OBSERVACIONES DE LA PRACTICA

Estudiante 1: Paola Campiño Cod 202020785

Estudiante 2: Santiago Ballén Cod 202023544

|  |  |  |
| --- | --- | --- |
|  | **Máquina 1** | **Máquina 2** |
| **Procesadores** | AMD Ryzen 5 2500U with radeon Vega Mobile Gfx, 2.00GHz | Intel Core i3-3220 CPU @ 3.30GHz |
| **Memoria RAM (GB)** | 6,00GB (5,65GB utilizable) | 4,00 GB (3,89 GB utilizable) |
| **Sistema Operativo** | Sistema operativo de 64 bits, procesador x64 | Sistema operativo de 64 bits, procesador x64 |

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 | 1909.38 | 1765.65 | 78.13 | 46.88 | 62.5 |
| 2000 | 6515.63 | 7078.13 | 187.50 | 203.13 | 125.00 |
| 4000 | 26546.88 | 28859.38 | 437.50 | 718.75 | 296.88 |
| 8000 | 108515.08 | 119062.50 | 1078.13 | 1453.13 | 718.75 |
| 16000 | 289427.08 | 485015.63 | 2578.00 | 1453.13 | 1312.50 |
| 32000 | 1752203.13 | 1904609.38 | 5875.00 | 3000.00 | 3062.50 |
| 64000 |  |  | 13744.79 | 6125.00 | 5875.00 |
| 128000 |  |  | 31744.79 | 12723.96 | 1278.25 |
| 256000 |  |  | 81494.79 | 27703.13 | 27817.70 |
| 512000 |  |  | 132791.67 | 62317.07 | 42520.83 |

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 | 85531.25 | 70953.13 | 4250.00 | 3265.63 | 421.88 |
| 2000 | 694406.25 | 599640.63 | 19593.75 | 15703.63 | 1640.25 |
| 4000 |  |  | 91515.63 | 65296.88 | 7015.63 |
| 8000 |  |  | 449515.63 | 335250.00 | 24531.25 |
| 16000 |  |  | 1282500.00 | 1282500.00 | 101843.75 |
| 32000 |  |  |  |  | 396328.13 |
| 64000 |  |  |  |  |  |
| 128000 |  |  |  |  |  |
| 256000 |  |  |  |  |  |
| 512000 |  |  |  |  |  |

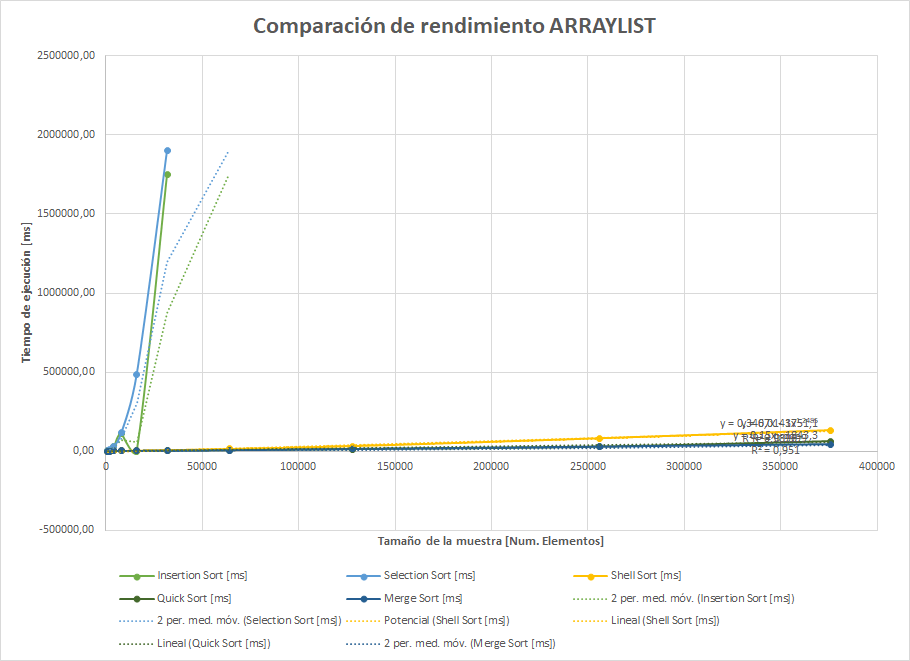
Tabla 3. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación lista enlazada.

