OBSERVACIONES DE LA PRACTICA

Estudiante 1 David A. Fuquen Flórez Cod 202021113

Estudiante 2 Juan Andrés Eslava Tovar Cod 202012035

# **Ambientes de pruebas**

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz | AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz |
| Memoria RAM (GB) | 8.00 GB | 8,00 GB |
| Sistema Operativo | Windows 10 Pro 64-bits | Windows 10 Home Single Language 64-bits |

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

# **Maquina 1**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Tamaño ArrayList | Insertion sort (ms) | Shell sort (ms) | Quick sort (ms) | Merge sort (ms) |
| Small | 768 | 46,88 | 35,25 | 46,88 | 31,25 |
|  | 768 | 46,88 | 46,88 | 46,88 | 31,25 |
|  | 768 | 46,88 | 34,25 | 31,25 | 62,5 |
|  | 768 | 31,25 | 46,88 | 78,16 | 31,25 |
|  | 768 | 31,25 | 15,63 | 31,25 | 16,63 |
| Promedio |  | 40,63 | 35,78 | 46,88 | 34,58 |
|  |  |  |  |  |  |
|  | Tamaño Arraylist | Insertion sort (ms) | Shell sort (ms) | Quick sort (ms) | Merge sort (ms) |
| 10 pct | 15008 | 578,13 | 1578,13 | 5015,63 | 609,38 |
|  | 15008 | 609,38 | 1578,13 | 5015,63 | 640,63 |
|  | 15008 | 562,5 | 1578,13 | 4937,5 | 656,25 |
|  | 15008 | 562,5 | 1578,13 | 4937,5 | 671,88 |
|  | 15008 | 562,5 | 1609,38 | 5093,78 | 671,88 |
| Promedio |  | 575,00 | 1584,38 | 5000,01 | 650,00 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Small | Tamaño Single-Linked | Insertion sort (ms) | Shell sort (ms) | Quick sort (ms) | Merge sort (ms) |
|  | 768 | 1234,38 | 1046,88 | 937,5 | 125 |
|  |  | 1234,38 | 1062,5 | 984,38 | 140,63 |
|  |  | 1234,38 | 1062,5 | 953,13 | 125 |
|  |  | 1265,63 | 1078,13 | 937,5 | 125 |
|  |  | 1218,75 | 1078,13 | 937,5 | 140,63 |
| Promedio |  | 1237,50 | 1065,63 | 950,00 | 131,25 |
|  |  |  |  |  |  |
| 10 pct | Tamaño Single-Linked | Insertion sort (ms) | Shell sort (ms) | Quick sort (ms) | Merge sort (ms) |
|  | 15008 | 358458,8 | 950640,6 | \* | 39500 |
|  |  | 358468,8 | 950500,7 |  | 39750 |
|  |  | 353562,5 | 950505,3 |  | 39550 |
| Promedio |  | 356830 | 950548,90 |  | 39600 |

\*Después de más de media hora de espera, se decidió no seguir con la prueba.

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| *Insertion Sort* | 575.00 | 356830.00 |
| *Shell Sort* | 1584.38 | 950548.90 |
| *Merge Sort* | 5000.00 | \* |
| *Quick Sort* | 650.004 | 39600 |

Tabla 4. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas para 10% de los datos

# **Maquina 2**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Tamaño ArrayList | Insertion sort (ms) | Shell sort (ms) | Quick sort (ms) | Merge sort (ms) |
| Small | 768 | 62,5 | 62,5 | 78,125 | 46,875 |
|  | 768 | 46,875 | 46,875 | 62,5 | 31,25 |
|  | 768 | 62,5 | 46,875 | 62,5 | 31,25 |
|  | 768 | 46,875 | 46,875 | 62,5 | 46,875 |
|  | 768 | 62,5 | 46,875 | 62,5 | 31.25 |
| Promedio |  | 56,25 | 50 | 65,625 | 39,0625 |
|  |  |  |  |  |  |
|  | Tamaño Arraylist | Insertion sort (ms) | Shell sort (ms) | Quick sort (ms) | Merge sort (ms) |
| 10 pct | 15008 | 843,75 | 2250 | 7046,875 | 968,75 |
|  | 15008 | 828,125 | 2187,5 | 6906,25 | 937,5 |
|  | 15008 | 781,25 | 2234,375 | 7140,625 | 953,125 |
|  | 15008 | 796,875 | 2250 | 7093,75 | 937,5 |
|  | 15008 | 765,625 | 2265,625 | 6921,875 | 968,75 |
| Promedio |  | 803,125 | 2237,5 | 7021,875 | 953,125 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Small | Tamaño Single-Linked | Insertion sort (ms) | Shell sort (ms) | Quick sort (ms) | Merge sort (ms) |
|  | 768 | 1.578,125 | 1406,25 | 1296,875 | 187,5 |
|  | 768 | 1656,25 | 1421,875 | 1281,25 | 187,5 |
|  | 768 | 1671,875 | 1390,625 | 1312,5 | 203,125 |
|  | 768 | 1625 | 1406,25 | 1343,75 | 171,875 |
|  | 768 | 1640,625 | 1375 | 1250 | 156,25 |
| Promedio |  | 1.634,375 | 1.400 | 1.296,875 | 181,250 |
|  |  |  |  |  |  |
| 10 pct | Tamaño Single-Linked | Insertion sort (ms) | Shell sort (ms) | Quick sort (ms) | Merge sort (ms) |
|  | 15008 | 460890,625 | 1226171,875 | \*\*\* | 58078,125 |
|  | 15008 | \*\*\* | \*\*\* |  | 53843,75 |
|  | 15008 | \*\*\* | \*\*\* |  | 54453,125 |
| Promedio |  | 460890,625 | 1226171,875 |  | 55458,33333 |

