OBSERVACIONES DE LA PRACTICA

Estudiante 1: José Vicente Vargas Panesso Cod: 201815601

Estudiante 2: Daniel Reales Cod 201822265

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | Intel core i7-7700HQ | Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, 1800 Mhz, 4 Core(s), 8 Logical Processor(s) |
| Memoria RAM (GB) | 16.00 | 8.00 |
| 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) | Quick Sort (ms) | Merge Sort (ms) |
| 1000 | 524.21 | 634.35 | 27.59 | 21.80 | 25.76 |
| 2000 | 2217.25 | 2493.50 | 70.30 | 69.55 | 50.74 |
| 4000 | 9220.95 | 9905.98 | 158.00 | 128.60 | 122.00 |
| 8000 | 36895.14 | 40755.87 | 359.71 | 277.64 | 235.92 |
| 16000 | 147906.40 | 170314.96 | 888.40 | 620.95 | 540.09 |
| 32000 | 624198.79 | 716931.52 | 2195.29 | 1317.26 | 1149.28 |
| 64000 | Más de 10 minutos. | Más de 10 minutos. | 4896.47 |  |  |
| 128000 | Más de 10 minutos. | Más de 10 minutos. | 10496.86 |  |  |
| 256000 | Más de 10 minutos. | Más de 10 minutos. | 25239.34 |  |  |

Tabla 2. Comparación de tiempos de ejecución para los ordenamientos iterativos y recursivos 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 | 39418.77 | 37099.99 | 1836.10 | 1693.86 | 205.86 |
| 2000 | 325156.76 | 299347.40 | 9294.74 | 9288.13 | 1036.42 |
| 4000 | Más de 10 minutos | Más de 10 minutos | 45590.37 | 38435.16 | 3493.09 |
| 8000 | Más de 10 minutos | Más de 10 minutos | 209505.94 | 163650.96 | 13559.31 |
| 16000 | Más de 10 minutos | Más de 10 minutos | 1033159.49 | 805650.29 | 55158.93 |
| 32000 | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos |  | 228388.40 |
| 64000 | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos |  |  |
| 128000 | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos |  |  |
| 256000 | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos |  |  |
| 512000 | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos |  |  |

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 implementación | Peor implementación |
| Quick sort | Mejor implementación | Peor implementación |

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 (ARRAY\_LIST) | Insertion Sort (ms) | Selection Sort (ms) | Shell Sort (ms) | Quick Sort (ms) | Merge Sort (ms) |
| 1000 | 520.64 | 675.22 | 25.45 | 22.94 | 21.44 |
| 2000 | 2086.95 | 2479.41 | 85.77 | 60.34 | 44.89 |
| 4000 | 9099.18 | 10598.21 | 160.09 | 105.25 | 100.73 |
| 8000 | 40956.55 | 42120.89 | 471.75 | 271.34 | 300.24 |
| 16000 | 152000.00 | 180500.00 | 871.20 | 550.05 | 527.08 |
| 32000 | 639500.00 | 777500.00 | 1957.80 | 1078.66 | 1089.60 |
| 64000 | Más de 10 minutos. | Más de 10 minutos. | 4877.96 | 2716.77 | 2612.01 |
| 128000 | Más de 10 minutos. | Más de 10 minutos. | 10593.71 | 7902.01 | 4752.34 |
| 256000 | Más de 10 minutos. | Más de 10 minutos. | 25283.89 |  |  |
| 512000 |  |  |  |  |  |

Tabla 2. Comparación de tiempos de ejecución para los ordenamientos iterativos y recursivos 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 | 47528.87 | 35438.24 | 2623.50 | 1364.38 | 197.95 |
| 2000 | 326460.50 | 322686.50 | 9726.01 | 7541.85 | 1014.82 |
| 4000 | Más de 10 minutos | Más de 10 minutos | 44297.59 | 32028.32 | 3813.77 |
| 8000 | Más de 10 minutos | Más de 10 minutos | 207811.50 | 141743.99 | 14710.72 |
| 16000 | Más de 10 minutos | Más de 10 minutos | 1097035.00 | 613134.03 | 53300.46 |
| 32000 | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos | 226746.81 |
| 64000 | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos |  |
| 128000 | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos |  |
| 256000 | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos |  |
| 512000 | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos | Más de 10 minutos |  |

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 implementación | Peor implementación |
| Quick sort | Mejor implementación | Peor implementación |

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** 
  + 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?

De forma general podemos ver que los algoritmos si se comportan de la manera esperada teóricamente. En el algoritmo Selection Sort, cuya complejidad temporal es se puede ver un aumento cuadrático en los tiempos de ejecución a medida que aumenta el tamaño de los datos, para ambos tipos de datos, tanto ARRAY\_LIST como LINKED\_LIST. Esto podemos verificarlo utilizando un ajuste cuadrático a la gráfica que compara el algoritmo y su tiempo de ejecución, en la implementación con la estructura de datos ARRAY\_LIST como se muestra a continuación:

Como se aprecia en el gráfico, el modelo matemático describe con un el comportamiento temporal en milisegundos del algoritmo implementado bajos las condiciones particulares de la Maquina 2 al momento de tomar los datos.