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

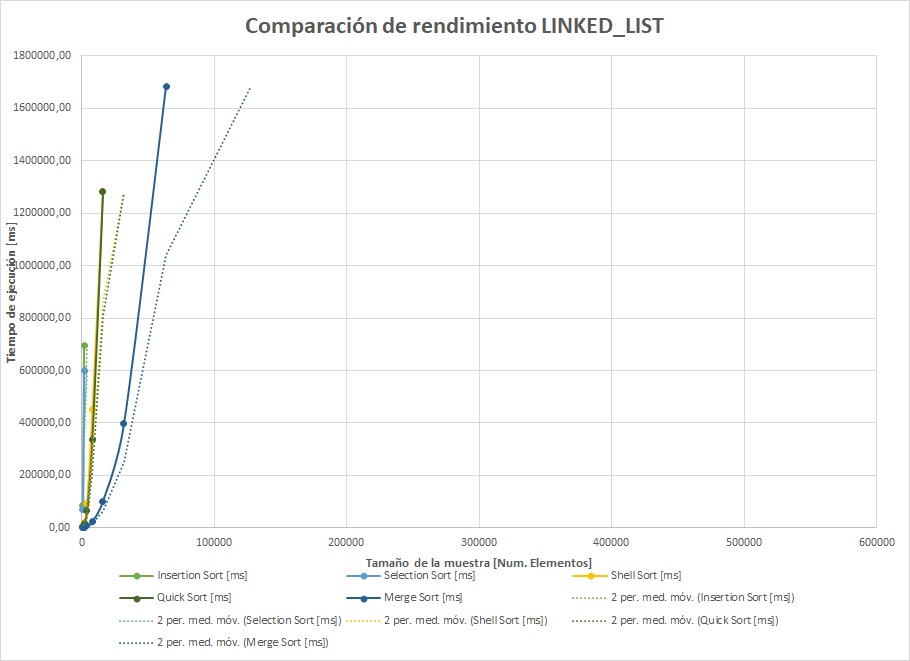
Tabla 4. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.

## **Graficas**

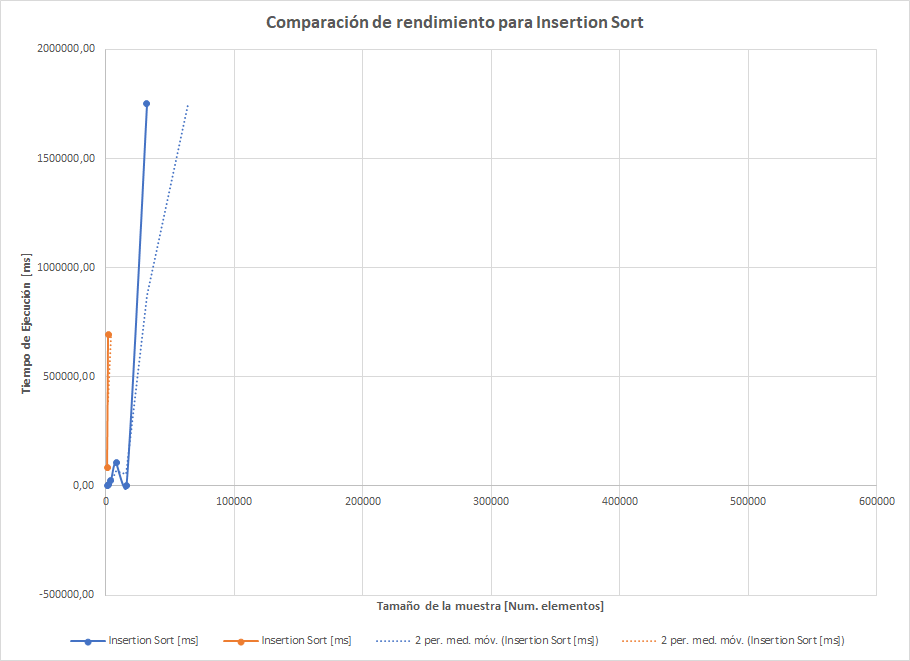
* Cinco gráficas generadas por los resultados de las pruebas de rendimiento en la **Maquina 1.**
  + Comparación de rendimiento ARRAYLIST.



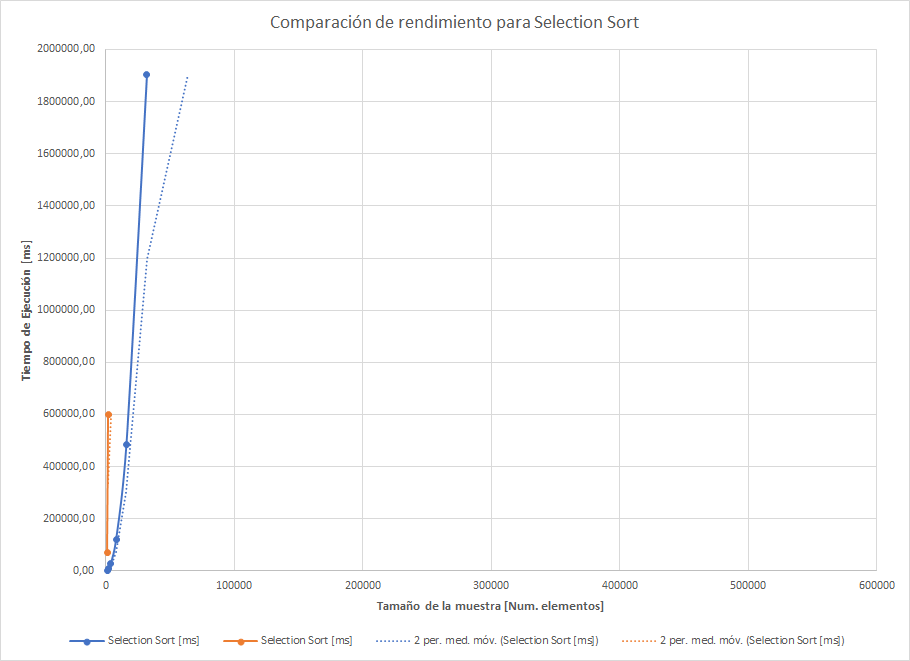
* + Comparación de rendimiento LINKED\_LIST.



