OBSERVACIONES DE LA PRACTICA

Estudiante 1 Cod 202013371

Estudiante 2 Cod 202022217

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | I5-9300H CPU 2.40GHZ | AMD Ryzen 5 3500u  2.10GHz |
| Memoria RAM (GB) | 16 | 12 |
| Sistema Operativo | Windows 10 | Windows 10 |

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 | 843.75 | 1119.70 | 67.70 |
| 2000 | 3294.62 | 3760.41 | 130.20 |
| 4000 | 16171.83 | 16057.27 | 203.12 |
| 8000 | 57557.28 | 68317.63 | 608.37 |
| 16000 | 236364.33 | 266760.40 | 1260.40 |
| 32000 | Tiempo excedido | Tiempo excedido | 3817.70 |
| 64000 |  |  | 9083.33 |
| 128000 |  |  | 16411.45 |
| 256000 |  |  | 44020.83 |
| 512000 | Excede tamaño | Excede tamaño | Excede tamaño |

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

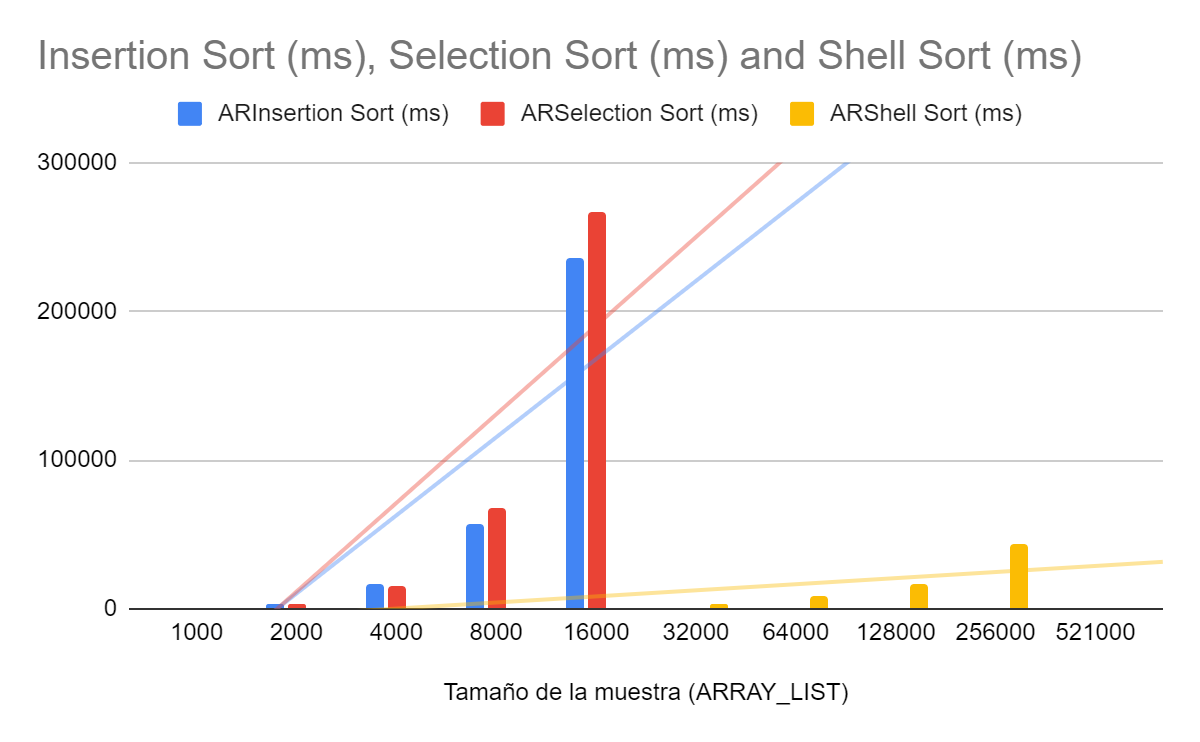
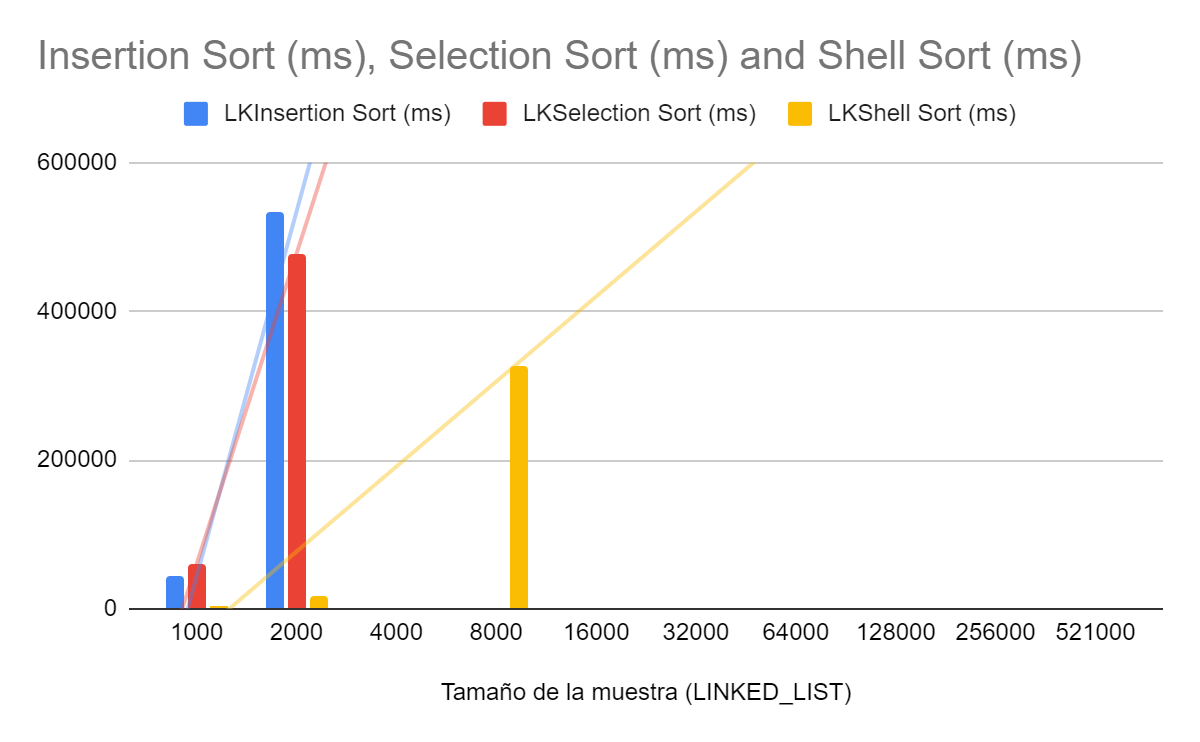
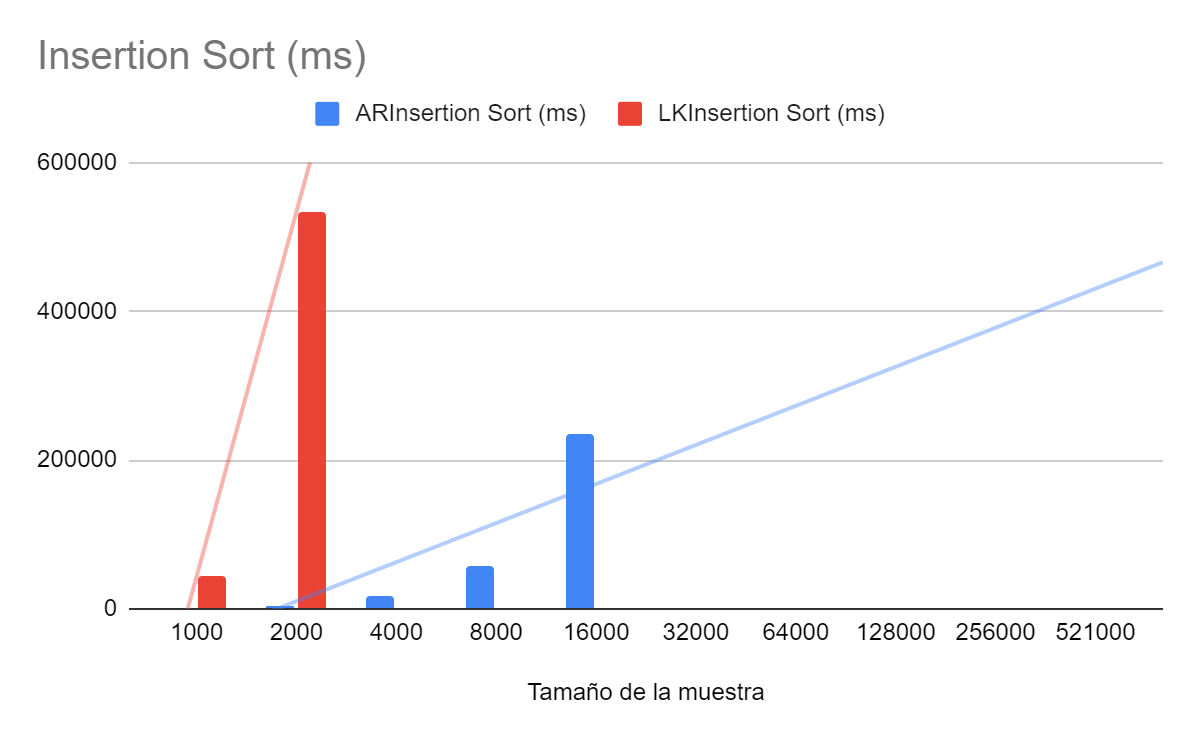
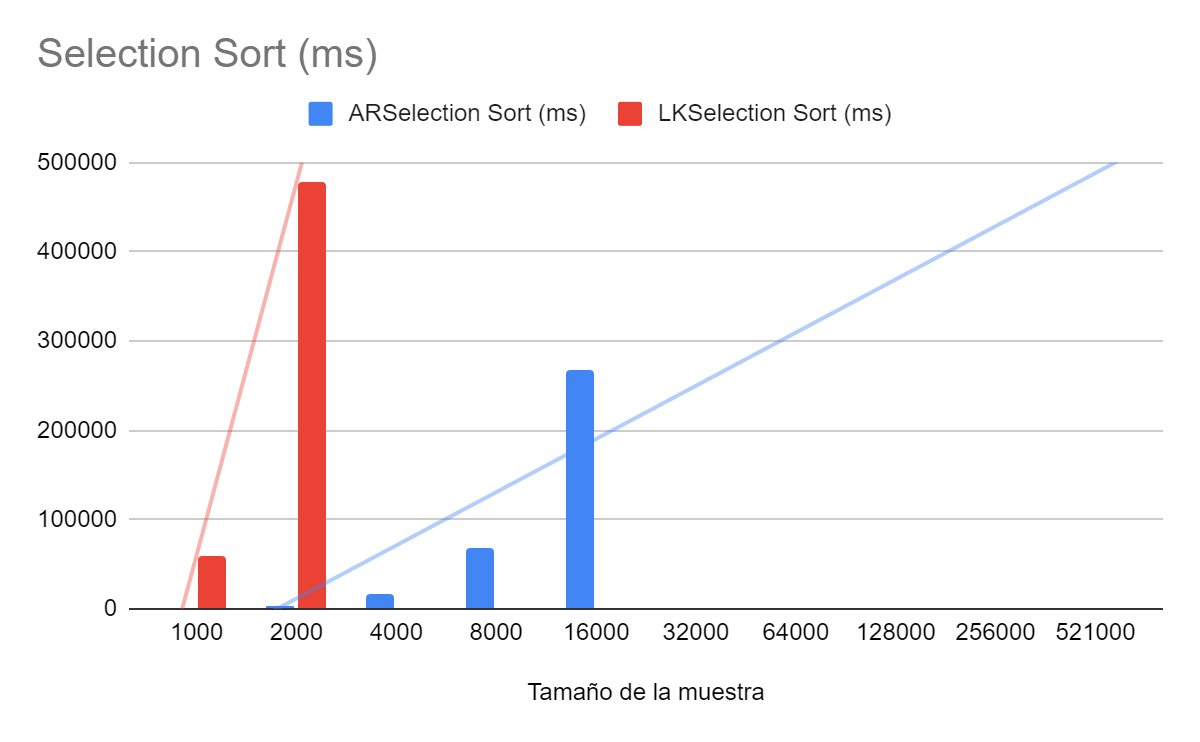
