OBSERVACIONES DE LA PRACTICA

Javier Cerino Cod 202020873

Marco Zuliani Cod 202022412

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | Intel(R) Core (TM)i5-1035G1 CPU @ 1.00GHz | AMD Ryzen 3 3300U with Radeon Vega Mobile Gfx 2.10 GHz |
| Memoria RAM (GB) | 8.0 GB (7.76 GB Utilizable) | 12.0 GB (9.92 GB Utilizable) |
| Sistema Operativo | Windows 10 Home Single Language | Windows 10 Home Single Language |

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

# **Maquina 1**

## **Resultados**

|  |  |  |  |
| --- | --- | --- | --- |
| Tamaño de la muestra (ARRAY\_LIST) | Insertion Sort (ms) | Selection Sort (ms) | Shell Sort (ms) |
| 1000 | 762.50 | 953.12 | 31,25 |
| 2000 | 3038.12 | 4109.37 | 78.12 |
| 4000 | 12348.95 | 16312.34 | 218.75 |
| 8000 | 48505.20 | 70187.26 | 531.25 |
| 16000 | 206859.37 | 272296.87 | 1187,5 |
| 32000 | 1180406.25 | 1649125.08 | 2765.62 |
| 64000 | 5079984.37 | 5065421.87 | 7059.35 |
| 128000 |  |  | 16109.37 |
| 256000 |  |  | 43296.87 |
| 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) |
| 1000 | 41368.27 | 45468.75 | 2000 |
| 2000 | 376529.25 | 398734.37 | 9562.5 |
| 4000 | 3286927.5 | 3054406.25 | 44046.87 |
| 8000 |  |  | 220109.37 |
| 16000 |  |  | 989375.0 |
| 32000 |  |  | 4870421.87 |
| 64000 |  |  |  |
| 128000 |  |  |  |
| 256000 |  |  |  |
| 512000 |  |  |  |

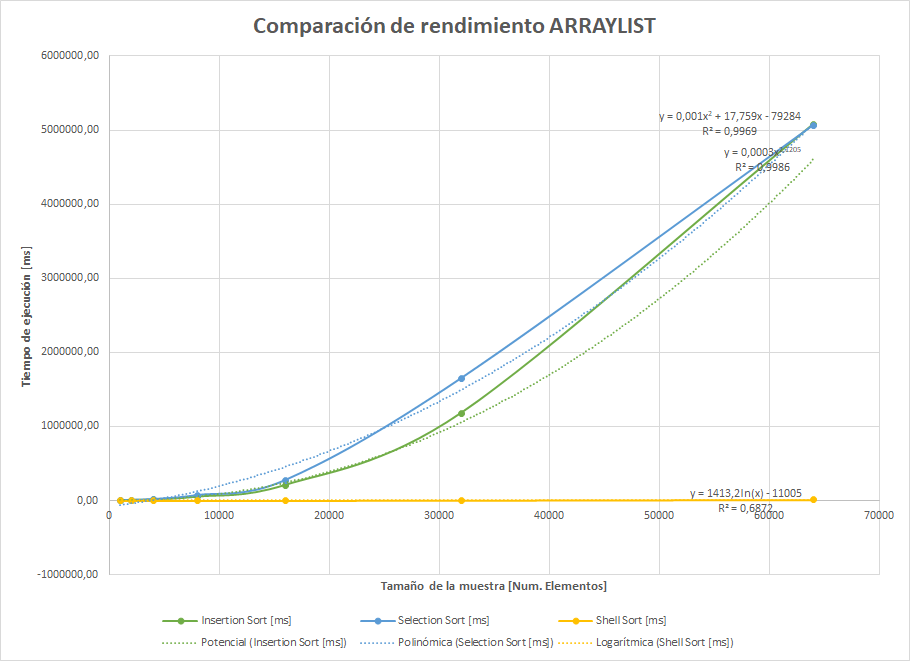
Tabla 3. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación lista enlazada.

