Isai Daniel Chacón Silva - 201912015

Nicolás Aparicio Claros - 201911357

Estructuras de datos y algoritmos

LabSorts-S05-G01

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

Características	**Máquina 1**	**Máquina 2**
Procesadores	Intel ® core TM i5-8250U CPU @1,60GHz	AMD Ryzen 7 5700U @1,80GHz
Memoria RAM /GB)	8 GB	8 GB
Sistema Operativo	Microsoft Windows 10 Pro basado en x64	Microsoft Windows 10 Home Single Language basado en x64

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

Porcentaje de la muestra	**Tamaño de la muestra (Array List)**	**Insertion[ms]**	**Shell [ms]**	**Quick [ms]**	**Merge [ms]**
small Máquina 1	768	78,12	85,94	88,54	70,32
small Máquina 2	768	36,46	62,5	74,22	46,88
10% Máquina 1	15008	718,75	2046,88	9880,21	973,96
10% Máquina 2	15008	406,25	1078,13	3750,00	500,00

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

Porcentaje de la muestra	**Tamaño de la muestra (Linked List)**	**Insertion[ms]**	**Shell [ms]**	**Quick [ms]**	**Merge [ms]**
small Máquina 1	768	2078,13	2937,50	2593,75	203,13
small Máquina 2	768	1640,63	1656,25	1390,63	203,13
10% Máquina 1	15008	659375,01	1368125,00	No convergió (recursos)	64703,13
10% Máquina 2	15008	239750,00	622890,63	1359140,63	27960,94

Compare los resultados obtenidos con la complejidad teórica de cada algoritmo y con los resultados obtenidos de las pruebas en ambas maquinas respondiendo las siguientes preguntas:

- ¿El comportamiento con relación al orden de crecimiento temporal de los algoritmos es acorde a lo enunciado teóricamente?

En primer lugar, la teoría indica que los órdenes de complejidad temporal para los algoritmos de ordenamiento para los peores casos vienen dados por las siguientes ecuaciones [1]:

Insertion: $\frac{N^2}{2}$

Shellsort: $N^{\frac{3}{2}}$

Quicksort: $\frac{N^2}{2}$

Merge: $N \log(N)$

Para observar la relación del crecimiento temporal respecto a los enunciados teóricos se construyó la Tabla. 4 que muestra la relación en cuanto al incremento de datos entre la experimentación del ordenamiento para las bases de datos -small y -pct10, así como el incremento en tiempo como fracción para cada algoritmo.

Tabla 4. Relación de cambio entre el tiempo de ejecución y aumento de datos.

ARRAY LIST	**Relación máquina 1**	**Relación máquina 2**
Datos	19,54166667	19,54166667
Insertion	9,200588838	11,14234778
Shell	23,81750834	17,25002667
Quick	111,590317	50,52550975
Merge	13,85035078	10,66552901
SINGLE LINKED	**Relación máquina 1**	**Relación máquina 2**
Datos	19,54166667	19,54166667
Insertion	317,2924721	146,132888
Shell	465,7446809	376,0849087
Quick	N/A	977,3560401
Merge	318,5306454	137,6504701

De la Tabla. 5 se puede observar que la relación más baja para los diferentes casos de arreglos numéricos (Array List y Single Linked) la mantuvo el algoritmo de ordenamiento Insertion mientras que la relación más alta corresponde al ordenamiento por el método Quick. Adicionalmente, para el caso de Quick se evidenció que en la máquina 1 nunca convergió el resultado.

En comparación de nuestros resultados con la teoría, encontramos que el algoritmo más implementado corresponde a Quick Sort; no obstante, para nuestro caso, se puede determinar que este método de ordenamiento fue el que mostró un peor rendimiento a lo largo de la fase de experimentación. Esto incluso cuando son precisamente estos dos algoritmos, así como se mostró anteriormente, poseen las mismas ecuaciones de orden. Esto permite inferir que el ordenamiento previo de la base de datos es bastante sensible al algoritmo que se vaya a implementar sobre esta.

Finalmente, encontramos un comportamiento particular con el método de ordenamiento Merge, ya que al ser implementado con una estructura de tipo Array List, Merge Sort tenía el segundo

mejor desempeño mientras que con una estructura de dato Single Linked le permitió obtener un menor tiempo de ejecución en comparación a los otros algoritmos.

- ¿Existe alguna diferencia entre los resultados obtenidos al ejecutar las pruebas en diferentes máquinas? De existir diferencias, ¿a qué creen que se deben?

De acuerdo con los resultados de las Tablas. 2 y 3 es posible concluir que, en efecto, realizar la ejecución de los códigos en diferentes máquinas tiene un impacto sobre el tiempo de procesamiento de cada uno de los algoritmos. Se puede evidenciar que en la máquina 2, los tiempos de reordenamiento son menores que los de la máquina 1. Es importante mencionar que, dentro de las propiedades y/o características que difieren entre cada una de las máquinas, se encuentra el procesador. La máquina 1 cuenta con un procesador Intel i5 mientras que la máquina 2 posee un Ryzen 7 (análogo a un Intel i7). Entre un procesador i5 y un procesador i7 cambia la potencia que genera cada uno de estos procesadores en las máquinas. Un procesador i5 posee una menor potencia que un i7, así como una menor memoria en cache. Finalmente, es importante mencionar que la memoria cache se encarga de almacenar datos necesarios para que el dispositivo pueda operar de forma continua. De esta manera, es posible justificar la diferencia de comportamiento a la hora de ejecutar los algoritmos en cada una de las máquinas [2]. Tanto la memoria RAM como el sistema operativo para ambas máquinas no presentan diferencias.

Tabla 5. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.

Algoritmo	**Array List**	**Linked List**	**Diferencia [ms]**
Insertion	406,25	239750,00	239343,75
Shell	1078,13	622890,63	621812,50
Merge	500,00	27960,94	27460,94
Quick	3750,00	1359140,63	1355390,63

- ¿Cuál Estructura de Datos (ARRAY_LIST o SINGLE_LINKED) funciona generalmente mejor si solo se tiene en cuenta los tiempos de ejecución de los algoritmos?

Teniendo en cuenta los tiempos de ejecución de los ordenamientos, la estructura de dato Array List resulta mucho más eficiente para la base de datos dada, así como la tarea que se quería realizar (ordenar las obras de acuerdo a la fecha de adquisición). Esto se afirma debido a que se observó cómo los tiempos fueron órdenes de magnitud menores en milisegundos para todos los casos experimentados en este laboratorio. Vale aclarar que esta comparación no tiene en cuenta la disponibilidad de espacio en memoria, sino solamente al tiempo.

- Teniendo en cuenta las pruebas de tiempo de ejecución reportadas por los algoritmos de ordenamiento probados (iterativos y recursivos), proponga un listado de estos ordenarlos de menor a mayor teniendo en cuenta el tiempo de ejecución que toma ordenar las obras de arte.

Teniendo en cuenta las pruebas en función del tiempo reportadas en este informe para los algoritmos iterativos como recursivos, el listado propuesto para ordenar de menor a mayor para el tiempo de ejecución sería: Insertion sort, Merge sort, Shell sort, Quick sort.

Referencias

[1] R. Sedgewick and K. Wayne, *Algorithms fourth edition*, vol. 53, no. 9. 2017.

[2] AcerStore. n.d. Diferencias entre procesadores i3, i5 e i7. [online] Available at: <https://www.acerstore.cl/news/diferencias-entre-procesadores-i3-i5-e-i7/> [Accessed 15 September 2021].