OBSERVACIONES DE LA PRACTICA

Estudiante 1: Juan Sebastián Ortega Romero | Cod 202021703

Estudiante 2: Yesid Camilo Almanza | Cod 201921773

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | Procesador Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz | Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz |
| Memoria RAM (GB) | 16GB | 16GB |
| Sistema Operativo | Windows 10 Home 64-bits | Windows 10 Home 64-bits |

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

De acuerdo a lo mencionado en clase magistral, el tiempo de ejecución máximo que se tomará en cuenta será de **15 minutos (900,000 ms)**. En los casos en los que el tiempo de espera máximo fue superado (excediento la capacidad efectiva de procesamiento de nuestros terminales) no se realizó un registro. Estos caso fueron consignados en la tabla como campos NAN.

# **Maquina 1**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 67125.30 | 51562.50 | 2796.88 | 7312.50 | 906.25 |
| 2000 | 438984.38 | 323828.13 | 14078.13 | 35921.87 | 4187.50 |
| 4000 | NAN | NAN | 61109.35 | 158703.13 | 16390.63 |
| 8000 | NAN | NAN | 300484.40 | 802359.40 | 63921.88 |
| 16000 | NAN | NAN | NA | NA | 243906.25 |
| 32000 | NAN | NAN | NA | NA | NA |
| 64000 | NAN | NAN | NA | NA | NA |
| 128000 | NAN | NAN | NA | NA | NA |
| 256000 | NAN | NAN | NAN | NA | NA |
| 512000 | NAN | NAN | NAN | NA | NA |

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]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 60453.13 | 55843.75 | 2734.38 | 2140.63 | 375.50 |
| 2000 | 471734.38 | 444406.25 | 13265.63 | 11109.38 | 1062.50 |
| 4000 | NAN | NAN | 62093.75 | 45531.25 | 4546.88 |
| 8000 | NAN | NAN | 334156.25 | 212578.13 | 16906.25 |
| 16000 | NAN | NAN | NAN | NAN | 75593.75 |
| 32000 | NAN | NAN | NAN | NAN | 288843.30 |
| 64000 | NAN | NAN | NAN | NAN | NAN |
| 128000 | NAN | NAN | NAN | NAN | NAN |
| 256000 | NAN | NAN | NAN | NAN | NAN |
| 512000 | NAN | NAN | NAN | NAN | NAN |

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

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| Merge sort |  |  |
| Quick 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.
  + Comparación de rendimiento para MergeSort.
  + Comparación de rendimiento para QuickSort.

# **Maquina 2**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 30750.30 | 34562.50 | 1828.13 | 1375.30 | 171.88 |
| 2000 | 279062.50 | 246703.13 | 8828.13 | 6187.50 | 656.25 |
| 4000 | 2326093.75 | 2063234.38 | 38625.25 | 25234.38 | 2531.25 |
| 8000 | NAN | NAN | 194500.30 | 133515.63 | 9875.30 |
| 16000 | NAN | NAN | 830093.75 | 500812.50 | 42125.20 |
| 32000 | NAN | NAN | NAN | NAN | 172687.50 |
| 64000 | NAN | NAN | NAN | NAN | NAN |
| 128000 | NAN | NAN | NAN | NAN | NAN |
| 256000 | NAN | NAN | NAN | NAN | NAN |
| 512000 | NAN | NAN | NAN | NAN | NAN |

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]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 30468.75 | 35640.63 | 1734.38 | 1281.25 | 171.875 |
| 2000 | 277578.13 | 249796.88 | 8109.38 | 6218.75 | 656.25 |
| 4000 | 2300375.0 | 2125640.63 | 36843.80 | 25093.75 | 2562.5 |
| 8000 | NAN | NAN | NAN | 146109.38 | 9921.88 |
| 16000 | NAN | NAN | NAN | 499593.75 | 42031.25 |
| 32000 | NAN | NAN | NAN | NAN | 170984.38 |
| 64000 | NAN | NAN | NAN | NAN | NAN |
| 128000 | NAN | NAN | NAN | NAN | NAN |
| 256000 | NAN | NAN | NAN | NAN | NAN |
| 512000 | NAN | NAN | NAN | NAN | NAN |

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

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| Merge sort |  |  |
| Quick 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 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.
  + Comparación de rendimiento para MergeSort.
  + Comparación de rendimiento para QuickSort.