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| Insertion sort | x |  |
| Selection sort | x |  |
| Shell sort | x |  |

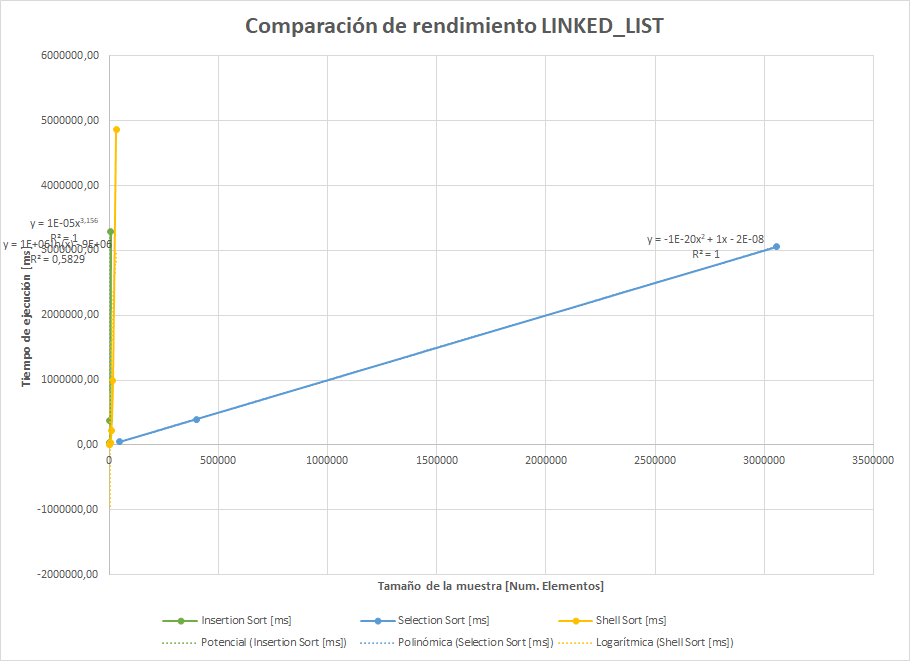
Tabla 4. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.

## **Graficas**

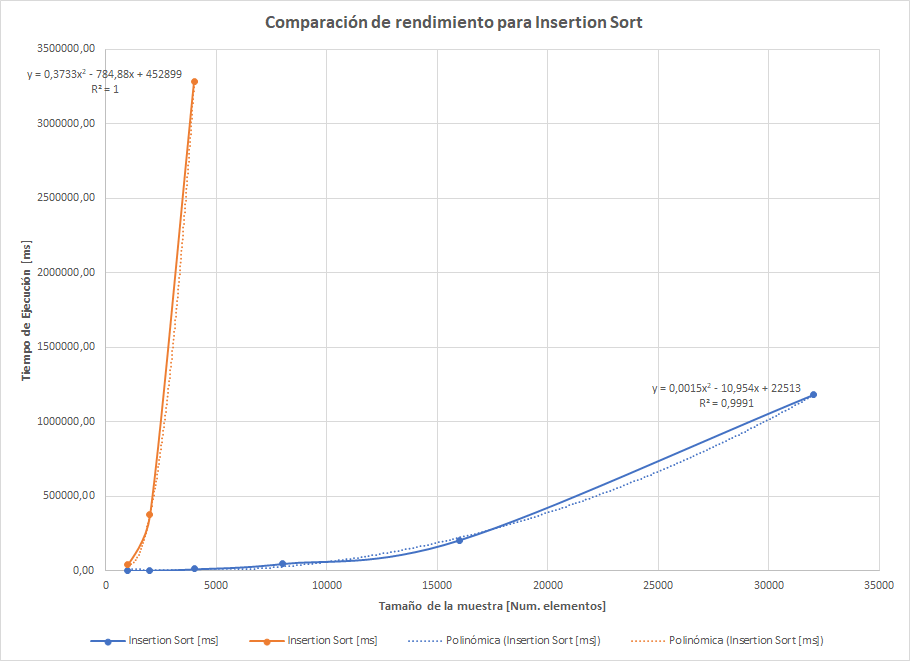
* Cinco gráficas generadas por los resultados de las pruebas de rendimiento en la **Maquina 1.**
  + Comparación de rendimiento ARRAYLIST.