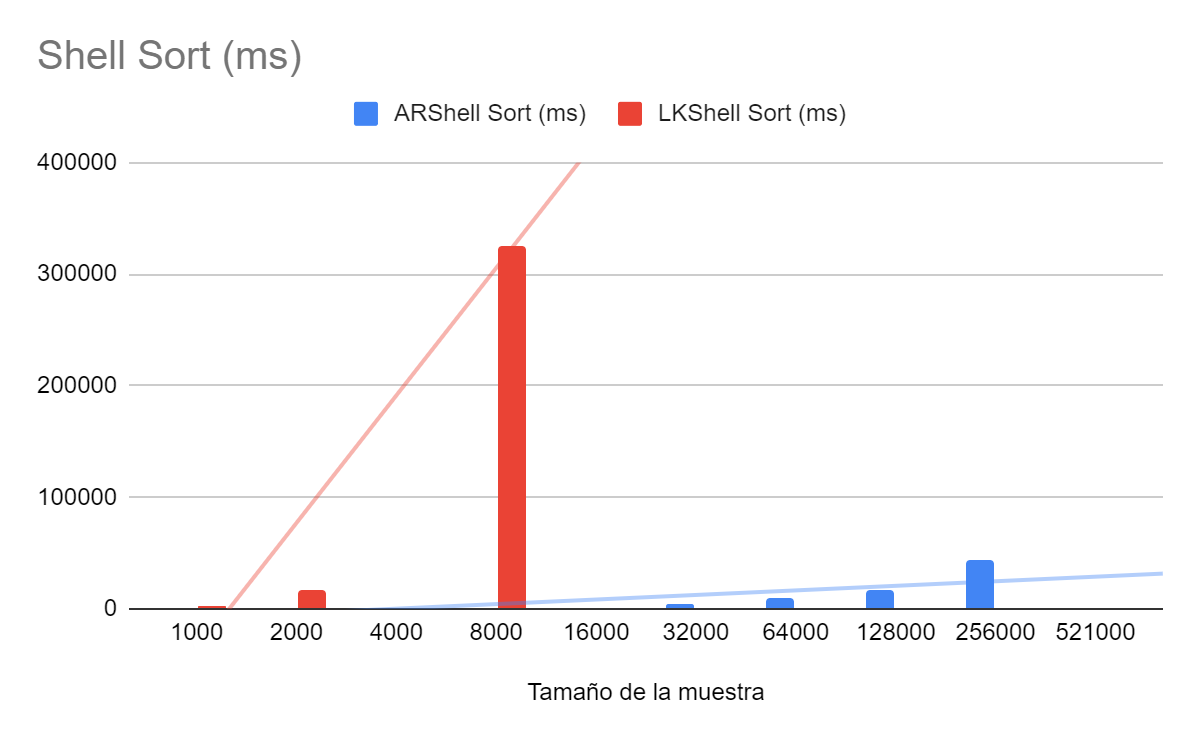
|  |  |  |  |
| --- | --- | --- | --- |
| Tamaño de la muestra (LINKED\_LIST) | Insertion Sort (ms) | Selection Sort (ms) | Shell Sort (ms) |
| 1000 | 43586.45 | 60241.00 | 3291.66 |
| 2000 | 533874.99 | 477401.04 | 16255.20 |
| 4000 | Tiempo excedido | Tiempo excedido | 65258,74 |
| 8000 |  |  | 325853.87 |
| 16000 |  |  | Tiempo Excedido |
| 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 | Ligeramente mejor que selection sort,pero igual poco eficiente | Poca eficiencia,en general igual al selection sort. |
| Selection sort | El peor algoritmo ,para nada eficiente | Poca eficiencia |
| Shell sort | Salto muy grande en la eficiencia(mucho mas eficiente comparado con insertion y selection) | No es eficiente si lo comparamos con su contraparte arraylist pero es el mejor algoritmo comparado con insertion sort y selection 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.
  + 