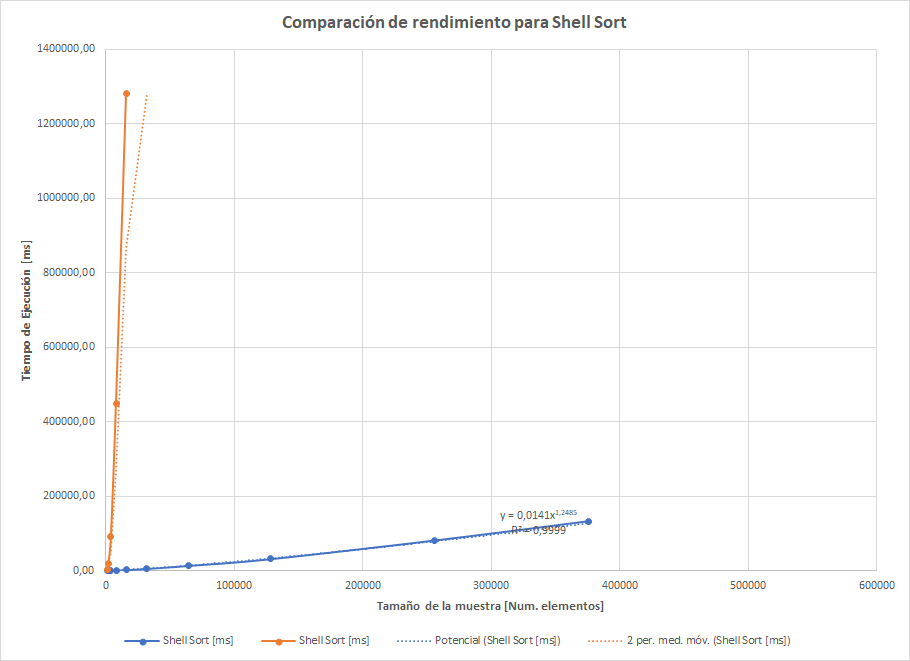
* + Comparación de rendimiento para Insertion Sort.



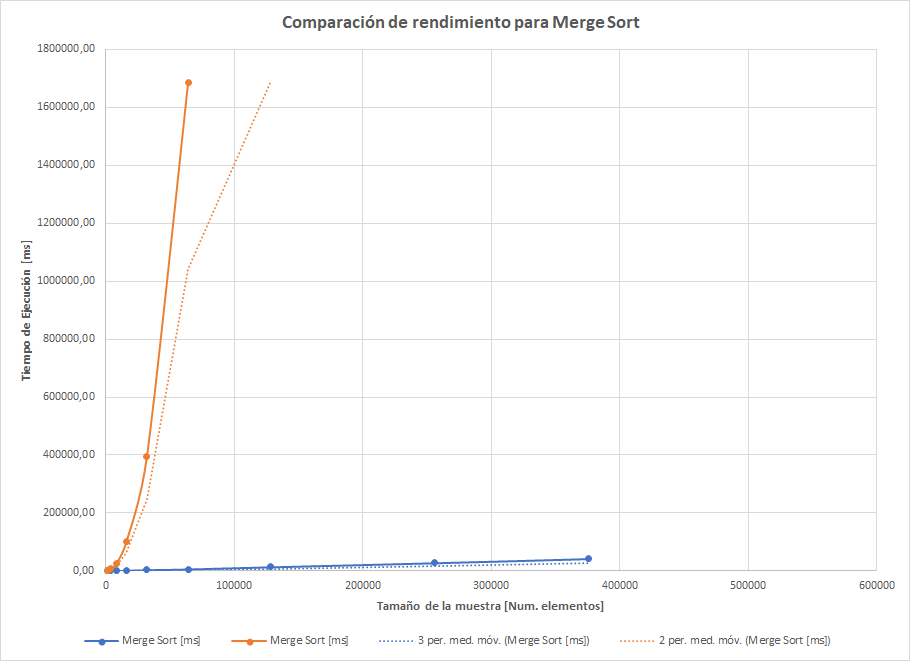
* + Comparación de rendimiento para Selection Sort.