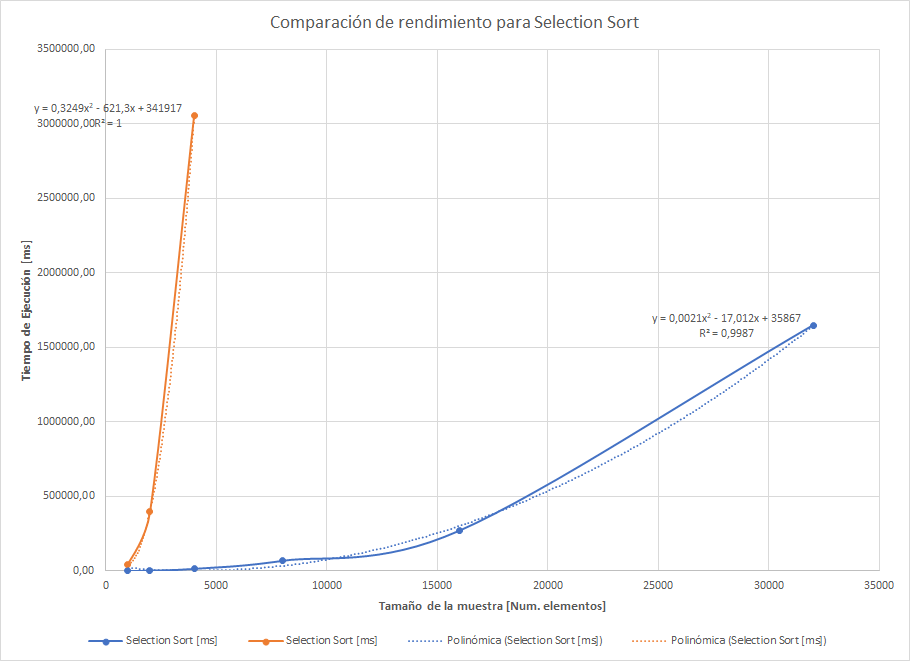
* + Comparación de rendimiento LINKED\_LIST.



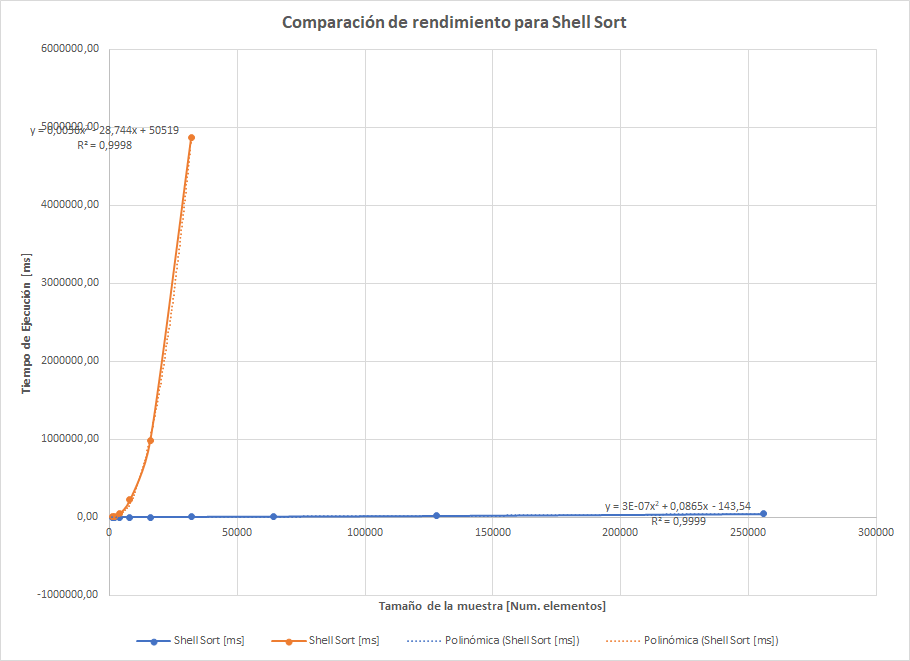
* + Comparación de rendimiento para Insertion Sort.



* + Comparación de rendimiento para Selection Sort.



* + Comparación de rendimiento para Shell Sort.