# **Maquina 2**

## **Resultados**

|  |  |  |  |
| --- | --- | --- | --- |
| Tamaño de la muestra (ARRAY\_LIST) | Insertion Sort (ms) | Selection Sort (ms) | Shell Sort (ms) |
| 1000 | 80046.88 | 69593.75 | 3843.75 |
| 2000 | 643796.88 | 813500.00 | 18218.60 |
| 4000 | Excedió tiempo | Excedió tiempo | 119437.50 |
| 8000 |  |  | 587218.75 |
| 16000 |  |  | Excedió tiempo |
| 32000 |  |  |  |
| 64000 |  |  |  |
| 128000 |  |  |  |
| 256000 |  |  |  |
| 512000 |  |  |  |

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

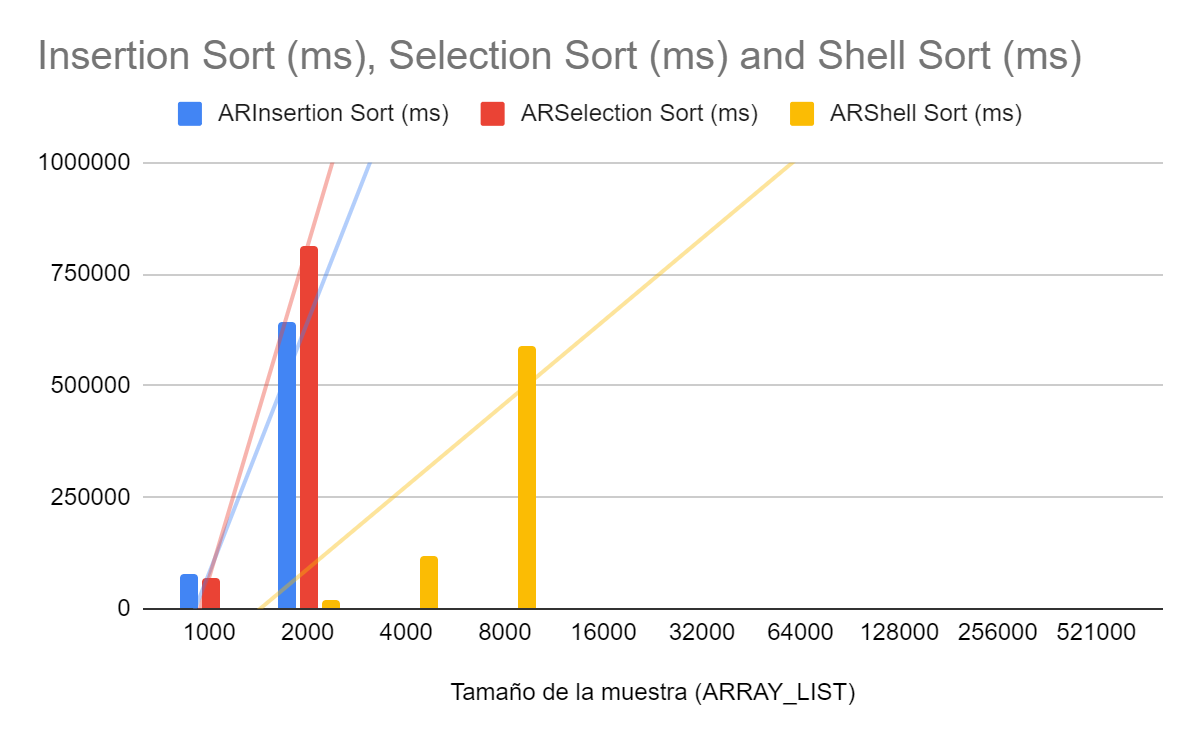
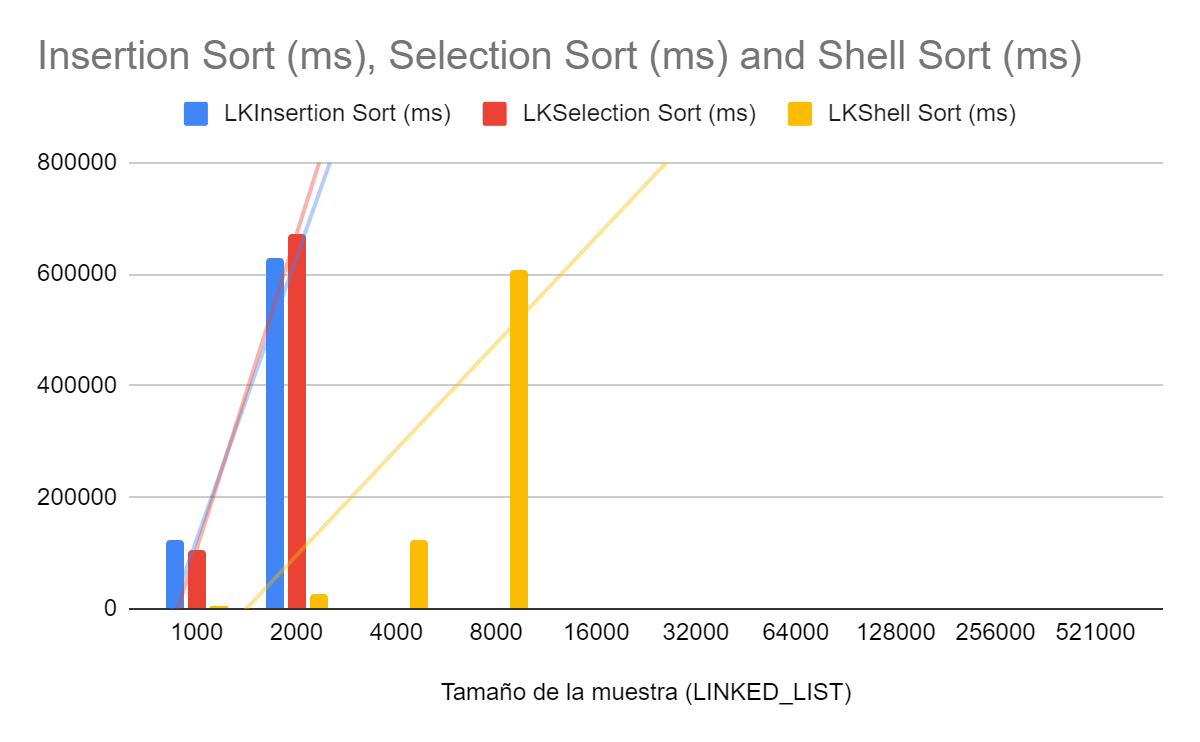
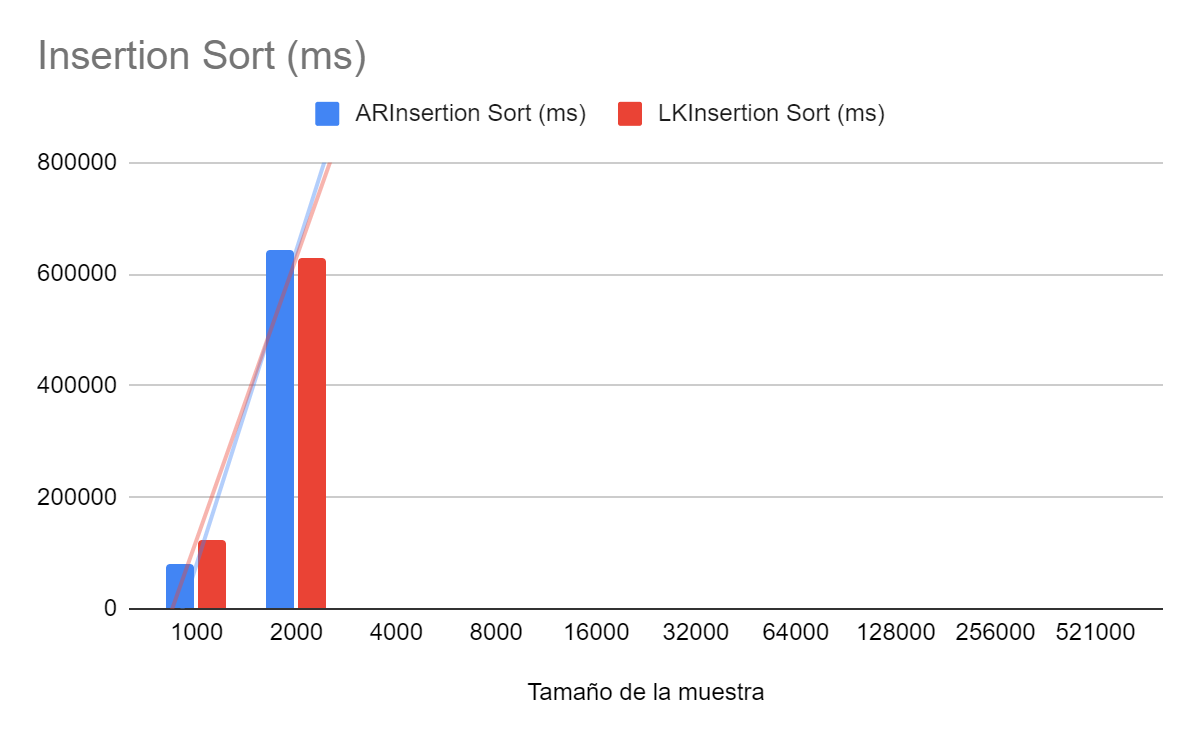
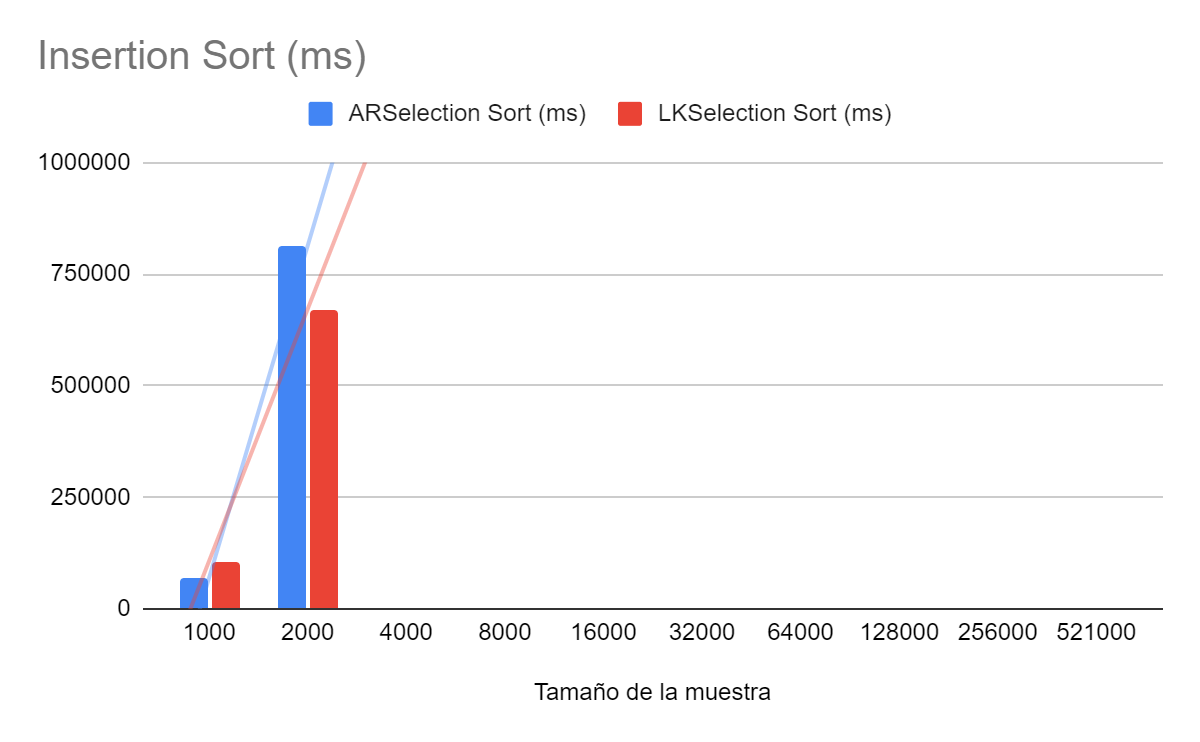
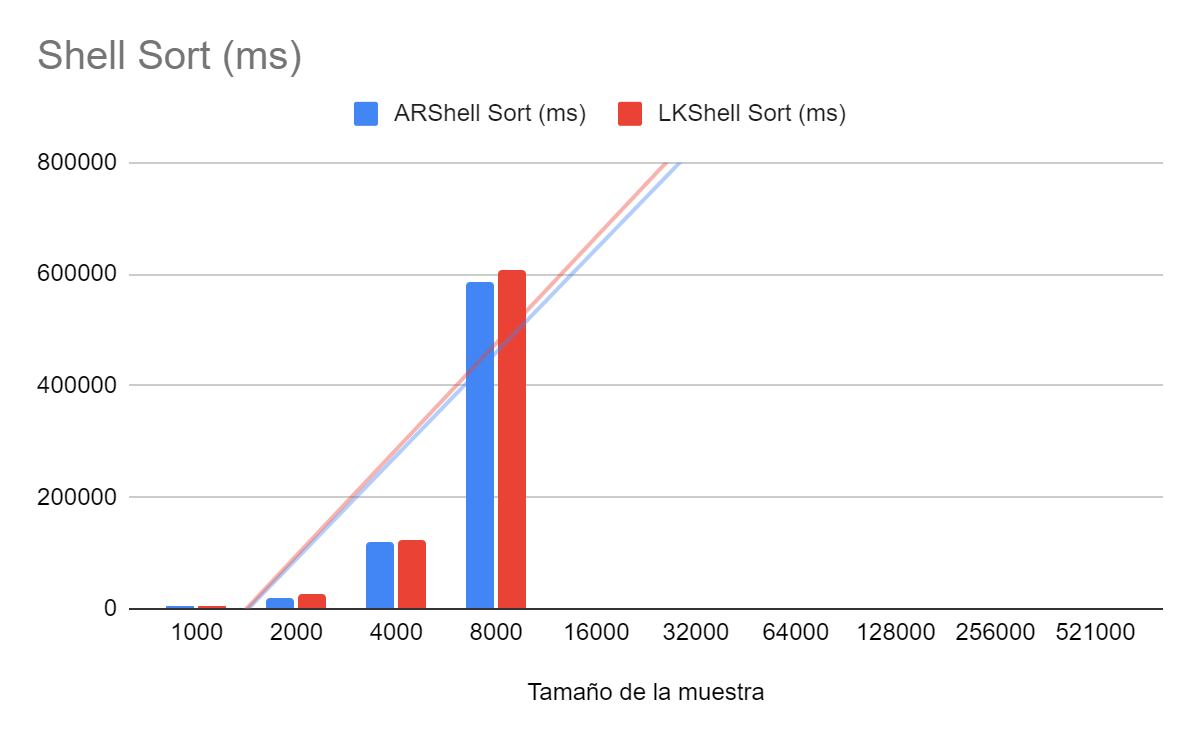
|  |  |  |  |