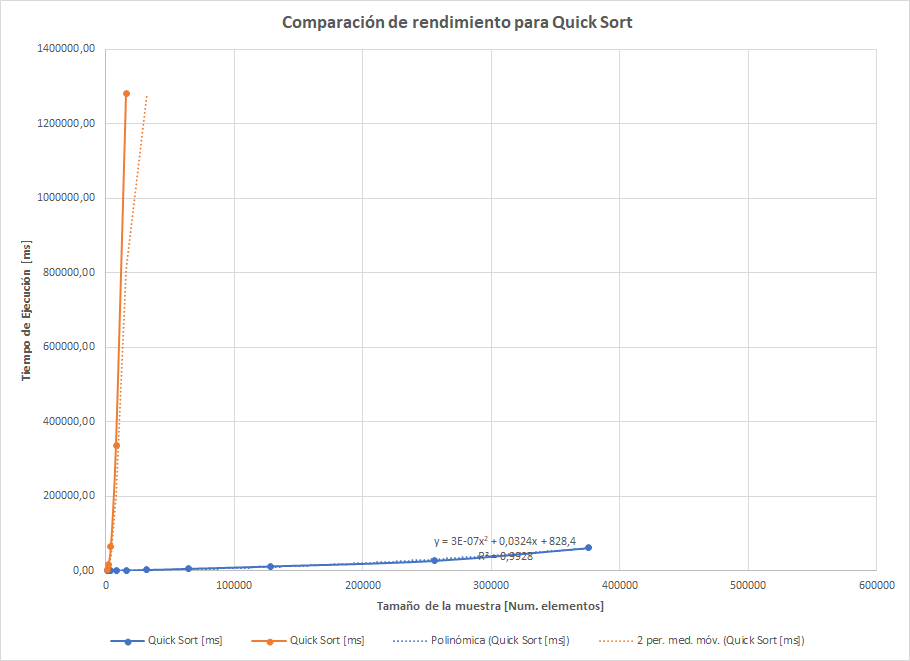
* + Comparación de rendimiento para Shell Sort.



* + Comparación de rendimiento para MergeSort.



* + Comparación de rendimiento para QuickSort.



# **Maquina 2**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **T**amaño de la muestra (ARRAYLIST) | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 1015,63 | 1156,25 | 46,88 | 31,25 | 46,88 |
| 2000 | 4078,13 | 4515,63 | 125,00 | 93,75 | 93,75 |
| 4000 | 16812,50 | 18593,75 | 281,25 | 171,88 | 171,88 |
| 8000 | 68250,00 | 76703,13 | 703,13 | 437,50 | 390,63 |
| 16000 | 290140,63 | 317328,13 | 1500,00 | 921,88 | 859,38 |
| 32000 | 1160703,13 | 1276609,38 | 3734,38 | 1750,00 | 1828,13 |
| 64000 | 5572843,75 | 5146765,63 | 9203,13 | 4171,88 | 3906,25 |
| 128000 | 22078062,50 | 20673546,88 | 21343,75 | 8390,63 | 8609,38 |
| 256000 |  |  | 58031,25 | 20078,13 | 18859,38 |
| 375942 |  |  | 92156,25 | 45609,38 | 28687,50 |

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 | 83750,00 | 66312,50 | 3906,25 | 3031,25 | 406,25 |
| 2000 | 708593,75 | 617781,25 | 18968,75 | 16171,88 | 1640,63 |
| 4000 |  |  | 92515,63 | 65984,38 | 6390,63 |
| 8000 |  |  | 466078,13 | 351781,25 | 26625,00 |
| 16000 |  |  |  |  | 96546,88 |
| 32000 |  |  |  |  | 399234,38 |
| 64000 |  |  |  |  |  |
| 128000 |  |  |  |  |  |
| 256000 |  |  |  |  |  |
| 512000 |  |  |  |  |  |

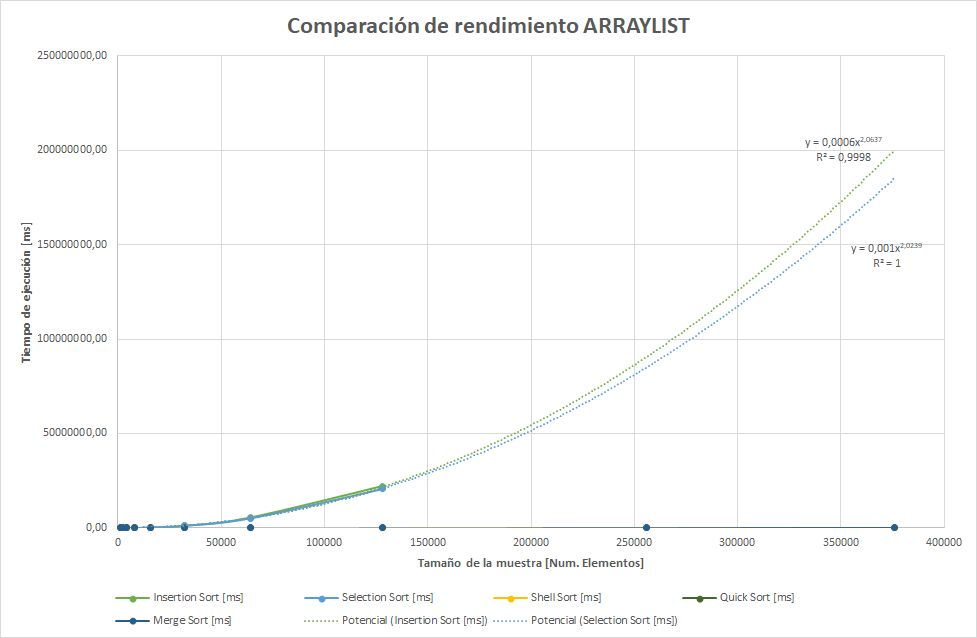
Tabla 3. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación lista enlazada.