# **Maquina 2**

## **Resultados**

|  |  |  |  |
| --- | --- | --- | --- |
| Tamaño de la muestra (ARRAY\_LIST) | Insertion Sort (ms) | Selection Sort (ms) | Shell Sort (ms) |
| 1000 | 790,62 | 921,87 | 46,87 |
| 2000 | 3168,75 | 3703,12 | 109,37 |
| 4000 | 13057,29 | 14796,87 | 234,37 |
| 8000 | 53348,95 | 64109,37 | 593,75 |
| 16000 | 229156,25 | 269843,75 | 1265,62 |
| 32000 | 914734,37 | 1639343,75 | 2984,37 |
| 64000 | 4346218,75 | 4762843,75 | 7125,0 |
| 128000 |  |  | 17187,5 |
| 256000 |  |  | 44218,75 |
| 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) |
| 1000 | 51843,75 | 56343,75 | 3875,0 |
| 2000 | 421015,62 | 449390,62 | 17265,62 |
| 4000 | 4110875,0 | 3265046,87 | 83531,25 |
| 8000 |  |  | 467437,5 |
| 16000 |  |  | 2745734,37 |
| 32000 |  |  |  |
| 64000 |  |  |  |
| 128000 |  |  |  |
| 256000 |  |  |  |
| 512000 |  |  |  |

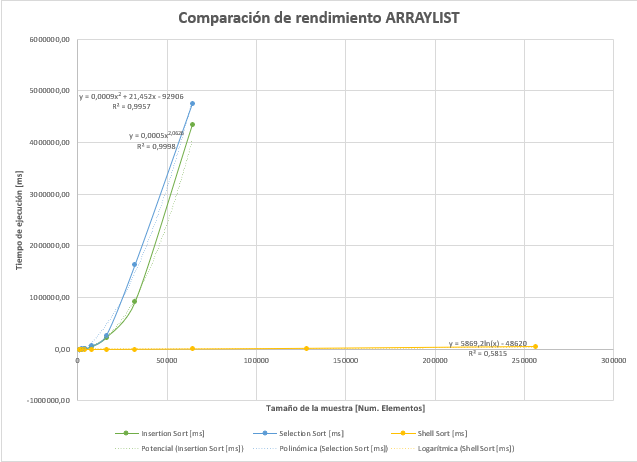
Tabla 3. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación lista enlazada.

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| Insertion sort | x |  |
| Selection sort | x |  |
| Shell sort | x |  |

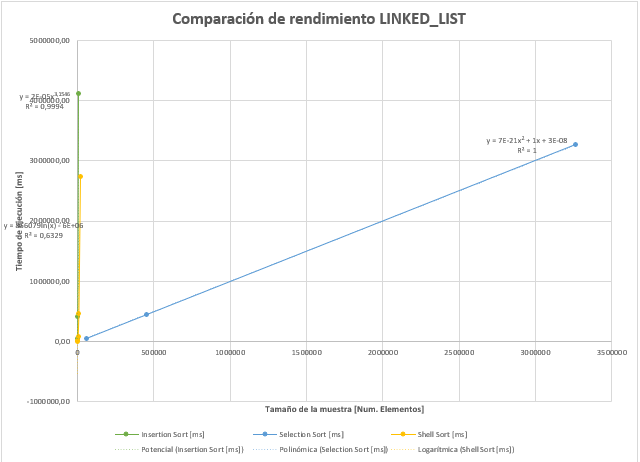
Tabla 4. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.

## **Graficas**

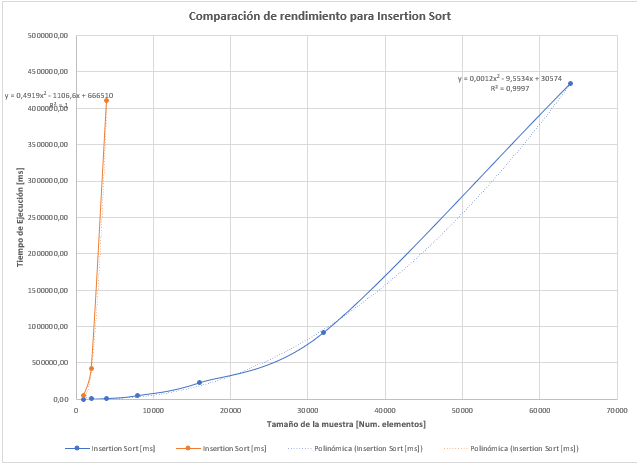
* Cinco gráficas generadas por los resultados de las pruebas de rendimiento en la **Maquina 2.**
  + Comparación de rendimiento ARRAYLIST.