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

|  |  |  |
| --- | --- | --- |
| **Algoritmo** | **Arreglo (ARRAYLIST)** | **Lista enlazada (LINKED\_LIST)** |
| *Insertion Sort* | 803,125 | 460890,625 |
| *Shell Sort* | 2237,5 | 1226171,875 |
| *Merge Sort* | 953,125 | 55458,33333 |
| *Quick Sort* | 7021,875 | \*\*\* |

Tabla 4. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.

# **Preguntas de análisis**

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

El comportamiento de los algoritmos no es acorde a lo enunciado teóricamente, excepto por el alogritmo Merge Sort. Vale recordar que, como fue explicado en clase que las complejidades temporales (para el peor caso) de los algoritmos son las siguiente:

Insertion sort: O(N^2)

Shell sort: O(N^3/2)

Quicksort: O(N^2)

Mergesort: O(NlogN)

Tal que, el orden teórico, de mayor eficiencia a menor eficiencia sería MergeSort, seguido de Shell Sort, y luego Insertion Sort y QuickSort con la misma eficiencia.

Evaluando nuestros resultados, este patrón no se mantiene, excepto por el MergeSort, cuya eficiencia es mejor en prácticamente todos los casos.

Para entender por qué el patrón no se mantuvo en nuestros resultados, hay que recordar que la complejidad teórica de los algoritmos se basa en el peor caso. En ese orden de ideas, es muy probable que los datos estuvieran ordenados de tal manera que no fuera el peor caso para algunos o todos de los algoritmos. Más aún, como se evaluaron diferentes bases de datos, el orden de los mismos varía así como que tan cerca están del peor caso posible para cada algoritmo.

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

Si existen diferencias, se vio que todos los resultados en la máquina 2 tomaron más tiempo que en la máquina 1. Sin embargo, se evidencia que la proporcionalidad entre los distintos tipos de ordenamiento se mantuvo, teniendo el merge sort como el método más rápido, y el quick sort como el más lento.

1. De existir diferencias, ¿a qué creen que se deben?

Estas diferencias pueden deberse a que en la máquina 2, por ejemplo, varias aplicaciones funcionan en segundo plano, lo que hacen más lento el proceso de ordenar los elementos de las listas.

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

Indudablemente, las listas de tipo ArrayList funcionan mejor que las listas enlazadas. Pues, tal como se vieron en los resultados, son indudablemente más eficientes (se ejecutan mucho más rápido).

1. Teniendo en cuenta las pruebas de tiempo de ejecución por todos los algoritmos de ordenamiento estudiados (iterativos y recursivos), proponga un ranking de los mismo de mayor eficiencia a menor eficiencia en tiempo para ordenar la mayor cantidad de obras de arte.

Aunque los resultados variaron mucho más de lo esperado en las diferentes pruebas, sin duda alguna le podemos dar el primer lugar al Merge Sort, pues fue el mejor en todos los casos posibles. Ahora, en cuanto al segundo, tercer y cuarto puesto, la certeza se reduce bastante. Dependiendo del caso y del archivo que se uso, variaron mucho. Por ende, la lista a continuación es tan sólo la mejor aproximación que podemos dar según los resultados obtenidos.

1(más eficiente)- Merge Sort

2- insertion sort

3- Shell sort

4(menos eficiente)- Quick sort