OBSERVACIONES DE LA PRACTICA

Estudiante 1 Cod 202113836

Estudiante 2 Cod 201211558

# **Ambientes de pruebas**

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | I5 9th generación | I5 1.6 GHz |
| Memoria RAM (GB) | 8 GB | 4GB |
| Sistema Operativo | Windows 11 | Window 10 |

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

# **Maquina 1**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 0.50% | 500 | 4687.5 | 484.375 | 593.75 | 359.375 |
| 100.00% | 1000 | 12046.87 | 796.87 | 734.375 | 562.5 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (LINKED\_LIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 0.50% | 500 | 18765.625 | 1250.0 | 1500.0 | 406.25 |
| 100.00% | 1000 | 66062.5 | 3562.5 | 3656.25 | 703.125 |

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

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| *Insertion Sort* | x |  |
| *Shell Sort* | X |  |
| *Merge Sort* | X |  |
| *Quick Sort* | X |  |

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

# **Maquina 2**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 0.50% | 500 | 4703.12 | 296.87 | 343.75 | 234.37 |
| 100.00% | 1000 | 7671.87 | 562.5 | 593.75 | 406.25 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (LINKED\_LIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 0.50% | 500 | 19812.5 | 1234.37 | 1468.75 | 375 |
| 100.00% | 1000 | 666625 | 3546.87 | 3781.25 | 687.5 |

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

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| *Insertion Sort* | x |  |
| *Shell Sort* | x |  |
| *Merge Sort* | x |  |
| *Quick Sort* | x |  |

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

# **Preguntas de análisis**

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

De los algoritmos iterativos el que mejor resultado en el tiempo fue el Insertions Sort, lo cual coincide con lo visto en la teoria, pero este tiempo es aun mejor con el Merge Sort, que emplea metodos recursivos.

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

Si, el resultado de la ejecución es similar, no obstante, varia entre las maquinas

1. De existir diferencias, ¿a qué creen que se deben?

Se puede deber a la memoria RAM que se está utilizando es ese momento, del sistema operativos y de los procesadores.

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

Array list, ya que como se observa en las tablas los tiempos de ejecución en general son mas bajos en esta estructura.

1. Teniendo en cuenta las pruebas de tiempo de ejecución por todos los algoritmos de ordenamiento estudiados (iterativos y recursivos), proponga un ranking de los mismo de mayor eficiencia a menor eficiencia en tiempo para ordenar la mayor cantidad de obras de arte.

1. Merge sort
2. Shell sort
3. Quick sort
4. Insertion sort