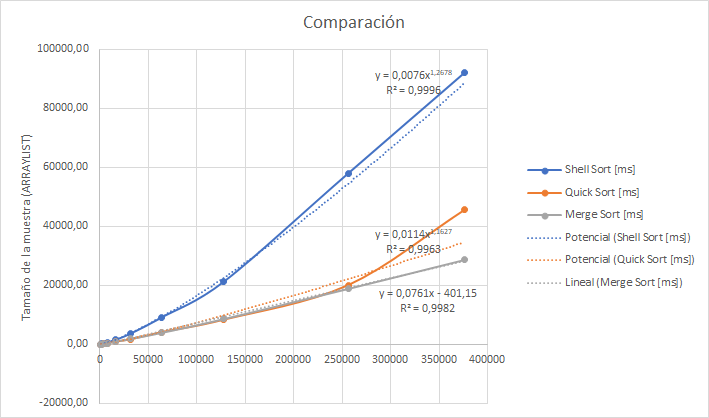
|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| Merge sort | ✓ | ✓ |
| Quick sort |  |  |

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

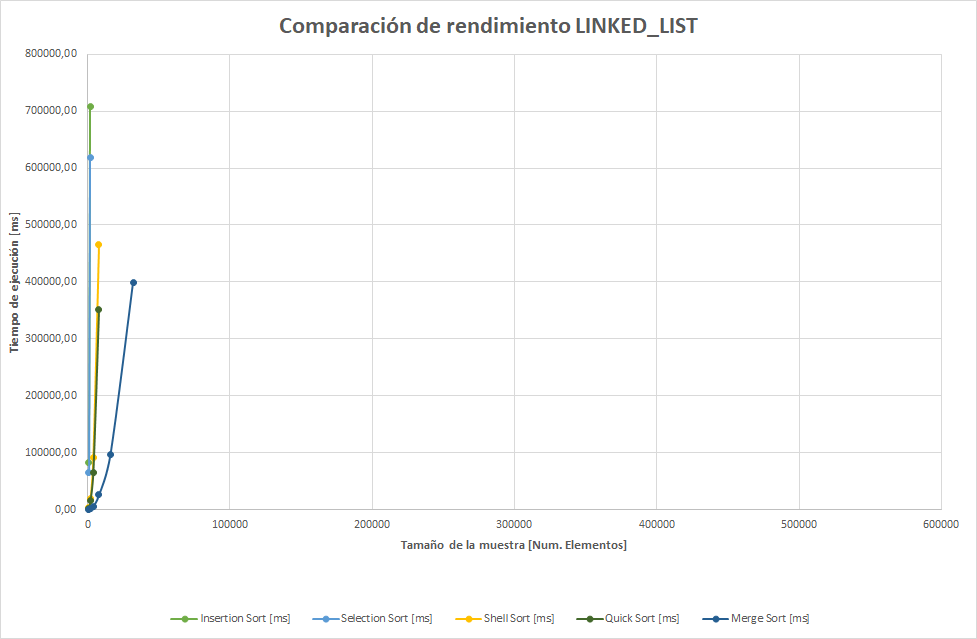
## **Graficas**

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