# **Preguntas de análisis**

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

Antes de siquiera iniciar el análisis de las gráficas es necesario resaltar el número máximo de elementos que cada algoritmo fue capaz de organizar en el tiempo máximo de espera (15 minutos). En ambas máquinas se logró observar que en tanto Linked List como Array List, los algoritmos de Shell Sort y Merge Sort fueron los únicos en organizar efectivamente más de 8000 elementos. No obstante, se destacan los resultados del Merge Sort que logró organizar los datos en un tiempo notablemente menor a cualquier otro algoritmo planteado.

En cuanto a las gráficas, es posible comparar los resultados obtenidos experimentalmente con los resultados teóricos de cada algorítmo mediante la evaluación de las lineas de tendencia. En el caso de Selection Sort e Insertion Sort, a pesar de que se tomaron pocos datos, es posible afirmar que estos son los dos algoritmos cuyo tiempo de ejecución crece con mayor rapidez, lo cual está soportado por su complejidad de O(n^2) (observable como tendencia cuadrática en las gráficas). El algoritmo Quick Sort, a pesar de contar con una complejidad temporal de O(n^2) en el peor de los casos, demostró ser más efectivo que sus contrapartes iterativas puesto que de forma experimental logró organizar más datos antes de sobrepasar el límite de tiempo establecido. Por otra parte, los ya mencionados Shell Sort y Merge Sort obtuvieron los mejores resultados, esto está soportado en su complejidad teórica que se aporxima a n\* log2 (n) y por lo tanto provee resultados mucho menores a los vistos en algoritmos con complejidad de O(n2). Sin embargo, Merge Sort cuenta con una superioridad temporal indiscutible para organizar la base de datos presentada.

Las gráficas generales de la máquina 1 y la máquina 2 presentaron bastante similares. Un fenómeno repetido en ambas es la aparición de dos grandes tendencias cuadraticas que hacen referencia a Selection Sort e Insertion Sort y opacan a las tendencias de todos los demás algoritmos. Este hecho evidencia la poca eficacia de estos dos algoritmos además de su inviavilidad para trabajar con grandes bases de datos.

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

Sí, aunque ambas máquinas cuentan con especificaciones técnicas muy similares, la máquina 2 realizó la mayoría de los procesamientos de forma mucho más efectiva que la máquina 1. En este caso, ninguna máquina sufrió errores al momento de procesar los datos, sin embargo, es posible observar en las tablas de datos que la máquina número 2 realizó la mayoría de los procedimientos en la mitad del tiempo que la máquina 1. A pesar de esta diferencia, las tendencias observadas en las gráficas relacionadas a cada máquina fueron bastante similares, demostrando que aunque la capacidad de procesamiento cambie, el tipo algoritmo seguirá incidiendo sustancialmente en el tiempo de ejecución.

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

Las diferencias se deben principalmente a la forma en la que cada máquina tiene organizada y dividida su memoria RAM. La máquina 1 cuenta con un solo stick de memoria RAM de 16GB, por lo que este puede verse afectado por una mayor cola de procesamiento, mientras que la máquina 2 cuenta con 2 sticks de 8GB de RAM cada uno, permitiéndole guardar más información temporal al mismo tiempo. No obstante, estos sticks pueden llenarse más rápido lo que explicaría la aparición de pantallas de error. Otro aspecto para considerar es que la máquina 2 utiliza un SSD, mientras que la máquina 1 tiene a VSCode instalado en un HDD.

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

En el caso de los algoritmos iterativos no se vio una diferencia tan marcada entre la Array List y la Single Linked list puesto que los resultados fueron bastante similares, así que en estos casos no es posible establecer una estructura ganadora con los datos recolectados. En los algoritmos recursivos y en la máquina 1 si es posible notar cierta diferencia entre ambas estructuras de datos, no obstante, estas se dan bajo parametros específicos y no de forma constante. En adición, la máquina 2 contabilizó valores bastante similares para los algoritmos recursivos bajo ambas estructuras de datos. Esto nos indica que en el momento de organizar una lista, la efectividad dependerá más del algoritmo implementado que de la estructura de la lista.

1. Para el caso analizado de ordenamiento de los videos, teniendo en cuenta los resultados de tiempo reportados por todos los algoritmos de ordenamiento estudiados (iterativos y recursivos), proponga un ranking de los algoritmos de ordenamiento (de mayor eficiencia a menor eficiencia en tiempo) para ordenar la mayor cantidad de videos.

**Ranking de los mejores algoritmos de ordenamiento (Orden descendiente)**

1. **Merge Sort**
2. Quick Sort
3. Shell Sort
4. Insertion Sort
5. Selection Sort