OBSERVACIONES DE LA PRACTICA

Tomás Bedoya Calixto 202020689 (Máquina 1)

Andrés Felipe Gómez García 202021189 (Máquina 2)

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | Intel Core i7-5820K CPU @ 3.30GHz 3.30 GHz | Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz 2.50 GHz |
| Memoria RAM (GB) | 16.0 GB | 12.0 GB |
| Sistema Operativo | Windows 10 Pro | Windows 10 Home Single |

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

# **Máquina 1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tamaño de la muestra (ARRAY\_LIST) | Insertion Sort Promedio (ms) | Selection Sort Promedio (ms) | Shell Sort Promedio (ms) | Merge Sort  (ms) | Quick Sort  (ms) |
| 1000 | 578.13 | 656.25 | 46.88 | 31.25 | 15.63 |
| 2000 | 2265.63 | 2531.25 | 78.13 | 62.5 | 46.88 |
| 4000 | 9234.38 | 10421.88 | 156.25 | 109.38 | 125.0 |
| 8000 | 36359.38 | 40937.5 | 390.63 | 250.0 | 296.88 |
| 16000 | 150828.125 | 193890.63 | 875.0 | 531.25 | 531.25 |
| 32000 | 621062.5 | 747828.125 | 2062.5 | 1125.0 | 1140.63 |
| 64000 | Tiempo excesivo | Tiempo excesivo | 4937.5 | 2390.63 | 2375.0 |
| 128000 | Tiempo excesivo | Tiempo excesivo | 12531.25 | 5140.625 | 5078.13 |
| 256000 | Tiempo excesivo | Tiempo excesivo | 32359.375 | 11015.625 | 11312.5 |
| 512000 | Mayor al número de elementos | Mayor al número de elementos | Mayor al número de elementos | Mayor al número de elementos | Mayor al número de elementos |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tamaño de la muestra (LINKED\_LIST) | Insertion Sort Promedio (ms) | Selection Sort Promedio (ms) | Shell Sort Promedio (ms) | Merge Sort  (ms) | Quick Sort  (ms) |
| 1000 | 41750.0 | 39750.0 | 2109.38 | 218.75 | 1593.75 |
| 2000 | 360468.75 | 327171.88 | 9953.125 | 843.75 | 7781.25 |
| 4000 | Tiempo excesivo | Tiempo excesivo | 47140.625 | 3281.25 | 32937.5 |
| 8000 | Tiempo excesivo | Tiempo excesivo | 254875.0 | 13031.25 | 192500.0 |
| 16000 | Tiempo excesivo | Tiempo excesivo | 1023453.125 | 56890.625 | 696656.25 |
| 32000 | Tiempo excesivo | Tiempo excesivo | Tiempo excesivo | 222406.25 | Tiempo excesivo |
| 64000 | Tiempo excesivo | Tiempo excesivo | Tiempo excesivo | 845953.125 | Tiempo excesivo |
| 128000 | Tiempo excesivo | Tiempo excesivo | Tiempo excesivo | Tiempo excesivo | Tiempo excesivo |
| 256000 | Tiempo excesivo | Tiempo excesivo | Tiempo excesivo | Tiempo excesivo | Tiempo excesivo |
| 512000 | Mayor al número de elementos | Mayor al número de elementos | Mayor al número de elementos | Mayor al número de elementos | Mayor al número de elementos |

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

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| Insertion sort | 4 Más rápido que Linked | 5 |
| Selection sort | 5 Más rápido que Linked | 4 |
| Shell sort | 3 Más rápido que Linked | 3 |
| Merge sort | 1 Más rápido que Linked | 1 |
| Quick sort | 1 Más rápido que Linked | 2 |

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 Quick Sort:
* Comparación de rendimiento para Merge Sort:

