**OBSERVACIONES – LABORATORIO 4**

Ana Sofía Castellanos 202114167

Martín Santiago Galván Castro 201911013

1. Máquinas de cómputo de los estudiantes:

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 (Ana Sofía) | Máquina 2 (Martín) |
| Procesadores | Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz 2.59 GHz | AMD Ryzen 3 3200G with Radeon Vega Graphics 3.60 GHz |
| Memoria RAM (GB) | 16.0 GB | 16.0 GB |
| Sistema Operativo | Windows 10 Home Single Language | 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) |
| 1000 | 462,49 | 403,12 | 21,67 |
| 2000 | 1806,25 | 1587,5 | 46,88 |
| 4000 | 7253,12 | 6346,87 | 112,504 |
| 8000 | 29050 | 27334,38 | 265,62 |
| 16000 | 125078,1 | 108010,4 | 671,88 |
| 32000 | 541536,5 | 456750 | 1446,87 |
| 64000 | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) | 3428,12 |
| 128000 | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) | 8244,37 |
| 256000 | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) | 19250 |

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) |
| 1000 | 29369,79 | 25651,04 | 1411,46 |
| 2000 | 233671,88 | 209942,73 | 6135,41 |
| 4000 | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) | 22093,75 |
| 8000 | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) | 139984,4 |
| 16000 | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) |
| 32000 | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) |
| 64000 | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) |
| 128000 | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) |
| 256000 | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) |
| 512000 | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) | Exceso de tiempo (>10 min) |

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 | Más eficiente | Menos eficiente |
| Selection sort | Más eficiente | Menos eficiente |
| Shell sort | Más eficiente | Menos eficiente |

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.

# **Maquina 2**

## **Resultados**

|  |  |  |  |
| --- | --- | --- | --- |
| Tamaño de la muestra (ARRAY\_LIST) | Insertion Sort (ms) | Selection Sort (ms) | Shell Sort (ms) |
| 1000 | 645.83 | 609.38 | 31.25 |
| 2000 | 2567.71 | 2375 | 78.12 |
| 4000 | 10427.08 | 9562.5 | 171.88 |
| 8000 | 42666.67 | 40890.62 | 406.25 |
| 16000 | 184203.12 | 159109.38 | 968.75 |
| 32000 | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) | 2156.25 |
| 64000 | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) | 5031.25 |
| 128000 | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) | 12171.88 |
| 256000 | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) | 28687.5 |
| 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) |
| 1000 | 30046.88 | 43984.38 | 1734.38 |
| 2000 | 241250 | 344046.88 | 7406.25 |
| 4000 | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) | 36218.75 |
| 8000 | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) | 165984.38 |
| 16000 | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) |
| 32000 | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) |
| 64000 | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) |
| 128000 | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) |
| 256000 | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) | Exceso de tiempo (> 10 min) |
| 512000 | - | - | - |

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 | Mas eficiente | Menos eficiente |
| Selection sort | Mas eficiente | Menos eficiente |
| Shell sort | Mas eficiente | Menos eficiente |

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?

Para la máquina 1 al analizar la comparación entre tipos de algoritmos de ordenamiento es posible evidenciar que el algoritmo Shell se comporta de la mejor manera, seguido de Insertion lo cual va de acuerdo a lo enunciado teóricamente en tanto que Shell en el peor de los casos se comporta como O(N3/2) a comparación de los otros algoritmos que se comportan en el peor de los casos como O(N2).

De igual manera al analizar la comparación de los tipos de lista y su comportamiento en el tiempo de los algoritmos es posible evidenciar que la estructura de datos ARRAY\_LIST es más veloz que la LINKED\_LIST lo cual va de acuerdo con lo encunciado teóricamente, ya que todos los algortimos de ordenamiento usados implementan getElement(…), función que tiene distintos ordenes de crecimiento temporal: O(1) para el arreglo y O(N) para la lista encadenada.

Anudado a lo anterior, cabe resaltar que las lineas de tendencia usadas permiten hallar funciones aproximadas a funciones polinómicas con grados enteros las cuales no permiten encontrar funciones demasiado precisas para los datos registrados, asi mismo para algunos datos se hallaron funciones en línea de tendencia que no corresponden a lo enunciado teóricamente, sin embargo, esto se debe a que se contaban con pocos datos en tanto que su tiempo de ejecución era muy largo.

Por parte de la maquina 2. El análisis de los datos para los algoritmos de insertion sort y selection sort se ven en parte limitados por la disponibilidad de datos. Adicionalmente, se ve limitado el análisis por el catálogo de líneas de tendencia de Excel. Sin embargo, se pueden establecer similitudes y diferencias con los datos teóricos.

Por parte de los algoritmos de selección selection sort e incertion sort, se observa que sus crecimientos en tiempos de ejecución se ven limitados por funciones cuadráticas. Por lo que se establece que, para este caso, se comportan acorde al peor caso de cada función. Por el lado del algoritmo de Shell sort, se evidencia que es el que mejor rendimiento tiene para ambos tipos de estructuras de datos. Adicionalmente, el algoritmo Shell sort con la estructura de datos array list, es el único que logra organizar todos los elementos en tiempos aceptables.

Por parte en cómo se comportan, se estima que se comportan como O(nlog(n)). Ya que, al compararlas con funciones lineales, son parecidas, y para datos mayores, el comportamiento del tiempo de ejecución tiende a ser mayor a una función lineal.

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

Si. Se puede estimar que la maquina uno fue capaz de organizar los datos en mas o menos la mitad o en tiempos menores a comparación de la maquina 2. Adicionalmente, para el caso de array\_list. La maquina 1 fue capaz de procesar mas datos que la maquina 2. En todo lo demás, no se encuentran diferencias significativas en el rendimiento de las dos máquinas.

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

Las diferencias entre los rendimientos de las maquinas se pueden deber a varios factores. Por un lado, se observa que entre las dos maquinas de cómputo, difiere el procesador de las maquinas. Una posible razón puede ser la diferencia de procesadores. Otra posible fuente de diferencia se puede deber a programas o servicios que corran en el fondo que no se hallan cerrado o tenido en cuenta. Por último, otra posible fuente de diferencia se puede deber a suspender el pc antes de las pruebas de cómputo. Si se pone en modo de suspensión durante mucho tiempo, la memoria podría no restablecerse correctamente. Como resultado, esto puede afectar el rendimiento del codigo

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

Considerando los tiempos de ejecución de los algoritmos y su comparación para las listas encadenadas y los arreglos es posible concluir que es mejor hacer uso del arreglo para implementar los algoritmos de ordenamiento, ya sea un ordenamiento del tipo selection sort, insertion sort o shell sort. Esto es debido a que estos algoritmos implementan en su código la función getElement(…) del TAD lista y el orden de crecimiento temporal de esta función difiere en estas estructuras de datos siendo O(1) para el arreglo y O(N) para la lista encadanada.