Sort.



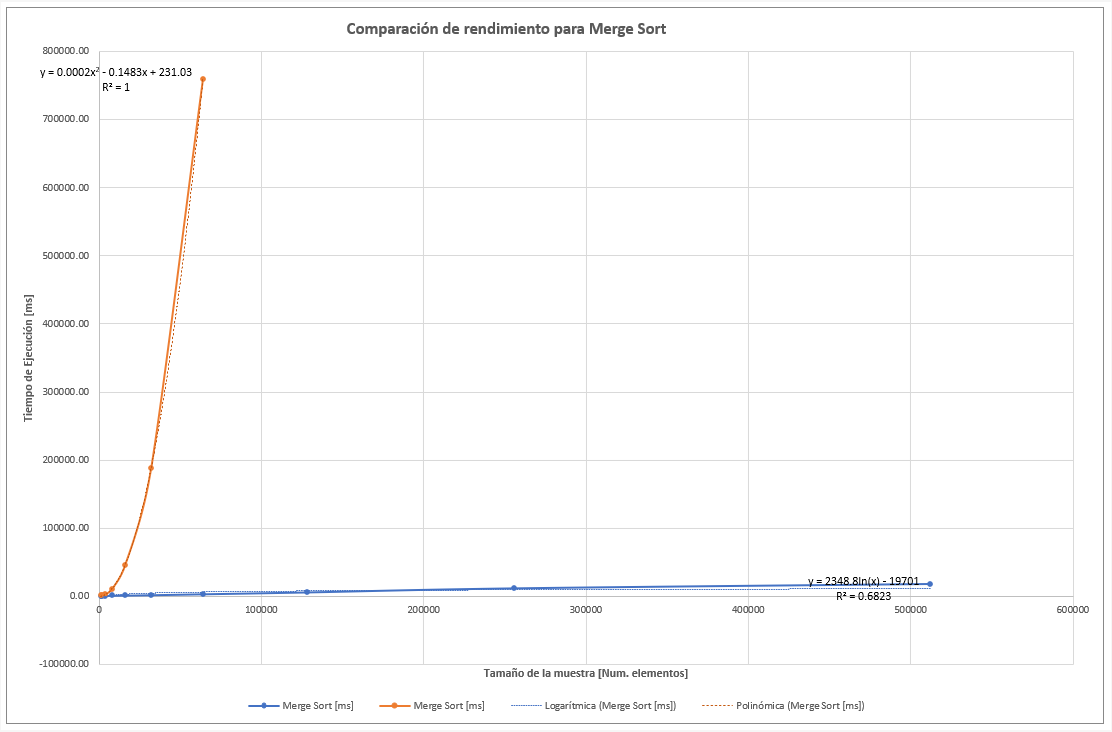
* + Comparación de rendimiento para Selection Sort.