| --- | --- | --- | --- |
| Tamaño de la muestra (LINKED\_LIST) | Insertion Sort (ms) | Selection Sort (ms) | Shell Sort (ms) |
| 1000 | 123984.375 | 107187.50 | 6031.25 |
| 2000 | 630437.50 | 671703.125 | 27171.875 |
| 4000 | Excedió tiempo | Excedió tiempo | 121812.50 |
| 8000 |  |  | 608593.75 |
| 16000 |  |  | Excedió tiempo |
| 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 en general** | Más eficiente con muestras pequeñas |
| Selection sort | **Mucho más eficiente con muestras grandes** | Más eficiente con muestras pequeñas |
| Shell sort | Levemente menos eficiente en general | **Más eficiente con muestras grandes** |

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?

* La teoría indica que insertion es generalmente superior a selection cuando los datos ya están parcialmente ordenados, así que tiene sentido que los datos, los cuales están repartidos al azar, sean más eficientemente ordenados con selection. Sin embargo, shell sort es significativamente más efectivo gracias a su habilidad de intercambiar valores muy separados, lo cual lo hace sumamente efectivo con muestras de datos extensas.

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

* Si. La máquina 1 resultó ser significativamente más rápida que la 2, especialmente con respecto al Array List, el cual permitió un mayor número de pruebas antes de exceder el tiempo. El Linked List presentó diferencias menos pronunciadas, pero la máquina 1 siguó siendo superior.

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

* Los principales factores que pudieron causar estas diferencias incluyen la diferencia en velocidad de procesamiento de los computadores (2.40 contra 2.10GHz) o los procesos de segundo plano que se pudieron desapercibidamente estar desarrollando.

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, el Array List resultó ser muy superior en la máquina 1, mientras que en la máquina 2 estuvo levemente detrás del Linked List. Sin embargo, la diferencia en la máquina 1 es tan significativa que vale la pena decir que el Array es mejor. Además de esto, la estructura shell sort es bastante superior a las demás cuando se trata de ordenar datos, y por un margen significativo en términos de productividad.