OBSERVACIONES DE LA PRACTICA

Julián Ricardo Villate Torres Cod 202020509

Sergio Pardo Gutiérrez Cod 202025720

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | AMD A9-9425 | Intel(R) Core(TM) i7-8565 CPU @ 1.80GHz 1.99 GHz |
| Memoria RAM (GB) | 8 | 16 |
| Sistema Operativo | Windows 10 x64 | Windows 10 Home 64 bits |

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 | 1218.752 | 1203.125 | 46 |
| 2000 | 4921.875 | 4906.25 | 140 |
| 4000 | 25468.75 | 26906.25 | 468.75 |
| 8000 | 100687.5 | 99984.375 | 953.125 |
| 16000 |  |  |  |
| 32000 |  |  |  |
| 64000 |  |  |  |
| 128000 |  |  |  |
| 256000 |  |  |  |
| 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 | 102385.41 | 86125.0 | 4109.375 |
| 2000 | 780796.875 | 674578.125 | 21796.875 |
| 4000 | 6342421.875 |  |  |
| 8000 |  |  |  |
| 16000 |  |  |  |
| 32000 |  |  |  |
| 64000 |  |  |  |
| 128000 |  |  |  |
| 256000 |  |  |  |
| 512000 |  |  |  |

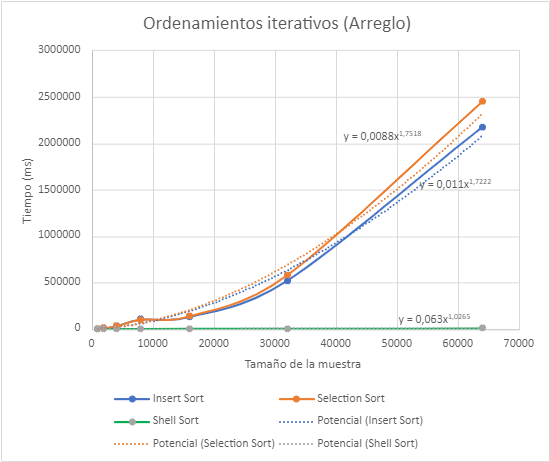
Tabla 3. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación lista enlazada.

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

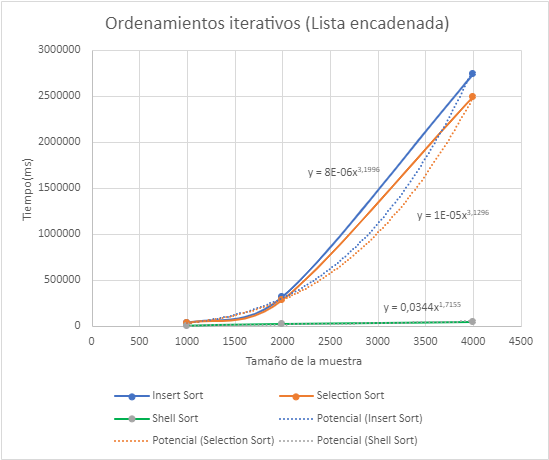
Tabla 4. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.

## **Graficas**

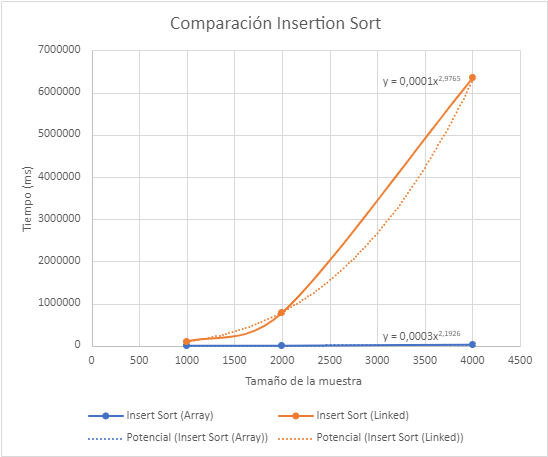
* Cinco gráficas generadas por los resultados de las pruebas de rendimiento en la **Maquina 1.**
  + **Comparación de rendimiento ARRAYLIST.**