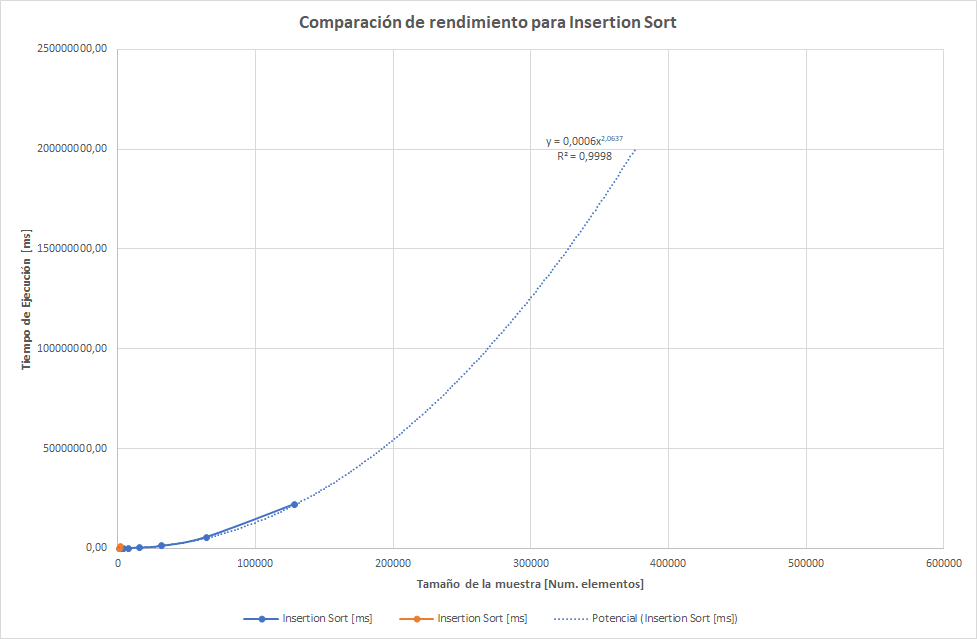




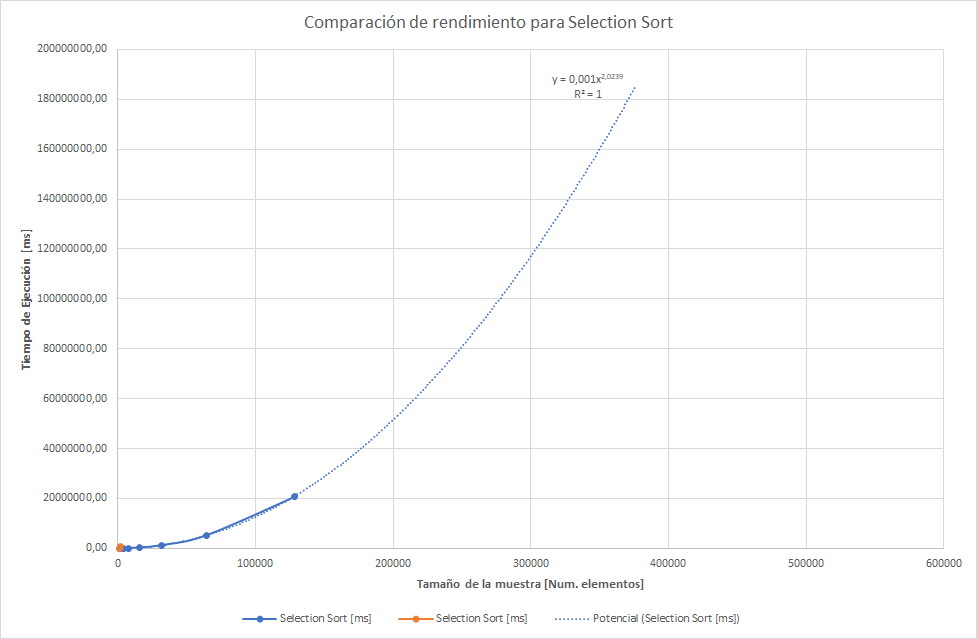
* + Comparación de rendimiento LINKED\_LIST.



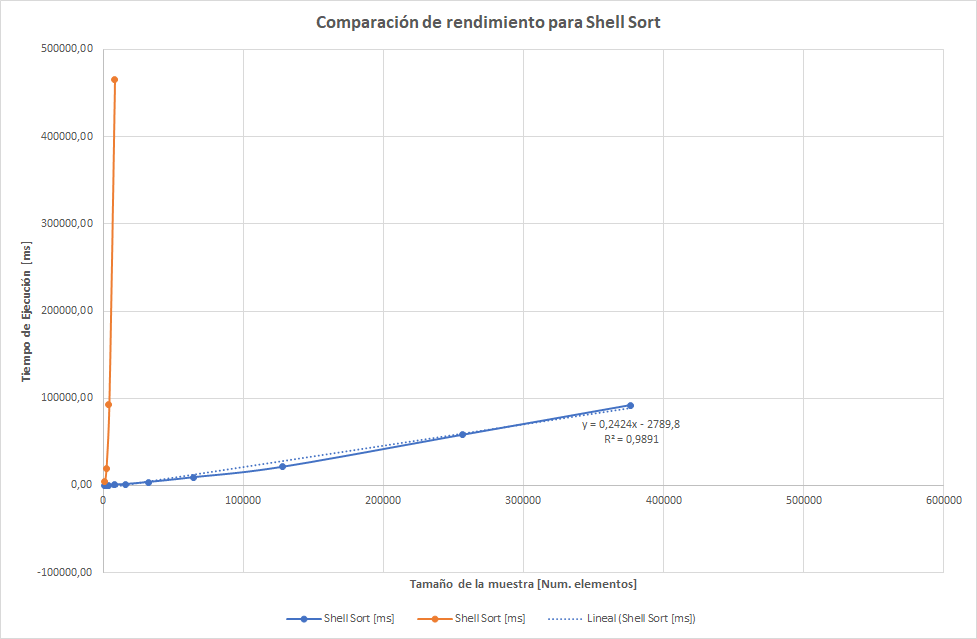
* + Comparación de rendimiento para Insertion Sort.



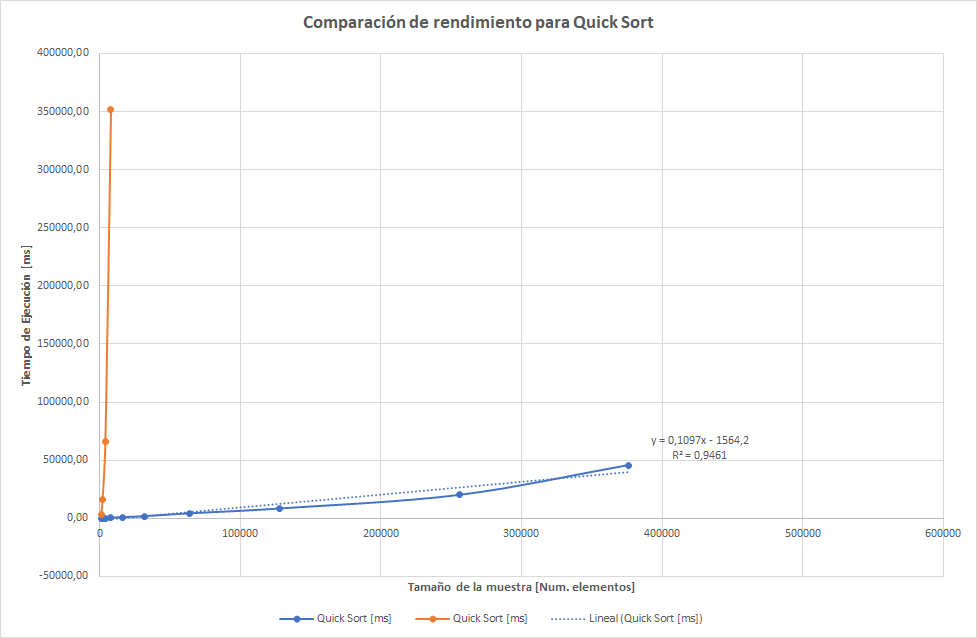
* + Comparación de rendimiento para Selection Sort.



