OBSERVACIONES DE LA PRACTICA

Daniel Arango Cruz Cod 202110646

Oscar Iván García Cod 201630048

# **Ambientes de pruebas**

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | AMD Ryzen 7 3700U with Radeon Vega Mobile Gfx 2.30 GHz | 2,5 GHz Dual-Core Intel Core i5 |
| Memoria RAM (GB) | 12,0 GB (9,95 GB usable) | 16 GB 1600 MHz DDR3 |
| Sistema Operativo | Windows 10 Home Single 64 bits | macOS Catalina |

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

(Debido a que presentamos varias confuciones respecto a que porcentaje de muestra debemos comparar ( Puesto que en el la guia se compara la muestra small con otro 10 % que no sabemos si se refiere al 10% de la muestra large o a la totalidad de la muestra large). Es así que los datos que compararemos en las tablas serán: La totalidad de la carga – Small y un porcentaje de la carga -Large que determinamos luego de hacer multiples intentos con cargas grandes de 120.000, 70.000, entre otras llegando a tiempos incluso de 10 minutos sin respuesta por lo que decidimos reducir la carga a 3.000 para poder hacer la comparación.

# **Maquina 1**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| Small | 294 | 1187.5 | 1171.875 | 1203.125 | 1203.125 |
| 100.00% | 3000 | 90578.125 | 89875.0 | 95437.5 | 93812.5 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (LINKED\_LIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| Small | 294 | 1187.5 | 1234.375 | 1203.125 | 1218.75 |
| 100.00% | 3000 | 93718.75 | 90625.0 | 92343.75 | 93953.125 |

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

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| *Insertion Sort* |  | 294 |
| *Shell Sort* | 3000 y 294 | 3000 |
| *Merge Sort* |  |  |
| *Quick Sort* |  |  |

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

# **Maquina 2**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| small | 294 | 1728.072 | 1692.106 | 1691.594 | 1746.587 |
| 100.00% | 3000 | 119447.384 | 120945.249 | 122933.021 | 120161.57 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (LINKED\_LIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| Small | 294 | 1616.701 | 1722.091 | 1777.539 | 1668.427 |
| 100.00% | 3000 | 120055.07 | 123072.48 | 118156.39 | 126075.53 |

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

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAYLIST) | Lista enlazada (LINKED\_LIST) |
| *Insertion Sort* | 294 y 3000 | 294 |
| *Shell Sort* |  |  |
| *Merge Sort* |  |  |
| *Quick Sort* |  | 3000 |

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?

Cooperativa: De acorde a lo enunciado teóricamente el algoritmo con mayor eficiencia aún en su peor caso debería ser el de Shell sort pues es y teniendo en cuenta los promedios obtenidos atiende a lo enunciado teóricamente, sin embargo, siento que pueden influir muchos factores como el que tan organizados estén los datos puede que la diferencia entre un algoritmo y otro que quizás en la teoría debería ser significativo en la prueba realmente no lo fue.

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

Daniel Arango: Existe una notable diferencia entre los resultados obtenidos al ejecutar las pruebas en diferentes maquinas de ellos, diferencias de casi 40 o inclusive más mseg.

Oscar Iván García: Claro que si existieron diferencias muy marcadas entre la máquina de mi compañero y la mía. En términos de tiempo fue evidente un aumento de entre 30% y 40% en términos de eficiencia temporal. Lo anterior claramente se indica en los registros de las tablas anteriores.

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

Daniel Arango: Teniendo en cuenta los datos suministrados al inicio del documento es posible que el agente diferenciador entre ambos testeos sea el procesador de cada pc, pues, aunque el Mac tuviera casi 4 GB de RAM adicionales esto no vimos que fuese influyente en la medida en la que a pesar de que se intentó cargar grandes cantidades de datos, en el administrador de tareas nunca llegó si quiera a ocupar el GB de memoria. Es por ello que tenemos el presentimiento que influencia el mucho el procesador.

Oscar Iván García: Es evidente que la memoria RAM no jugó un papel fundamental en esta diferencia de desempeños entre mi ordenador y el ordenador de mi compañero. En efecto, considero que el punto de inflexión radicó básicamente en el procesador, el cuál claramente es mucho más eficiente que el mío.

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

Daniel Arango: Teniendo en cuenta mis resultados la mejor estructura de datos teniendo en cuenta exclusivamente los tiempos de ejecución seria el tipo de estructura ArrayList pues ante el mismo volumen de datos logro obtener en varios algoritmos un mejor tiempo de ejecución en comparación al SingleLinked.

Oscar Iván García: Es evidente que, si nos remitimos exclusivamente a los tiempos de desempeño en los algoritmos de ordenamiento, es posible concluir que para una cantidad pequeña de datos utilizando la estructura de datos ArrayList, los mejores tiempos de desempeño fueron los de ShellSort y QuickSort (Además teniendo en cuenta que tuvieron diferencias muy cercanas). Por otro lado, para una carga de 3000 datos (Tuvimos que elegir ese número teniendo en cuenta los recursos computacionales limitados en la carga

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.

Daniel Arango: 1) ShellSort, 2) MergeSort, 3) InsertionSort, 4) QuickSort.

Oscar Iván García : Es evidente que es imprescindible clasificar los algoritmos en función de la cantidad de datos ( Resulta importante señalar que esto también depende de la manera en que los datos en el archivo se encuentran organizados). En consecuencia:

Ranking de algoritmos de ordenamiento para un número reducido de datos en ArrayList (Del más eficiente al menos eficiente)

1. Quick Sort

2. Shell Sort

3. Insertion Sort

4. Merge Sort

Ranking de algoritmos de ordenamiento para un gran número de datos en ArrayList (Del más eficiente al menos eficiente)

1. Insertion Sort

2. Merge Sort

3. Shell Sort

4. Quick Sort

Ranking de algoritmos de ordenamiento para un número reducido de datos en LinkedList(Del más eficiente al menos eficiente)

1. Insertion Sort

2. Merge Sort

3. Shell Sort

4. Quick Sort

Ranking de algoritmos de ordenamiento para un gran número de datos en LinkedList(Del más eficiente al menos eficiente)

1. Quick Sort

2. Insertion Sort

3. Shell Sort

4. Merge Sort