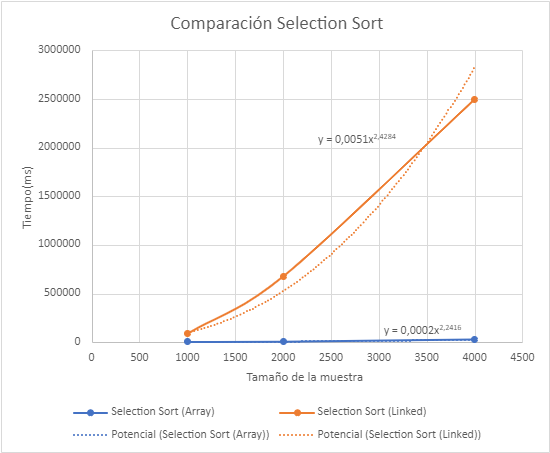
* + **Comparación de rendimiento LINKED\_LIST.**



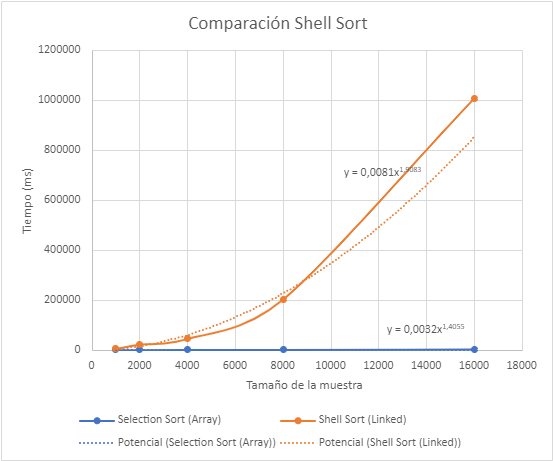
* + **Comparación de rendimiento para Insertion Sort.**



* + **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 | 407.25 | 453.12 | 15.62 |
| 2000 | 1,671.88 | 1,937.5 | 46.88 |
| 4000 | 6,921.88 | 7,734.38 | 125 |
| 8000 | 29,921.88 | 31,875 | 281.25 |
| 16000 | 128,156.25 | 135,125 | 671.88 |
| 32000 | 516,812.5 | 575,296.88 | 1,640.62 |
| 64000 | 2’173,296.88 | 2’445,812.5 | 4,015.62 |
| 128000 |  |  | 10,078.12 |
| 256000 |  |  | 23,625 |
| 375942 |  |  | 37,390.62 |

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 | 32,473.96 | 32,578.13 | 1,531.25 |
| 2000 | 315,328.13 | 282,500 | 9,000 |
| 4000 | 2’740,937.5 | 2’495,437.5 | 44,218.75 |
| 8000 |  |  | 200,750 |
| 16000 |  |  | 996,515.62 |
| 32000 |  |  |  |
| 64000 |  |  |  |
| 128000 |  |  |  |
| 256000 |  |  |  |
| 512000 |  |  |  |

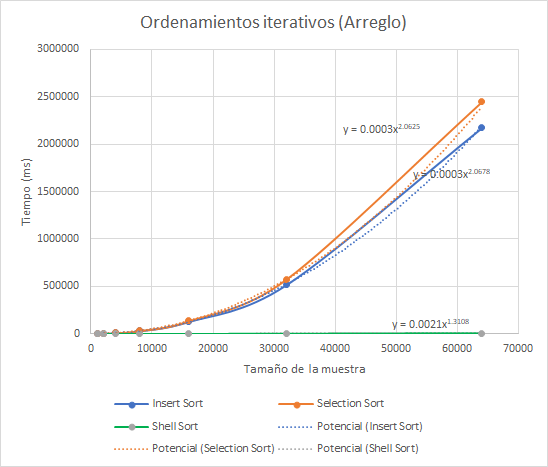
Tabla 3. Comparación de tiempos de ejecución para los ordenamientos iterativos en la representación lista enlazada.

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

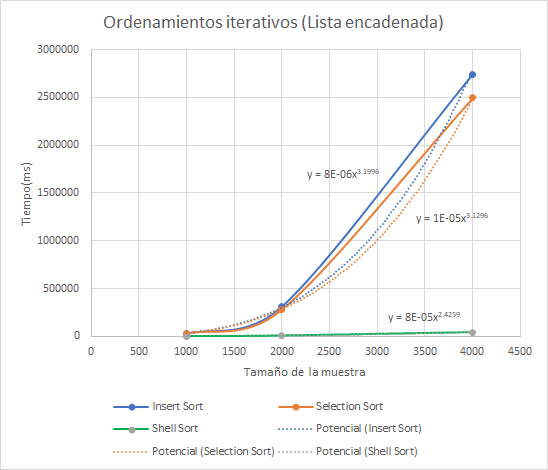
Tabla 4. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.

