OBSERVACIONES DE LA PRACTICA

Estudiante 1: Tomas Angel, Codigo: 202020366

Estudiante 2: Daniel Osorio, Codigo: 202022996

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | 2,3 GHz Intel Core i7 de cuatro núcleos | AMD Ryzen 5 4600H 3.0 GHz |
| Memoria RAM (GB) | 32 GB | 8 GB |
| Sistema Operativo | macOs Catalina Version 10.15.7 | Windows 10 Home 64 bits |

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

# **Maquina 1**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 458,21 | 530,70 | 39,02 | 33,33 | 33,51 |
| 2000 | 1794,89 | 2068,14 | 73,64 | 59,66 | 53,68 |
| 4000 | 7214,92 | 8106,61 | 152,44 | 102,16 | 105,73 |
| 8000 | 31360,21 | 33693,65 | 340,34 | 205,24 | 207,79 |
| 16000 | 123395,26 | 146712,59 | 1696,85 | 419,81 | 418,22 |
| 32000 | 512074,77 | 614969,52 | 4216,33 | 856,29 | 871,06 |
| 64000 | 1710441,67 |  | 10310,33 | 1863,06 | 1806,59 |
| 128000 |  |  | 25492,56 | 4273,62 | 3908,09 |
| 256000 |  |  |  | 9077,16 | 8566,34 |
| 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]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 35207,27 | 30416,33 | 1785,23 | 1515,45 | 178,82 |
| 2000 | 287040,20 | 252845,32 | 7385,52 | 6426,91 | 638,89 |
| 4000 |  |  | 38550,48 | 26206,65 | 2576,07 |
| 8000 |  |  | 197798,90 | 499841,78 | 10671,99 |
| 16000 |  |  |  |  | 43790,09 |
| 32000 |  |  |  |  | 172426,58 |
| 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) |
| Merge sort | Mejor | Peor |
| Quick sort | Mejor | Peor |

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.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

* + Comparación de rendimiento LINKED\_LIST.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

* + Comparación de rendimiento para Insertion Sort.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

* + Comparación de rendimiento para Selection Sort.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

* + Comparación de rendimiento para Shell Sort.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

* + Comparación de rendimiento para MergeSort.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

* + Comparación de rendimiento para QuickSort.

Gráfico

Descripción generada automáticamente

# **Maquina 2**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 432,30 | 500,00 | 20,83 | 20,80 | 19,53 |
| 2000 | 1776,04 | 1989,58 | 46,88 | 41,66 | 41,66 |
| 4000 | 7078,13 | 8031,25 | 114,58 | 78,125 | 83,33 |
| 8000 | 29578,13 | 34536,46 | 291,67 | 182,29 | 187,50 |
| 16000 | 120421,88 | 149916,67 | 708,33 | 401,04 | 406,25 |
| 32000 | 509833,33 | 619421,88 | 1598,96 | 869,78 | 864,58 |
| 64000 |  |  | 4000,00 | 1927,08 | 1848,96 |
| 128000 |  |  | 9802,03 | 4302,08 | 4026,04 |
| 256000 |  |  | 24192,70 | 9281,25 | 8562,50 |
| 512000 |  |  | 39677,08 |  |  |

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 | 32239,58 | 27614,58 | 1651,04 | 1234,38 | 156,25 |
| 2000 | 259765,63 | 221390,63 | 6812,50 | 5687,50 | 604,17 |
| 4000 | 2134781,25 | 1814484,38 | 38015,63 | 22442,70 | 2286,46 |
| 8000 |  |  | 177140,63 | 100343,75 | 9255,04 |
| 16000 |  |  | 837359,37 | 444906,25 | 37953,125 |
| 32000 |  |  |  |  | 158937,50 |
| 64000 |  |  |  |  | 636000,00 |
| 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) |
| Merge sort | Mejor | Peor |
| Quick sort | Mejor | Peor |

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.

Imagen que contiene Gráfico de líneas

Descripción generada automáticamente

* + Comparación de rendimiento LINKED\_LIST.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

* + Comparación de rendimiento para Insertion Sort.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

* + Comparación de rendimiento para Selection Sort.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

* + Comparación de rendimiento para Shell Sort.

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

* + Comparación de rendimiento para MergeSort.

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

* + Comparación de rendimiento para QuickSort.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

# **Preguntas de análisis**

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

Basandonos solamente en los resultados nuevos obtenidos en este laboratorio, podemos decir que el comportamiento de los algoritmos si es acorde a lo enunciado teoricamente puesto que merge sort y quick sort tienen un tiempo de ejecucion de O(n log(n)) en todos los casos cuando se tiene los datos en un arreglo, mientras que al tenerlos ordenados en una lista enlazada el tiempo de ejecucion de estos es de O(n^2).

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

Si, al ejecutar las preubas en diferentes maquinas, se pudo ver que la maquina 2 corria los datos en un menor tiempo un poco menor cuando lo comparabamos a la maquina 1.

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

Las diferencias se deben dar a las caracteristicas especificas de cada uno de los computadores; la cantidad de nucleos que cada procesador tiene, y la frecuencia a la que este mismo trabaja. Asi mismo, puede que otras hubieran aplicaciones corriendo de fondo en el computador de las cuales no sabiamos, ocupando una parte de la memoria y el procesador.

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

La mejor de estructura de datos que pudimos ver al realizar las pruebas, fue el de Merge sort cuando los datos estan ordenados en un arreglo.

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.

Haciendo un ranking de los algoritmos de ordenamiento (iterativos o recursivos), de mayor a menor eficiencia, se pueden encontrar los siguiente:

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