# **Máquina 2**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tamaño de la muestra (ARRAY\_LIST) | Insertion Sort Promedio (ms) | Selection Sort Promedio (ms) | Shell Sort Promedio (ms) | Merge Sort (ms) | Quick Sort (ms) |
| 1000 | 744.79 | 916.67 | 41.67 | 31.25 | 31.25 |
| 2000 | 3010.42 | 4041.67 | 98.96 | 93.75 | 78.13 |
| 4000 | 12593.75 | 15604.17 | 218.75 | 140.63 | 145.84 |
| 8000 | 54135.42 | 65552.08 | 614.59 | 296.88 | 364.59 |
| 16000 | 245598.96 | 310140.63 | 1250.00 | 640.63 | 692.71 |
| 32000 | Excede los 15 min | Excede los 15 min | 3000.00 | 1484.38 | 1536.46 |
| 64000 | Excede los 15 min | Excede los 15 min | 7098.96 | 2953.13 | 3119.80 |
| 128000 | Excede los 15 min | Excede los 15 min | 18109.38 | 7000.00 | 6750.00 |
| 256000 | Excede los 15 min | Excede los 15 min | 44161.46 | 13703.13 | 15927.09 |
| 512000 | Mayor al número de elementos | Mayor al número de elementos | Mayor al número de elementos | Mayor al número de elementos | Mayor al número de elementos |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tamaño de la muestra (LINKED\_LIST) | Insertion Sort Promedio (ms) | Selection Sort Promedio (ms) | Shell Sort Promedio (ms) | Merge Sort (ms) | Quick Sort (ms) |
| 1000 | 61005.21 | 53660.42 | 2989.58 | 265.63 | 1921.88 |
| 2000 | 581354.17 | 464260.42 | 13776.04 | 1015.63 | 9531.25 |
| 4000 | Excede los 15 min | Excede los 15 min | 65557.29 | 3968.75 | 45093.75 |
| 8000 | Excede los 15 min | Excede los 15 min | 358901.05 | 16593.75 | 341953.13 |
| 16000 | Excede los 15 min | Excede los 15 min | Excede los 15 min | 65000.00 | Excede los 15 min |
| 32000 | Excede los 15 min | Excede los 15 min | Excede los 15 min | 348140.63 | Excede los 15 min |
| 64000 | Excede los 15 min | Excede los 15 min | Excede los 15 min | Excede los 15 min | Excede los 15 min |
| 128000 | Excede los 15 min | Excede los 15 min | Excede los 15 min | Excede los 15 min | Excede los 15 min |
| 256000 | Excede los 15 min | Excede los 15 min | Excede los 15 min | Excede los 15 min | Excede los 15 min |
| 512000 | Mayor al número de elementos | Mayor al número de elementos | Mayor al número de elementos | Mayor al número de elementos | Mayor al número de elementos |

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

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| Insertion sort | 4 Más rápido que Linked | 5 |
| Selection sort | 5 Más rápido que Linked | 4 |
| Shell sort | 3 Más rápido que Linked | 3 |
| Merge Sort | 1 Más rápido que Linked | 1 |
| Quick Sort | 1 Más rápido que Linked | 2 |

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 Quick Sort
* Comparación de rendimiento para Merge Sort

# **Preguntas de análisis**

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

En el sentido de ciertos algoritmos siendo más rápidos que otros, el comportamiento graficado, tanto en la maquina 1 como en la maquina 2, efectivamente es congruente con lo enunciado teóricamente. Se observa fácilmente que los algoritmos de ordenamiento recursivos son considerablemente más rápidos que los iterativos, y la diferencia entre ambos en el array list es casi imperceptible. Por su parte el shell sort fue el algoritmo de ordenamiento iterativo más rápido en las dos estructuras de datos, seguido del insertion y, finalmente, del selection en el array\_list, es decir la estructura de datos para la que fueron diseñados estos algoritmos. No obstante, vale la pena mencionar que, al ejecutarlos en el linked\_list, el selection finalizó la organización de forma más rápida que el insertion y el merge sort fue de considerable mejor eficiencia que el quick sort. Adicionalmente, se evidencia que todos los algoritmos de ordenamiento fueron mucho más eficientes al ser utilizados con el array\_list que cuando se utilizaron con la linked\_list, ateniéndose una vez más al razonamiento teórico.

Por otro lado, teniendo en cuenta los órdenes de crecimiento temporales que, en teoría, describen los tiempos de ejecución de cada algoritmo, se evidencia que lo presentado en las gráficas es semejante a dichos ordenes de complejidad en las ejecuciones del array\_list. Por ejemplo, podemos ver que el insertion, con complejidad O(n2), asemeja una función cuadrática al ser graficado, y los algoritmos merege sort y quick sort, ambos de complejidad nlog(n), tienen duraciones de gran similitud en las array lists.

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

Comparativamente hablando se observa el mismo patrón de eficiencia en ambas máquinas. Es decir que los ordenamientos recursivos fueron de mayor rapidez, que merge sort es más eficiente en una lista enlazada que quick sort, y que shell sort continúa siendo el algoritmo iterativo más rapido, aunque insertion es el segundo más rápido en el caso del array\_list. Las discrepancias apreciables corresponden a diferencias en el tiempo neto de ejecución debido a las diferencias entre los componentes de los computadores.

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

Como se mencionó anteriormente, es manifiesto que las diferencias son debidas a los diversos componentes que conforma cada máquina de computo. Esto puesto que la máquina uno es un maquinón concebido por las deidades, mientras que la segunda podría reemplazarse por una waflera y la diferencia sería negligible.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| Merge Sort | 1 Más rápido que Linked | 1 |
| Quick Sort | 1 Más rápido que Linked | 2 |

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

Es mejor utilizar el array\_list porque todos los algoritmos de ordenamiento son más eficientes en términos de tiempo de ejecución. Esto se observa en los comentarios que se plasmaron en la tabla 4.

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.

Tomando en cuenta que en la lista por arreglos el merge sort y el quick sort tienen similar eficiencia, pero en la lista enlazada el merge sort es considerablemente superior, para casos generales nuestro ranking sería:

Merge sort

Quick sort

Shell Sort

Insertion Sort

Selection Sort