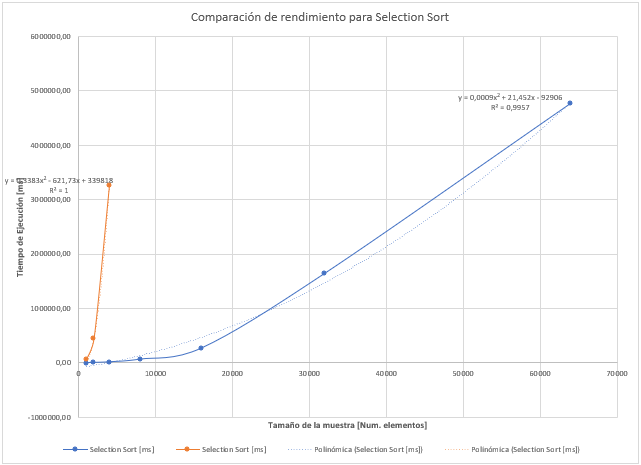
* + Comparación de rendimiento LINKED\_LIST.



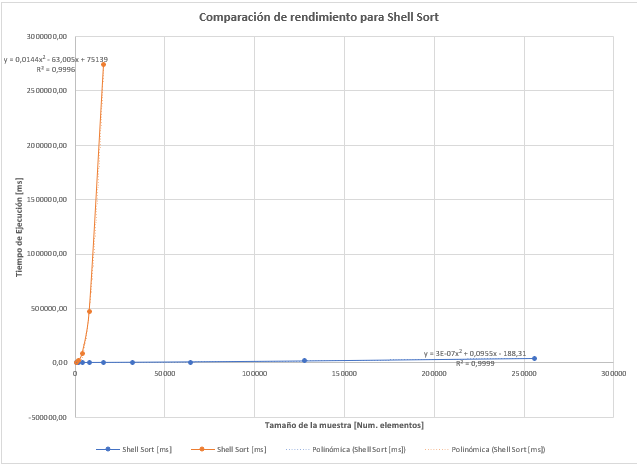
* + Comparación de rendimiento para Insertion Sort.



* + Comparación de rendimiento para Selection Sort.



* + Comparación de rendimiento para Shell Sort.



# **Preguntas de análisis**

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

El comportamiento de los algoritmos de ordenamiento tienen un comportamiento similar a lo estipulado teoricamente. Esto se puede apreciar por ejemplo en la creación de las ecuación de insertion y selection que tiene una complejidad de O(n2) y las ecuaciones lo muestran a la perfección. El único que se comporta diferente es el Shell que se esperaria una complejidad de O(n\*lg(n)) pero en estos casos se representan con una ecuación de tipo cuadratica aunque el coeficiente del termino a adquiere un valor muy muy pequeño al momento de trabajar con array list.

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

Si, existe una diferencia al momento de ejecturar las mismas pruebas en máquinas diferentes, esto se puede apreciar sobretodo al momento de trabajar con los single linked donde vemos unas diferencias de tiempo un poco más elevadas.

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

Puede ser debido a la memoria ram que tienen a disposición los diferentes equipos, no obstante, también prodría ser culpa del procesador y de los diversos subprocesos que se desarollan dentro de la máquina.

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

Si solo se quiere tener en cuenta el tiempo de ejecución de los algoritmos la estructura de datos más eficiente sin duda alguna sería el Array List con el método de filtrado Shell Sorting. Si analizamos los datos obtenidos en las dos muestras observamos la notoria diferencia entre los tiempos de espera. Tal es la diferencia que se lograron ordenar 256000 datos con el shell en Array list, algo que ni se podía imaginar con los demás métodos de ordenamiento que para ordenas 64000 objetos se demoraban más de 4 millones de milisegundos.