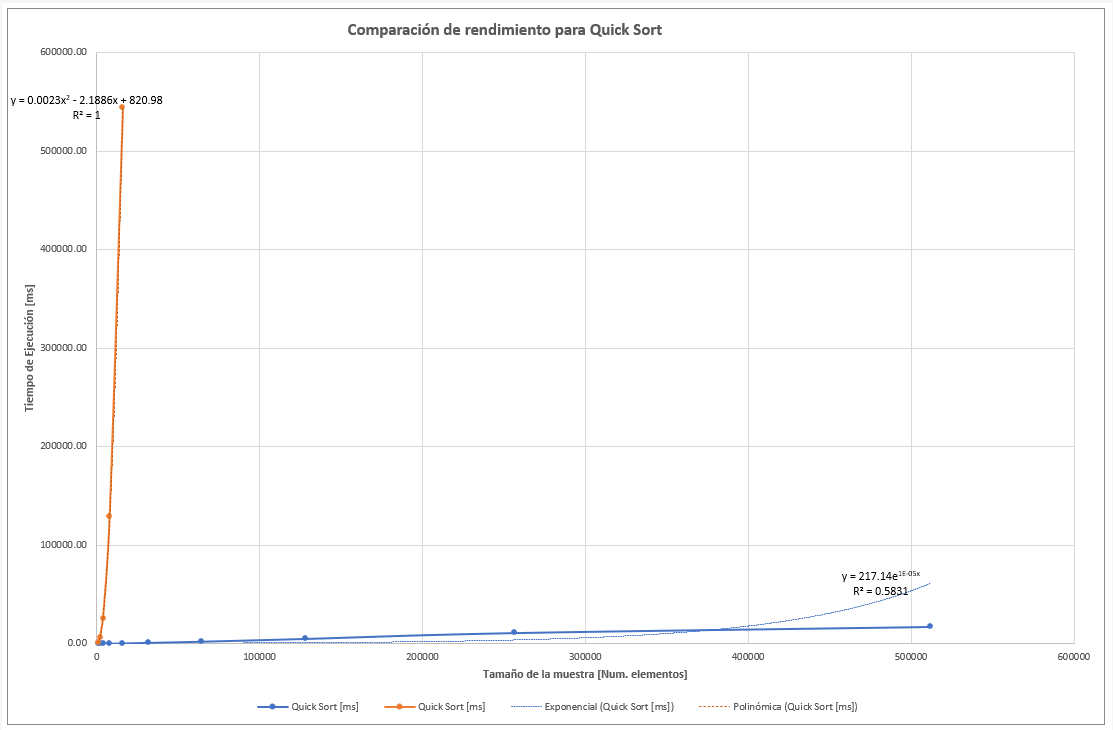
* + Comparación de rendimiento para Shell Sort.



* + 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?

Teóricamente se ha observado que, de acuerdo con los órdenes de complejidad de cada algoritmo, debería ser más eficiente el Merge Sort.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algoritmo** | **Mejor caso** | **Promedio** | **Peor caso** |
| Quick Sort |  |  |  |
| Merge Sort |  |  |  |

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

Sí. Como se muestra en la tabla 1, las dos máquinas son bastante distintas entre sí. Esto se vio claramente evidenciado en los datos; si bien los excesos de tiempo (15 minutos) fueron alcanzados para tamaños de muestra parecidos, cada tiempo de ejecución le tomó a la máquina 1 en general el doble que a la máquina 2.

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

Estas diferencias podrían deberse principalmente al rendimiento de los procesadores de cada máquina (se ha encontrado que tiene un mejor rendimiento el procesador de la máquina 2, con un rendimiento global estimado en 217.3% mejor que el del procesador correspondiente a la máquina 1). La diferencia en memoria RAM indica que la máquina 1 debería ser más eficiente, pero este no fue el caso.

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

Se ha encontrado que la mejor estructura de datos ha sido el arreglo (array\_list), pues exhibió menores tiempos de ejecución para todos los ordenamientos que la lista encadenada (linked\_list), y en las gráficas en general exhibe órdenes de complejidad correspondientes a algoritmos de mayor eficiencia.

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.

|  |  |  |
| --- | --- | --- |
| **Ranking Algoritmos** | **Array list** | **Linked list** |
| 1 | Quick Sort | Merge Sort |
| 2 | Mergue Sort | Quick Sort |
| 3 | Shell Sort | Shell Sort |
| 4 | Selection Sort | Selection Sort |
| 5 | Insertion Sort | Insertion Sort |

Utilizando los algoritmos con la base de datos de los videos, la eficiencia de los algoritmos a veces varía, respecto a los demás, con la cantidad de datos suministrada (e incluso dependiendo del rendimiento de la máquina). Por ejemplo, es el caso de Insertion sort y Selection sort con el Array List. En todo caso, en general, el mejor Algoritmo encontrado es el Quick Sort, siendo mas especificos en el Array List. Ya que el Quick sort toma un valor al azar como pivote, vuelve el ordenamiento de orden .

# **Referencias**

[1] Technical City. (s.f.). AMD A10-8700P vs. Ryzen 5 3500U. *Technical City*. Obtenido de https://technical.city/es/cpu/A10-8700P-vs-Ryzen-5-3500U

[2] Wikipedia. (2021). *Wikipedia*. Obtenido de https://en.wikipedia.org/wiki/Sorting\_algorithm#Classification