OBSERVACIONES DE LA PRACTICA

Diego Acosta Corredor Cod 202110516

Mateo Cote Canal 2 Cod 202022609

# **Ambientes de pruebas**

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | AMD A9-9425 RADEON R5, 5 COMPUTE CORES 2C+3G 3.10 GHz | Apple M1 |
| Memoria RAM (GB) | 8 GB | 8 GB |
| Sistema Operativo | Windows 10 Home Single 64-bits | macOS BigSur |

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% | 50 | 3.125 | 9.375 | 0.00 | 0.00 |
| 50,00% | 5000 | 27027.34 | 281.25 | 979.17 | 260.42 |
| 100.00% | 10000 | 111568.8 | 598.96 | 3895.83 | 572.92 |

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% | 50 | 20.83 | 5.21 | 5.21 | 10.41 |
| 50,00% | 5000 | El tiempo de espera supero lo esperado | El tiempo de espera supero lo esperado |  | 7359.375 |
| 100.00% | 10000 | El tiempo de espera supero lo esperado |  |  | 29859.375 |

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

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| *Insertion Sort* | 46199.76 | 20.83 |
| *Shell Sort* | 296.53 | 5.21 |
| *Merge Sort* | 277.78 | 12409.5 |
| *Quick Sort* | 1625 | 5.21 |

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% | 50 | 3.81 | 1.52 | 1.53 | 2.02 |
| 50,00% | 5000 | 10172.85 | 142.83 | 436.55 | 135.16 |
| 100.00% | 10000 | 40964.86 | 271.94 | 1580.59 | 256.04 |

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% | 50 | 11.27 | 4.31 | 4.51 | 3.2 |
| 50,00% | 5000 | 3155986.15 | 27098.87 | 75122.71 | 2873.54 |
| 100.00% | 10000 | El tiempo de espera supero lo esperado | El tiempo de espera supero lo esperado | El tiempo de espera supero lo esperado | 11723.99 |

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

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| *Insertion Sort* | 17047.17 |  |
| *Shell Sort* | 138.76 |  |
| *Merge Sort* | 131.07 |  |
| *Quick Sort* | 672.89 |  |

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?

El algoritmo con más eficiencia era el MergeSort, tanto teóricamente como en las pruebas realizadas en este laboratorio, este algoritmo realiza los ordenamientos de una manera significativamente más eficiente que los otros algoritmos. Igualmente, el algoritmo que peor se comportaba temporalmente fue el InsertionSort, que como se vio en clase, en las pruebas realizadas tardaba mucho más que los demás algoritmos.

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

Según los datos obtenidos sobre la eficiencia tanto de estructuras cómo de algoritmos, se puede observar que la ejecución de pruebas cambia drásticamente de una máquina a otra. Los datos de la máquina #2 casi en todos los casos tuvieron un tiempo de ejecución menor frente aquellos de la máquina #1. Esto significa que alguna funcionalidad o característica en la máquina #2 diferente a la máquina #1 está haciendo que el tiempo de ejecución de las funciones sea de mayor rapidez.

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

Ya que existen diferencias de procesamiento de algoritmos y de ejecución de funciones entre las dos máquinas, se infiere que también debe existir una diferencia o características clave entre estas que está permitiendo dichos cambios de velocidad en la eficiencia. Observando las características de procesamiento de datos para las máquinas se puede asumir que donde esté presente la diferencia de desempeño, también estará presente la diferencia en cuanto eficiencia, por lo tanto, cómo la máquina #2 presenta un procesador Apple M1 a diferencia de la máquina #1 con un procesador de menor capacidad, la máquina #1 tendrá menor eficiencia tal cómo se pudo concluir acorde a los datos tomados. Por otra parte, las diferencias también pueden darse debido a el sistema operativo ya que una de las máquinas trabaja con mac OS mientras que la otra con windows 10 home 64-bits

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

Según lo enunciado teóricamente, la estructura que mayor eficiencia tenía al momento de realizar ordenamientos era el ArrayList: al tener todos sus elementos almacenados uno tras otro -permitiendo su acceso mucho más rápido-, era mucho más rápido realizar comparaciones entre sus elementos. Asimismo, en las pruebas realizadas (resultados que se registraron en las tablas 3 y 4), esta estructura de datos tardó menos en todos los casos de ordenamientos, así hubiera variado el tipo de algoritmo o el número de datos, el ArrayList se comportó mejor que el SingleLinked.

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.

**RANKING DE MAYOR A MENOR EFICIENCIA:**

* Merge Sort
* Shell Sort
* Quick Sort
* Insertion Sort