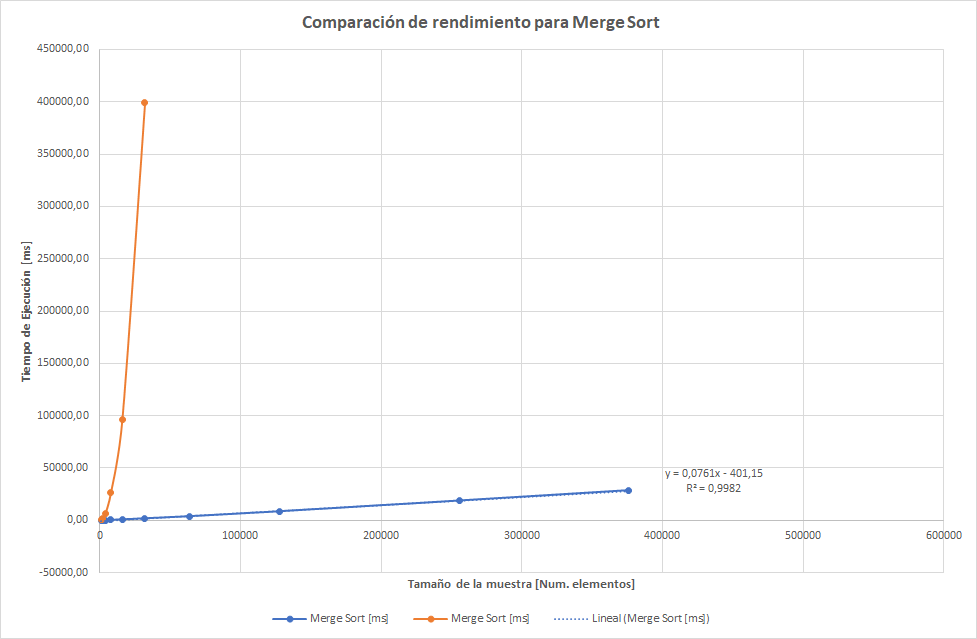
* + Comparación de rendimiento para Shell Sort.



* + 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?

**Insertion Sort:**

ARRAY LIST:

En teoría la conplejidad del Insertion Sort en su peor caso es de O(N^2) y en su mejor O(N). Sin embargo al tomar los tiempos de respuesta por parte de este algorithmo, se pudo evidenciar una grafica exponencial.

SINGLE LINKED:

En teoría la conplejidad del Insertion Sort en su peor caso es de O(N^2) y en su mejor O(N). Sin embargo al tomar los tiempos de respuesta por parte de este algorithmo, se pudo evidenciar una grafica exponencial con tiempos mayores a cuando los datos se encontraban organanizados en un arreglo.

**Selection Sort:**

ARRAY LIST:

En teoría la complejidad del Selection Sort en su peor y mejor caso debe ser de O(n^2). Al poner en practica el algorithmo fue muy diferente a lo que esperabamos, ya que al graficar los datos se pudo evidenciar una grafica muy similar a una expoencial.

SINGLE LINKED:

En teoría la complejidad del Selection Sort en su peor y mejor caso debe ser de O(n^2). Al poner en practica el algorithmo fue muy diferente a lo que esperabamos, ya que al graficar los datos se pudo evidenciar una grafica muy similar a una expoencial.

**Shell Sort:**

ARRAY LIST:

En teoría la complejidad del Shell Sort en su peor caso es O(n^3/2), y en el mejor O(Nlog3(N)). Al poner en practica el algorithmo fue muy similiar una grafica O(Log(N)). Los tiempos fueron mucho mejores a lo que se esperaba.

SINGLE LINKED:

En teoría la complejidad del Shell Sort en su peor caso es O(n^3/2), y en el mejor O(Nlog3(N)). Al poner en practica el algorithmo fue muy similiar una grafica Exponencial. Los tiempos fueron mucho peores a lo que se esperaba.

**Merge Sort:**

ARRAY LIST:

En teoria el algorithmo de ordenaminteo Merge sort debería tener un orden de complejidad de O(NLog(N) ). Sin embargo, en la practica el tiempo de respuesta fue menor y la grafica se asimilaba a un algorithmo con complejidad de O(Log(N)).

SINGLE LINKED:

En teoria el algorithmo de ordenaminteo Merge sort debería tener un orden de complejidad de O(NLog(N) ). Sin embargo, en la practica el tiempo de respuesta fue mucho mayor ya que la grafica se asimilaba a la de un algorithmo de tipo exponencial

**Quick sort:**

ARRAY LIST:

El algorithmo Quick sort en teoría debería tener una complejidad de O(½ N^2). Sin emabargo al graficarse los tiempos de respuesta fue posible ver una grafica de O(Log(N)).

