OBSERVACIONES DE LA PRACTICA

Estudiante 1: Daniel Gomez | Cod 201728920
Estudiante 2: Jenifer Arce | Cod 202014993

Ambientes de pruebas

	Máquina 1	Máquina 2
Procesadores	Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz, 2601 Mhz, 2 procesadores principales, 4 procesadores lógicos	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Memoria RAM (GB)	16 GB	8 GB
Nombre del SO	Microsoft Windows 10 Pro. 64-bits	Windows 10 64-bits

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					
0.5%	690	248.13	265.63	421.88	218.77
10%	13815	5187.52	9687.54	52296.87	3515.63
100.00%	138150	46093.75	268171.87	+1800000[1]	71312.50

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					
0.5%	690	1898.34	2609.37	2125.34	468.45
10.00%	138150	770468.75			105765.62

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

Algoritmo[2]	Arreglo (ARRAYLIST)	Lista enlazada (LINKED_LIST)
Insertion Sort	71.3%	
Shell Sort	72.88%	
Merge Sort	100%	

[1] A pesar de cambiar el límite de recursión previo a la prueba, el programa se detuvo con error después de 30 minutos corriendo porque sobrepasó el nuevo límite de recursión.

[2] Se utilizó una formula de eficiencia relativa, dónde el mejor algorítmo tiene una porcentaje 100% y los demás están en función de ese porcentaje.

Quick Sort	20%	

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	768	421.88	453.125	**484.375**	234.38
0.5%	690	328.13	328.13	**421.88**	375.0
100.00%	Hecho con rango de fechas[3]	49921.85	234281.25	**Falló** [4]	74437.5

Tabla 5. 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	3156.25	2875	2937.5	515.63
0.50%	690	1500	2093.75	1984.375	437.5
10.00%	138150	-	-[5]	-	103906.25

Tabla 6. Comparación de tiempos de ejecución para los ordenamientos en la representación lista enlazada.

Algoritmo	Arreglo (ARRAYLIST)	Lista enlazada (LINKED_LIST)
Insertion Sort	x	
Shell Sort	x	
Merge Sort	x	
Quick Sort	x	

Tabla 7. 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?

Respuesta:

Sí, observamos el comportamiento indicado en la complejidad teorica en la mayoría de los ordenamientos. En especial, en el caso de los array_list pudimos observar que los ordenamiento iterativos tienen mejores tiempos en comparación a los ordenamientos recursivos. Adicionalmente, pudimos observar que el insertion sort tuvo el mejor tiempo de los cuatro algoritmos, de tal modo que se puede ver representado el mejor caso de complejidad que es O(n). En el caso del merge sort tuvo el segundo mejor desempeño en tiempo de los cuatro algoritmos, teniendo una complejidad temporal de su mejor caso O(nLog(n)). Por otro lado, en el caso de los ordenamientos recursivos pudimos notar el peor caso en el quick sort (O(n^2)),

[3] Fecha inicial: 1984-03-21 Fecha final: 2002-04-25

[4] El límite de recursión fue superado, además los intentos se estaban tardando más de 40 minutos.

[5] Las pruebas de insertion, shell y quick se demoraron más de 30 minutos

su prueba de rendimiento duró más de 40 minutos en cada máquina, además de causar el error de "recursion depth exceeded in comparison" en dos de los tres intentos. Dado que este algoritmo se demoró tanto tiempo decidimos parar de hacer la prueba en el tercer intento.

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

Respuesta:
Lo que pudimos observar es que la máquina #1 tiene un tiempo más rápido en la mayoría de las pruebas. Tanto en ambos tipos de lista, linked_list y array_list, además de los distintos tipos de algoritmos pudimos observar tiempos más rápidos, con diferencias de por lo menos 3000 ms en cada intento. Otro aspecto a resaltar es que como se mencionó anteriormente, hubo problemas al hacer las pruebas con el quick sort, en los intentos hechos la máquina que fallaba primero era la maquina #2 (tiene menor ram).

3) Sí, existe una diferencia entre ambas máquinas. De existir diferencias, ¿a qué creen que se deben?

Respuesta:
Creemos que esta diferencias en las pruebas de rendimiento ocurren principalmente por diferencias en los procesadores y ram de cada máquina. Como es observado en la tabla 1, la máquina 1 tiene tiene una mayor memoria ram, además de un procesador Intel-i7, comparado con 8gb de ram y procesador intel i5 de la máquina 2 . Estas diferencias hacen que tengan un mejor desempeño la máquina #1.

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

Respuesta:
Las pruebas indican que MergeSort es el mejor algorítmo. Lo anterior se puede observar en los resultados de las pruebas. Este resultado está de acuerdo con la teoría que nos dice que la complejidad de este algorítmo es linearítmica, la menor complejidad de todos los algorítmos evaluados. Ahora bien, en las pruebas que se realizaron del 100% en ArrayList salió un resultado inesperado: InsertionSort tuvo un mejor tiempo que MergeSort, creemos que el uso de espacio de memoria física de MergeSort finalmente hizo que tuviera un peor rendimiento que InsertionSort.

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.

Respuesta:

Ranking algoritmos
MergeSort
InsertionSort
ShellSort
QuickSort