De manera similar, el algoritmo Insertion Sort (implementado en la Maquina 2 mediante una estructura de datos tipo ARRAY\_LIST) aunque mejor relativo al algoritmo Selection Sort, sigue presentando un comportamiento cuadrático. Este es el resultado esperado por la teoría debido a que la complejidad de este es tanto en el peor caso, así como en el promedio. Realizando un ejercicio similar al anterior, vemos que una línea de tendencia polinómica de grado 2 ajusta los datos de una manera cercana bajo el modelo matemático

.

La mejora temporal relativa al algoritmo Selection Sort se puede apreciar también mediante el análisis del coeficiente que acompaña al término cuadrático. En el caso del algoritmo Insertion Sort, este coeficiente es menor vis-à-vis el coeficiente del modelo matemático que describe al algoritmo Selection Sort.

Para el algoritmo Shell Sort, se observa un comportamiento considerablemente mejor en comparación a los dos desempeños analizados anteriormente. Esto es de esperar desde la perspectiva teórica debido a que las complejidades en el peor caso, caso promedio y mejor caso son de , y respectivamente. Realizando un análisis gráfico apoyado en las líneas de tendencia, podemos ajustar los datos recolectados mediante un ajuste cuadrático para obtener el modelo matemático . Aunque teóricamente la complejidad de este algoritmo es menor a , el programa Excel no permite realizar ajustes con exponentes fraccionales. Por esta razón, con un de ajustamos mediante un polinomio de grado 2 como una **aproximación.** Sin embargo, de ser posible, se debe ajustar un modelo que represente las complejidades teóricas.

Para Merge Sort, el primero de los algoritmos iterativos, utilizados en este laboratorio, vemos que el crecimiento temporal es ligeramente superior al lineal. Este hecho lo confirmamos al realizar una gráfica con línea de tendencia lineal como se muestra a continuación:

Debido a que **no es posible realizar un ajuste linearitmico en Excel** se considera el ajuste más cercano y apropiado el lineal. El modelo resultante de este ejercicio de ajuste es la siguiente ecuación del tiempo en milisegundos en función del número de datos a organizar: En este resultado vemos que el crecimiento temporal teórico se aproxima de manera cercana al evidenciado experimentalmente. Sin embargo, como es de esperar, el comportamiento temporal experimental es mayor que la línea de tendencia debido a que .

Finalmente, Quick Sort, el segundo algoritmo iterativo utilizado en el laboratorio presenta complejidades teóricas de y para el peor, caso promedio y mejor caso, respectivamente. Para validar la concordancia entre los datos recolectados y esta observación teórica se realizarán dos ajustes: 1) un ajuste lineal debido a que es el más cercano a la complejidad y 2) un ajuste cuadrático para modelar el peor caso.

Para el ajuste lineal tenemos el siguiente resultado:

El modelo matemático resultante ajusta los datos con un e indica que por un incremento de 1 elemento en el conjunto de datos, el tiempo en milisegundos que toma el algoritmo incrementa en 852.79, en promedio. Para el ajuste cuadrático tenemos el resultado:

El modelo matemático que describe este segundo ajuste es . Mas aún, la linea de tendencia tiene un de por lo que vemos un mejor ajuste en este segundo caso. Esto puede interpretarse que para los datos estudiados, se está más cerca del peor caso que del mejor.

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

Si, existen diferencias menores en los tiempos de ejecución de los algoritmos para las distintas máquinas. En general, la Maquina 2 toma un mayor tiempo para realizar el ordenamiento de los datos independientemente del tamaño de la lista, el algoritmo utilizado o el tipo de Estructura de Datos seleccionada.

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

Las diferencias entre los desempeños observados pueden ser atribuidad principalmente a dos factores. En primer lugar, el hardware que ambas maquinas utilizan para realizar los calculos y el amacenamiento es distinto. Esto lo podemos evidenciar debido a que la Máquina 1 posee un procesador Intel i7 mientras que la Máquina 2 tiene un procesador i5. Más aún, la capacidad de memoria principal (RAM), en la primera máquina es el doble que en la otra. El segundo conjunto de factores que pueden contribuir a las diferencias observadas en rendimiento son las condiciones particulares de ejecución. Es decir, incluso si las Máquinas fueran identicas en el hardware, si existen diferencias en la cantidad de software que estan ejecutando al momento de realizar los ordenamientos, entonces se observaran desempeños diferentes.

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

La mejor estructura para todos los algoritmos es la estructura ARRAY\_LIST. Esto se debe al funcionamiento interno de cada uno de los algoritmos. Ya que todos utilizan la función getElement(), disponible en la API del TAD Lista, la complejidad de esta instrucción cuando se utiliza un ARRAY \_LIST es O(1) mientras que para la SINGLE\_LINKED es O(n). Esto implica que al evaluar la complejidad del algoritmo no es necesario tener solo en cuenta su complejidad intrínseca, sino también la complejidad de esta función, la cual es claramente superior en una estructura de datos tipo ARRAY\_LIST.

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.

Antes de determinar el ranking de los algoritmos es importante resaltar que el orden relativo de los mismos no cambia así se implementen los datos utilizando un ARRAY\_LIST o un LINKED\_LIST. Esta afirmación puede ser validada por las siguientes dos gráficas. Como se observa, el orden relativo se conserva.

Ahora bien, el ranking que se sugiere de mejor a peor (siendo 1 el mejor y 5 el peor) es:

|  |  |
| --- | --- |
| RANKING | ALGORITMO |
| 1 | Merge Sort |
| 2 | Quick Sort |
| 3 | Shell Sort |
| 4 | Insertion Sort |
| 5 | Selection Sort |