## **Graficas**

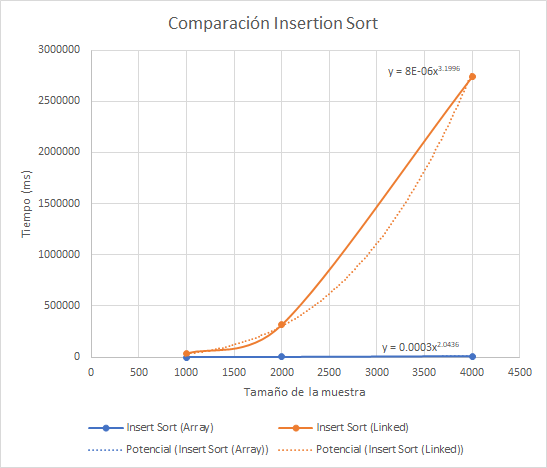
**Comparación de rendimiento ARRAYLIST.**



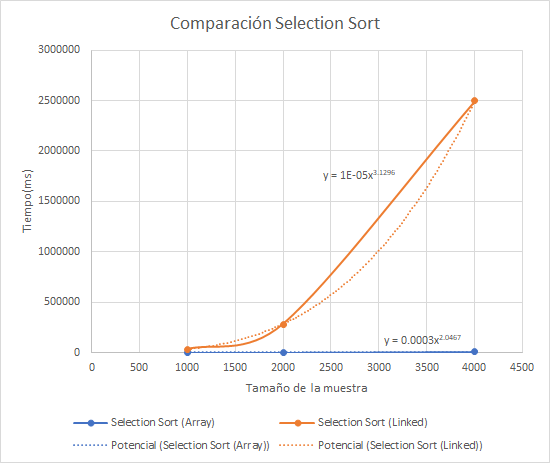
**Comparación de rendimiento LINKED\_LIST.**



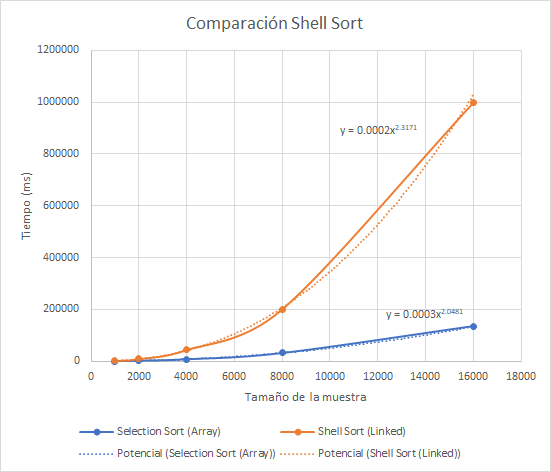
**Comparación de rendimiento para Insertion Sort.**



**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?



En términos generales esta práctica se asemeja a los peores casos teóricos de los ordenamientos iterativos. Esto podría ser debido a que la teoría se desarrolla en un caso matemáticamente ideal, que no toma en cuenta la memoria RAM, su uso dentro de la máquina y el espacio que pueden ocupar otras aplicaciones ejecutándose en segundo plano.

Específicamente, los ordenamientos que más se asemejan al peor caso teórico son los del arreglo. Como podemos ver en las gráficas generadas por los tiempos de la máquina 2, las funciones de Insertion y Selection se acercan al n^2, mientras el shell sort se acerca al orden de n ^(3/2).

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

Sí, casi todos los tiempos registrados por la máquina 2 son triplicados por la máquina 1.

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

Esto se debe tanto a la memoria RAM como al procesador de las máquinas. Un procesador más reciente contiene una mayor cantidad de transistores lo que le permite ejecutar una mayor cantidad de tareas al mismo tiempo. Asimismo, una mayor capacidad de RAM asegura a su vez que los procesos efectuados por los transistores puedan ser comunicados con otros componentes. En otras palabras, el procesador y la RAM tienen efecto el uno sobre el otro y entre más eficientes sean ambos, más eficiente será 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 tiene en cuenta los tiempos de los algoritmos el arreglo es la mejor estructura para utilizar. Esto puede deberse a que cada vez que se realiza un intercambio de elementos para ordenarlos, internamente el computador solo debe intercambiar su lugar en memoria. Mientras que para la lista encadenada, cada que vez que se efectúa este proceso, se deben cambiar los punteros del elemento de “atrás” y de al “frente”.