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**

## **Resultados**

|  |  |  |  |
| --- | --- | --- | --- |
| Tamaño de la muestra (ARRAY\_LIST) | Insertion Sort Promedio (ms) | Selection Sort Promedio (ms) | Shell Sort Promedio (ms) |
| 1000 | 578.13 | 656.25 | 46.88 |
| 2000 | 2265.63 | 2531.25 | 78.13 |
| 4000 | 9234.38 | 10421.88 | 156.25 |
| 8000 | 36359.38 | 40937.5 | 390.63 |
| 16000 | 150828.125 | 193890.63 | 875.0 |
| 32000 | 621062.5 | 747828.125 | 2062.5 |
| 64000 | Tiempo excesivo | Tiempo excesivo | 4937.5 |
| 128000 | Tiempo excesivo | Tiempo excesivo | 12531.25 |
| 256000 | Tiempo excesivo | Tiempo excesivo | 32359.375 |
| 512000 | 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) |
| 1000 | 41750.0 | 39750.0 | 2109.38 |
| 2000 | 360468.75 | 327171.88 | 9953.125 |
| 4000 | Tiempo excesivo | Tiempo excesivo | 47140.625 |
| 8000 | Tiempo excesivo | Tiempo excesivo | 254875.0 |
| 16000 | Tiempo excesivo | Tiempo excesivo | 1023453.125 |
| 32000 | Tiempo excesivo | Tiempo excesivo | Tiempo excesivo |
| 64000 | Tiempo excesivo | Tiempo excesivo | Tiempo excesivo |
| 128000 | Tiempo excesivo | Tiempo excesivo | Tiempo excesivo |
| 256000 | Tiempo excesivo | Tiempo excesivo | Tiempo excesivo |
| 512000 | 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 | 2 Más rápido que Linked | 3 |
| Selection sort | 3 Más rápido que Linked | 2 |
| Shell sort | 1 Más rápido que Linked | 1 |

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.

# **Máquina 2**

## **Resultados**

|  |  |  |  |
| --- | --- | --- | --- |
| Tamaño de la muestra (ARRAY\_LIST) | Insertion Sort Promedio (ms) | Selection Sort Promedio (ms) | Shell Sort Promedio (ms) |
| 1000 | 744.79 | 916.67 | 41.67 |
| 2000 | 3010.42 | 4041.67 | 98.96 |
| 4000 | 12593.75 | 15604.17 | 218.75 |
| 8000 | 54135.42 | 65552.08 | 614.59 |
| 16000 | 245598.96 | 310140.63 | 1250.00 |
| 32000 | Excede los 15 min | Excede los 15 min | 3000.00 |
| 64000 | Excede los 15 min | Excede los 15 min | 7098.96 |
| 128000 | Excede los 15 min | Excede los 15 min | 18109.38 |
| 256000 | Excede los 15 min | Excede los 15 min | 44161.46 |
| 512000 | 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) |
| 1000 | 61005.21 | 53660.42 | 2989.58 |
| 2000 | 581354.17 | 464260.42 | 13776.04 |
| 4000 | Excede los 15 min | Excede los 15 min | 65557.29 |
| 8000 | Excede los 15 min | Excede los 15 min | 358901.05 |
| 16000 | Excede los 15 min | Excede los 15 min | Excede los 15 min |
| 32000 | Excede los 15 min | Excede los 15 min | Excede los 15 min |
| 64000 | 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 |
| 256000 | 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 |

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 | 2 Más rápido que Linked | 3 |
| Selection sort | 3 Más rápido que Linked | 2 |
| Shell sort | 1 Más rápido que Linked | 1 |

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.

# **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 el shell sort fue el algoritmo de ordenamiento 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. Adicionalmente, se evidencia que estos tres tipos de algoritmos de ordenamiento fueron mucho mas 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 ordenes 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.

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 el shell sort continúa siendo el algoritmo más rápido en ambos casos, y el 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.

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

Como se ha discutido en varias de las preguntas previas, el shell sort es supremamente mas eficiente que el resto de los algoritmos de ordenamiento tratados.

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 mas eficientes, en términos de tiempo de ejecución. Esto se observa en los comentarios que se plasmaron en la tabla 4.