OBSERVACIONES DE LA PRACTICA

Gregorio Salazar 202022085

Valentina Uribe 201817485

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz | Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz |
| Memoria RAM (GB) | 8 | 8 |
| Sistema Operativo | Windows 10 | Windows 10 |

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

# **Graficas están en archivos de excel**

# **Maquina 1**

## **Resultados**

|  |  |  |  |
| --- | --- | --- | --- |
| Tamaño de la muestra (ARRAY\_LIST) | Insertion Sort (ms) | Selection Sort (ms) | Shell Sort (ms) |
| 1000 | 812.51 | 968.75 | 46.88 |
| 2000 | 3567.72 | 4218.51 | 109.38 |
| 4000 | 13484.38 | 15375.01 | 218.75 |
| 8000 | 56890.63 | 65062.51 | 578.13 |
| 16000 | 212187.51 | 262515.63 | 1312.51 |
| 32000 | 895343.75 | 1076437.51 | 2828.13 |
| 64000 | TE | TE | 7296.88 |
| 128000 | TE | TE | 16640.63 |
| 256000 | TE | TE | 42234.38 |
| 375942 | TE | TE | 69359.38 |

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 | 50781.31 | 37531.25 | 2421.88 |
| 2000 | 356781.21 | 318828.25 | 9421.88 |
| 4000 | TE | TE | 46906.23 |
| 8000 | TE | TE | 233859.38 |
| 16000 | TE | TE | 1085000.21 |
| 32000 | TE | TE | TE |
| 64000 | TE | TE | TE |
| 128000 | TE | TE | TE |
| 256000 | TE | TE | TE |
| 375942 | TE | TE | TE |

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 | Segundo más eficiente | Menos eficiente |
| Selection sort | Segundo más eficiente | Menos eficiente |
| Shell sort | Más eficiente por mucho | Segundo 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 | 718.75 | 843.75 | 46.875 |
| 2000 | 2906.25 | 3656.25 | 93.75 |
| 4000 | 11500 | 13281.25 | 218.75 |
| 8000 | 47437.5 | 53937.5 | 515.625 |
| 16000 | 188453.125 | 226584.375 | 1140.625 |
| 32000 | 798750.0 | 94131.5 | 2593.75 |
| 64000 |  |  | 6312.5 |
| 128000 |  |  | 15375 |
| 256000 |  |  | 37843.75 |
| 375942 |  |  | 61484.375 |

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 | 42953.125 | 38375.0 | 2281.25 |
| 2000 | 352656.25 | 312531.25 | 9296.875 |
| 4000 |  |  | 45781.875 |
| 8000 |  |  | 232546.875 |
| 16000 |  |  | 1079078.125 |
| 32000 |  |  |  |
| 64000 |  |  |  |
| 128000 |  |  |  |
| 256000 |  |  |  |
| 375942 |  |  |  |

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 | Segundo más eficiente | Menos eficiente |
| Selection sort | Segundo más eficiente | Menos eficiente |
| Shell sort | Más eficiente por mucho | Segundo 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?

Si, se comporta como se esperaba teóricamente con los algoritmos selection e insertion teniendo una complejidad de O(n^2) en teoría y al hacer un ajuste de potencia en la gráfica se ve como ambos se ajustan a una gráfica del estilo x^2 (se ve 2.1 o similares pero es aproximadamente cercano a lo que se predice) y el Shell teniendo un mucho mejor rendimiento, que se espera que esté entre O(n^3/2) y O(nlogn) que es justo lo que se observa en el análisis de las gráficas, se ve que el mejor ajuste es o un ln(x) por una constante, o una potencia con exponente 1.1 o 1.2. Los resultados son lo esperado.

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

La máquina 2 registró consistentemente tiempos ligeramente menores a los de la maquina 1, sin embargo, el comportamiento de los tiempos (la forma en la que creció) y en general los órdenes de magnitud son los mismos, mostrando que la diferencia es mínima.

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

No hay ninguna diferencia significativa. La pequeña diferencia que vemos (máquina 2 es ligeramente más rápida) se puede deber a detalles minúsculos que no afectan realmente las conclusiones generales

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

La estructura de datos más útil es sin duda la lista de arreglo. Los resultados son contundentes, se cargan 375942 datos en un minuto con el shell, y con el mismo algoritmo shellsort tarda cerca de 20 minutos en cargar tan solo 16000. Esto se puede explicar ya que las operaciones con ARRAY son mejores en tiempo de ejecución que las de SINGLE\_LINKED, ya que muchas operaciones con complejidad O(n) en la lista encadenada tienen complejidad O(1) en el arreglo.