LINKED LIST:

El algorithmo Quick sort en teoría debería tener una complejidad de O(½ N^2). Sin emabargo al graficarse los tiempos de respuesta fue posible ver una grafica que parecía ser Exponencial.

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

A simple vista la tendencia de las curvas parece ser la misma en todos y cada uno de los graficos a excepción del Insertion Sort.

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

Está diferencia problemente se deba algún error durante la toma de datos, como por ejemplo algún tiempo mal copiado. También esto se puede deber a un caso particularmente mal para el algoritmmo en cuestion y por esto se ve un pico en la curva.

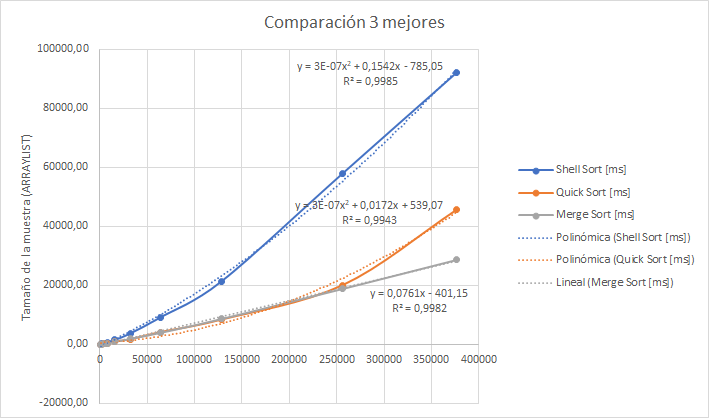
1. ¿Cuál Estructura de Datos es mejor utilizar si solo se tiene en cuenta los tiempos de ejecución de los algoritmos?

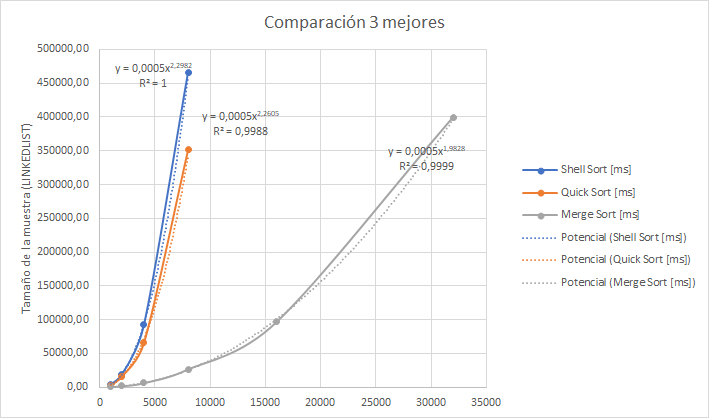
Se puede concluir que es mejor tener los datos organizados en un ARRAY LIST y en cuanto al algoritmo con los mejores tiempos de repuesta se puede afirmar que el Merge es el mejor.

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.

Debido a la falta de datos causada por la ineficiencia de los algoritmos Selection Sort y Insertion Sort, además de no haber demostrado tener ningún tipo de tendencia logarítmica, consideramos pertinente primeramente descalificar a estos dos algoritmos ya que son virtualmente incomparables con el resto de los algoritmos. Con base en lo anterior, decidimos dividir el ranking en dos partes, la primera, los tres mejores algoritmos, disputado entre: Shell Sort , Quick Sort y Merge Sort; y la segunda, los dos peores algoritmos: Insertion Sort y Selection Sort.

Utilizando los siguientes graficos:

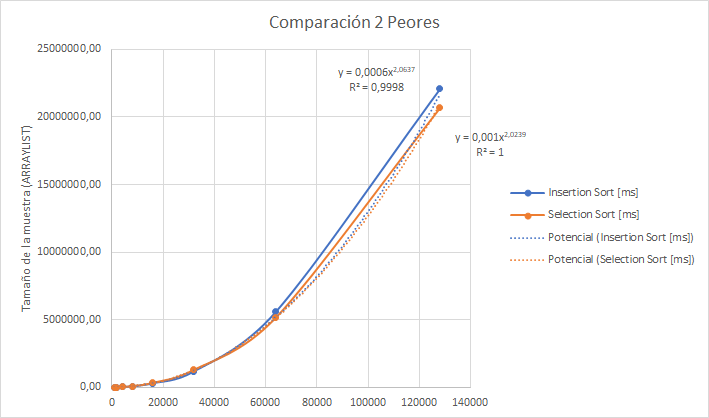




Llegamos a la conclusión que los 3 mejores algoritmos de mejor a peor serian:

1. Merge Sort
2. Quick Sort
3. Shell Sort

Ahora, ultilizando el siguiente grafico(No tomamos en cuenta el LINKED LIST ya que apenas si tenía datos en este caso):



Llegamos a la conclusión que los 2 mejores algoritmos de peor a aún más peor serian:

1. Selection Sort
2. Insertion Sort

Finalemente, uniendos ambos resultados, obtenemos que el orden final sería:

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