OBSERVACIONES DE LA PRACTICA

Juan Camilo Neira Campos - Cod 201922746
Daniel Dorado Toro - Cod 201821010

Ambientes de pruebas

	Máquina 1	Máquina 2
Procesadores	Intel® Core™ i7-1065G7 CPU @ 1.30GHz 1.50GHz	Intel® Core™ i5 Dual-Core @ 2.5 GHz
Memoria RAM (GB)	8.0 GB	8.0 GB
Sistema Operativo	Windows 10 Pro 64-bits	macOS Catalina

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	750	18.75	21.87	15.62	25
10.00%	15000	271.88	756.25	2179.7	350

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	750	927	796.9	781.25	93.75
10.00%	15000	-	-	-	34484.38

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

Algoritmo	Arreglo (ARRAYLIST)	Lista enlazada (LINKED_LIST)
Insertion Sort	18.75	927
Shell Sort	21.87	796.9
Merge Sort	25	93.75
Quick Sort	15.62	781.25

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	750	77.88	46.53	46.75	40.00
10.00%	13418	1260.79	1434.04	4230.69	809.15

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	750	3957.33	1632.30	1866.95	221.36
100.00%	13418	-	-	-	64654.40

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

Algoritmo	Arreglo (ARRAYLIST)	Lista enlazada (LINKED_LIST)
Insertion Sort	77.88	3957.33
Shell Sort	46.53	1632.30
Merge Sort	46.75	221.36
Quick Sort	40	1866.95

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?

Los datos no permiten ver claramente la complejidad temporal, puesto que las pruebas solo contienen dos puntos. No obstante, es posible hacer una comparación de la velocidad de crecimiento. El de mayor velocidad de crecimiento es quick sort y el de menor es merge sort. Esto es consistente con la teoría, pues quick sort en el peor caso es O(n^2) (al igual que insertion y shell sort), mientras merge es O(n log n) en el peor de los casos.

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

Sí, la máquina 2 es apreciablemente más lenta que la máquina 1.

3) De existir diferencias, ¿a qué creen que se deben?

Estas diferencias entre los resultados de tiempo se pueden deber a las siguientes tres razones. Primero, el tamaño de la memoria RAM significa que un computador tiene mayor memoria principal, donde se almacena una cantidad cada vez mayor de datos. Esto, asimismo, depende del espacio ocupado de memoria RAM, el cual no depende de las características de la máquina, sino del usuario. Segundo, el número de núcleos permite llevar a cabo una mayor cantidad de tareas simultaneas y por tanto ejecutar el programa en un menor tiempo. Tercero, a mayor GHz la máquina tiene un mayor rendimiento.

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

La estructura de datos que funciona generalmente mejor y específicamente mejor en algoritmos de ordenamiento iterativos son los arreglos o arrayList. Esto se debe a que la función Exchange, que se utiliza para intercambiar dos valores entre posiciones, tiene un orden de crecimiento O(1) en los arreglos, puesto que estas estructuras de datos retornan las posiciones en O(1). Esto contrasta con las listas encadenas, ya que retornar las posiciones tiene un orden de crecimiento temporal O(n) al

igual que la función Exchange cuando el algoritmo de ordenamiento de implementa sobre esta estructura de datos.

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. Quick Sort.
    2. Shell Sort.
    3. Merge / Insertion
    4. Merge / Insertion