OBSERVACIONES DE LA PRACTICA

Juan José Osorio (202017020)

Thais Tamaio Ramírez (202022213)

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | 3,1 GHz Intel Core i5 de dos núcleos | Intel(R) Core(TM) i7-3667U CPU @ 2.00GHz 2.50 GHz |
| Memoria RAM (GB) | 8 GB 2133 MHz LPDDR3 | 4.00 GB (3.90 GB usable) |
| Sistema Operativo | macOS Big Sur versión: 11.2.1 | Windows 10 Home 64-bits |

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

# **Maquina 1**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 665,03 | 756,61 | 44,53 | 30,73 | 31,15 |
| 2000 | 2626,41 | 2985,29 | 91,4 | 62,18 | 61,06 |
| 4000 | 10759,13 | 12601,12 | 195,3 | 115,07 | 120,56 |
| 8000 | 44174,44 | 53291,52 | 482,21 | 243,85 | 250,46 |
| 16000 | 184837,87 | 213631,37 | 1057,72 | 515,47 | 535,67 |
| 32000 |  |  | 2460,17 | 1097,2 | 1140,41 |
| 64000 |  |  | 5808,17 | 2425,52 | 2445,72 |
| 128000 |  |  | 14018,36 | 5346,13 | 5232,64 |
| 256000 |  |  | 35920,55 | 11397,8 | 11099,27 |
| 375942 |  |  |  |  |  |

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]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 46673 | 41332 | 2314 | 1940,88 | 259,64 |
| 2000 | 379918 | 341968 | 10910 | 8848,88 | 923,17 |
| 4000 | 3113859 | 2726708 | 50537 | 35066,52 | 3626,66 |
| 8000 |  |  | 252813,92 | 156185,83 | 14505,27 |
| 16000 |  |  |  | 669874,13 | 59039,22 |
| 32000 |  |  |  |  | 238846,93 |
| 64000 |  |  |  |  | 963661,04 |
| 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) |
| Merge sort | x |  |
| Quick sort | x |  |

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.
  + Comparación de rendimiento para MergeSort.
  + Comparación de rendimiento para QuickSort.

# **Maquina 2**

## **Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Selection Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 953,125 | 1125 | 62,5 | 31,25 | 62,5 |
| 2000 | 3859,375 | 4375 | 140,625 | 78,13 | 93,75 |
| 4000 | 15750,01 | 17625 | 312,5 | 187,5 | 171,88 |
| 8000 | 62859,375 | 73734,375 | 718,75 | 328,13 | 375 |
| 16000 | 272937,5 | 311406,25 | 1578,125 | 718,75 | 781,25 |
| 32000 |  |  | 3484,38 | 1562,5 | 1625 |
| 64000 |  |  | 8593,75 | 3359,38 | 3421,88 |
| 128000 |  |  | 20281,25 | 7843,75 | 7328,13 |
| 256000 |  |  | 53000 | 15015,63 | 15875 |
| 375942 |  |  | 84328,125 | 23687,5 | 16921,798 |

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]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| 1000 | 62031,25 | 57390,625 | 3359,375 | 2453,13 | 343,75 |
| 2000 | 542203,125 | 468906,25 | 14875 | 10218,75 | 1156,25 |
| 4000 | 4245234,38 | 3782546,875 | 67484,375 | 50484,38 | 4781,25 |
| 8000 |  |  | 364671,88 | 198640,63 | 19015,63 |
| 16000 |  |  |  | 946359,38 | 77984,38 |
| 32000 |  |  |  |  | 323765,63 |
| 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) |
| Merge sort | x |  |
| Quick sort | x |  |

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.
  + Comparación de rendimiento para MergeSort.
  + Comparación de rendimiento para QuickSort.

# **Preguntas de análisis**

**• ¿El comportamiento de los algoritmos es acorde a lo enunciado teóricamente?**

Con base en lo enunciado teóricamente con respecto al tiempo de ejecución y a los órdenes de crecimiento de los diferentes ordenamientos que se emplearon, se conoce que en el mejor de los casos los ordenamientos más rápidos deberían ser Merge Sort y Quick Sort, y que en el peor de los casos, el ordenamiento más rápido debería ser el de tipo Merge Sort. Por otro lado, con base en la teoría de las dos estructuras de datos que se utilizaron, se conoce que recorrer una lista para organizar los elementos de esta y cargar los datos con una ARRAY LIST es más rápido que con una SINGLE LINKED LIST. Esto se debe a que los elementos en una ARRAYLIST se encuentran en posiciones contiguas en memoria, por lo que acceder a cada uno de estos se puede conseguir con un tiempo constante. Por el contrario, al usar una SINGLE LINKED LIST los elementos no están en posiciones adyacentes, por lo que cada elemento debe conservar una referencia a la posición del elemento siguiente. Esto evita que recorrer la lista se pueda realizar en un tiempo constante y, al mismo tiempo, ocupa más memoria.

