OBSERVACIONES DE LA PRACTICA

Estudiante 1 -Ana Sofía Villa Benavides,201923361, as.villa@uniandes.edu.co
Estudiante 2 - Daniela Alejandra Camacho Molano,202110974, d.camachom@uniandes.edu.co

Ambientes de pruebas

	Máquina 1	**Máquina 2**
Procesadores	Intel® Core™2 Duo Processor T6670 2.20 GHz	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
Memoria RAM (GB)	4,00 GB (3,87 GB usable)	8.00 GB (7.70 GB utilizable)
Sistema Operativo	Windows 10 pro	Sistema operativo de 64 bits, procesador x64

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

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)	768	58.69	229.16	15432.16	239.58
05.00%	7572	411.33	2947.91	_	2880.21
10.00%	15008	802.08	6343.75	_	6140.625
20.00%	29489	1585.93	13405.25	_	12531.25
30.00%	43704	2296.875	29570.315	_	19210.94
50.00%	71432	3812.34	35593.75	_	32929.68
80.00%	111781	5963.54	58536.45	_	55414.10
100.00%	138150	7367.18	74625.65	_	73286.45

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)	768	125.0	630.20	68296.875	437.15
05.00%	7572	9292.87	65875.0	_	23691.56

10.00%	15008	38015.6	303578.125	_	92656.25
20.00%	29489	144312.5	_	_	_
30.00%	43704	_	_	_	
50.00%	71432			_	
80.00%	111781			_	
100.00%	138150			_	

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

Algoritmo	Arreglo (ARRAYLIST)	Lista enlazada (LINKED_LIST)
Insertion Sort	411.33	9292.87
Shell Sort	2947.91	65875.00
Merge Sort	2880.21	232691.56
Quick Sort	_	_

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

**los datos de tiempo se basan en tamaño de muestra 5% ya que es hasta domingo se tienen datos de tiempo para todas las categorías*

La combinación estructura/ algoritmo mas eficiente es insertion sort.

Maquina 2

Resultados

*dani en estas tablas toca ir avanzando de porcentaje en los archivos hasta que ya tu compu no de mas, no son solo 0,5% y 100%

Porcentaje de la muestra [pct]	Tamaño de la muestra (ARRAY_LIST)	Insertion Sort [ms]	Shell Sort [ms]	Quick Sort [ms]	Merge Sort [ms]
small	294	20.83	44.75	734.75	67.67
0.50%	691	31.25	93.75	4255.21	93.75
10.00%	13815	270.83	2135.41667		1546.88
20.00%	27630	285.42	3348.95833		3223.96
30.00%	41445	567.71	5234.375		4906.25
40.00%	55260	750.00	6864.58333		6557.29
50.00%	69075	942.708333	9083.33333		9007.81
60.00%	82890	1125.00	10578.125		10218.75
70.00%	96705	1296.88	12593.75		11843.75

80.00%	110520	1468.75	14640.625		14656.25
90.00%	124335	1671.88	18046.875		16046.88
100.00%	138150	2015.63	19109.375		18906.25

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	15.63	62.5	1750	62.50
0.50%	691	46.88	218.75	18687.50	140.63
10.00%	13815	11625.00	92265.625		30515.63
20.00%	27630	49281.25			
30.00%	41445				
40.00%	55260				
50.00%	69075				
60.00%	82890				
70.00%	96705				
80.00%	110520				
90.00%	124335				
100.00%	138150				

LOS DATOS VACIOS SE DEBEN A QUE LA MAQUINA NO RESPONDIO CON ESE TAMAÑO DE LA MUESTRA.

Tabla 5. Comparación de tiempos de ejecución para los ordenamientos en la representación lista enlazada.

Algoritmo	Arreglo (ARRAYLIST)	Lista enlazada (LINKED_LIST)
Insertion Sort	152.08	15242.19
Shell Sort	757.972222	30848.95
Merge Sort	569.43	10239.58
Quick Sort	2494.97916	10218.75

Tabla 6. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas

La combinación estructura/ algoritmo mas eficiente es insertion sort.

Preguntas de análisis

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

El comportamiento general de los algoritmos en cuanto a las dos estructuras de datos es acorde a lo que enuncia la teoría, los tiempos asociados a array list son menores lo que coincide con que la complejidad de los algoritmos es menor al usar este tipo de estructura. Por otro lado, al fijarse únicamente en los algoritmos entre ellos no coincide ya que en general se obtuvo que el más rápido fue inserción luego merge, shell y finalmente quick sort. Esto no va acorde con sus complejidades vistas en clase.que indican que al tratarse del peor caso el orden de menor a mayor complejidad es: merge O(nlogn), shell O(n^3/2), insertion O(n^2), quicksort O(n^2) [caso promedio= O(nlogn)]. Estas complejidades son con arraylist.

2) ¿Existe alguna diferencia entre los resultados obtenidos al ejecutar las pruebas en diferentes máquinas?
   Si existe una diferencia entre los resultados obtenidos, de manera global la máquina 1 presenta tiempos de ejecución mayores a la maquina 2.
3) De existir diferencias, ¿a qué creen que se deben?
   El core i5 tiene procesadores de alta gama y tienen mejor rendimiento grafico debido a los microprocesadores de 4 núcleos fabricados en 45nm. Por otro lado, el core 2 es una tecnología antigua trabaja con: trabaja a 45nm o 65nm y 65 W de consumo. La mayor diferencia es la antigüedad de cada máquina, ya que al ser una tecnología más avanzada permite que el computador procese los datos de manera más rapida.
   INFORMACIÓN SACADA DE: https://qloudea.com/blog/diferencias-intel-core-2-duo-i3-i5-i7/
4) ¿Cuál Estructura de Datos funciona mejor si solo se tiene en cuenta los tiempos de ejecución de los algoritmos?
   Si solo se tiene en cuenta los tiempos de ejecución de los algortimos, la estructura de datos que funciona mejor es ARRAY ya que tiempos promedio mucho menores. Esto coincide con lo visto en la teoría.
5) 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.

    1. Insertion
    2. Merge
    3. Shell
    4. Quick