OBSERVACIONES DE LA PRACTICA

Wilder Jimenez Cod 201821314

Pablo Castrillon Cod 202122150

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | AMD Ryzen 5 4500U with Radeon Graphics 2.38 GHz |  |
| Memoria RAM (GB) | 8 GB |  |
| Sistema Operativo | Windows 10 |  |

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

# **Maquina 1**

## **Resultados**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (ARRAY\_LIST)** | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 5.00% | 245 | 23 | 18.2 | 25.4 | 16.5 | 11.3 |
| 20.00% | 490 | 14.9 | 14.9 | 11.5 | 13.5 | 30.7 |
| 30.00% | 1470 | 16.9 | 14.1 | 18.4 | 12.6 | 18.1 |
| 50.00% | 2451 | 11.1 | 16.1 | 28.6 | 16.4 | 12.4 |
| 100.00% | 4903 | 15.5 | 14.7 | 23.4 | 11.0 | 14.8 |

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (LINKED\_LIST)** | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 5.00% |  |  |  |  |  |  |
| 20.00% |  |  |  |  |  |  |
| 30.00% |  |  |  |  |  |  |
| 50.00% |  |  |  |  |  |  |
| 100.00% |  |  |  |  |  |  |

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

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAY\_LIST) | Lista enlazada (LINKED\_LIST) |
| Merge sort |  |  |
| Quick sort |  |  |

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 ARRAY\_LIST.
  + Comparación de rendimiento LINKED\_LIST.
  + Comparación de rendimiento para Insertion Sort.
  + Comparación de rendimiento para Selection Sort.
  + Comparación de rendimiento para Shell Sort.
  + Comparación de rendimiento para MergeSort.
  + Comparación de rendimiento para QuickSort.

# **Maquina 2**

## **Resultados**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (ARRAY\_LIST)** | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 5.00% |  |  |  |  |  |  |
| 20.00% |  |  |  |  |  |  |
| 30.00% |  |  |  |  |  |  |
| 50.00% |  |  |  |  |  |  |
| 100.00% |  |  |  |  |  |  |

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (LINKED\_LIST)** | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 5.00% |  |  |  |  |  |  |
| 20.00% |  |  |  |  |  |  |
| 30.00% |  |  |  |  |  |  |
| 50.00% |  |  |  |  |  |  |
| 100.00% |  |  |  |  |  |  |

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

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arreglo (ARRAY\_LIST) | Lista enlazada (LINKED\_LIST) |
| Merge sort |  |  |
| Quick sort |  |  |

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 ARRAY\_LIST.
  + Comparación de rendimiento LINKED\_LIST.
  + Comparación de rendimiento para Insertion Sort.
  + Comparación de rendimiento para Selection Sort.
  + Comparación de rendimiento para Shell Sort.
  + Comparación de rendimiento para MergeSort.
  + Comparación de rendimiento para QuickSort.

# **Preguntas de análisis**

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

Sí, ya que el aumento de tiempo de los ordenamientos va acorde a sus complejidades. Por ejemplo, el crecimiento del algoritmo merge tiene una aumento de tiempo muy bajo, siguiento una función logarítmica.

Por otro lado, los algoritmos tardaron más en listas de tipo Linked list que en Array list, esto debido a que es más difícil buscar espaciios vacíos de memoria. Sin embargo, el algoritmo merge, el cual es el que más memoria requiere, tuvo la menor diferencia entre su tiempo en Linked list y su tiempo en Array list

1. ¿Existe alguna diferencia entre los resultados obtenidos al ejecutar las pruebas en diferentes máquinas?
2. De existir diferencias, ¿A qué creen ustedes que se deben dichas diferencias?

La memoria es un factor, ya en una lista de tipo Linked List, el computador debe buscar espacios en la memoria donde quepan los datos. Si la memoria está llena, será más difícil buscar estos espacios. Por otro lado, el método de ordenamiento de cada algoritmo hace que la complejidad suba, y hay computadores que pueden realizar operaciones en menor tiempo que otros, lo que hace que en grandes conjuntos de datos hayan grandes diferencias.

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

Array List, ya que en todos los algoritmos tuvo un menor tiempo. En todos los algoritmos menos merge, el tiempo de ejecución en una lista encadenada fue casi el doble que el tiempo de la lista arreglada.

1. Para el caso analizado de ordenamiento de las películas, teniendo en cuenta los resultados de tiempo reportados por todos los algoritmos de ordenamiento (iterativos y recursivos), proponga un ranking de los algoritmos de ordenamiento (de mayor eficiencia a menor - con relación a los tiempos de ejecución) para ordenar la mayor cantidad de películas.

El algoritmo más eficiente es selection para un menor número de datos. Sin embargo, merge es el más eficiente en los grandes grupos de datos.

Después de selection y merge, el algoritmo más eficiente es quick sort, seguido de Shell, y por último, insertion es el algoritmo más ineficiente en función del tiempo de ejecución. Excepto en grandes grupos de datos, ya que en la lista más grande, insertion fue el segundo algoritmo más rápido después de merge.