Tomando como referencia los diferentes tiempos de ejecución obtenidos al realizar las pruebas con diferentes ordenamientos, estructuras de datos y cantidades de datos, se puede determinar que el comportamiento de los algoritmos es acorde a lo enunciado teóricamente. Esto se debe a que los ordenamientos nuevos que se implementaron en este laboratorio (Merge Sort y Quick Sort) utilizan un método de recursión para ordenar los datos, lo cual conlleva a menores tiempos de ejecución en comparación con los ordenamientos iterativos utilizados en el laboratorio pasado (Insertion Sort, Selection Sort y Shell Sort). Esto se debe, a que el ordenamiento Insertion Sort, Selection Sort y Shell Sort tienen en el mejor caso un orden de crecimiento de O(n), O(n^2) y O(n^1.25) y en el peor caso, un orden de crecimiento de O(n^2), O(n^2) y O(n^3/2) respectivamente.

Por el contrario, se conoce que en promedio y en el mejor caso, Merge Sort y Quick Sort tienen un orden de crecimiento de O(n log (n)), lo cual explica porqué la diferencia de los tiempos de ejecución con un mismo número de datos y estructura de datos entre estos dos ordenamientos suele ser bastante pequeña (sobre todo al utilizar la estructura de datos ARRAY LIST). Asimismo, al contar con un orden de crecimiento que es significativamente menor al de Shell Sort, Insertion Sort y Selection Sort, se podría concluir que los ordenamientos Merge Sort y Quick Sort serán más eficientes (en términos del tiempo de ejecución). No obstante, la teoría indica que en el peor de los casos el ordenamiento Quick Sort puede llegar a tener un orden de crecimiento de O(n^2), lo que podría explicar porqué existe una diferencia significativa en los tiempos de ejecución entre este ordenamiento y Merge Sort al usar SINGLE LINKED LIST.

**• ¿Existe alguna diferencia entre los resultados obtenidos al ejecutar las pruebas en**

**diferentes máquinas?**

Teniendo en cuenta los resultados al ejecutar el algoritmo con dos estructuras de datos diferentes (ARRAY LIST y SINGLE LINKED LIST), al utilizar cinco tipos de ordenamientos (Insertion Sort, Selection Sort, Shell Sort, Merge Sort y Quick Sort) y al utilizar cantidades diferentes de datos, se pudo apreciar una diferencia significativa en el tiempo de ejecución del algoritmo para cada máquina. Esto se debe, a que todos los tiempos de ejecución fueron menores en la máquina 1, y que la diferencia de los tiempos de ejecución en las dos máquinas (a medida que la cantidad de datos aumenta) puede llegar a ser bastante significativa.

**• De existir diferencias, ¿A qué creen ustedes que se deben dichas diferencias?**

Consideramos que las diferencias que se presentaron en los tiempos de ejecución de los algoritmos, al utilizar diferentes estructuras de datos, cantidades de datos y ordenamientos, son producto de los diferentes procesadores y las respectivas características de estos en las dos máquinas utilizadas. Con base en esto, la primera máquina utilizada cuenta con un procesador que tiene 3,1 GHz y la segunda máquina cuenta con una capacidad que va desde 2,0 GHz a 2,50GHz. Esta medida es conocida como Gigahercios, la cual representa la cantidad de operaciones que puede realizar un procesador en un período de un segundo, en otras palabras, mide la velocidad de un procesador. Por consiguiente, se puede decir que la máquina 1 cuenta con un procesador más rápido que la máquina 2, lo cual explica la diferencia de los tiempos de ejecución.

**• ¿Cuál Estructura de Datos es mejor utilizar si solo se tiene en cuenta los tiempos de**

**ejecución de los algoritmos?**

Si solo se toman en cuenta los tiempos de ejecución de los algoritmos, se podría decir que el más eficiente en este aspecto, es la estructura de datos de tipo ARRAY LIST. Esto se debe, a que los tiempos de ejecución fueron menores (para este algoritmo en específico de encontrar videos con mayor número de vistas) con todos los ordenamientos utilizados y con diferentes cantidades de datos para procesar.

**• Para el caso analizado de ordenamiento de los videos, teniendo en cuenta los resultados de tiempo reportados por todos los algoritmos de ordenamiento estudiados (iterativos y recursivos), proponga un ranking de los algoritmos de ordenamiento (de mayor eficiencia a menor eficiencia en tiempo) para ordenar la mayor cantidad de videos.**

1. Merge Sort
2. Quick Sort
3. Shell Sort
4. Insertion Sort
5. Selection Sort