OBSERVACIONES DEL RETO 1

Estudiante 1 David A. Fuquen Flórez Cod 202021113. Requerimiento individual: 3

Estudiante 2 Juan Andrés Eslava Tovar Cod 202012035. Requerimiento individual: 4

Análisis de complejidad:

Requerimiento 1:

En este primer requerimiento, tenemos 2 indicios que nos indican como podría ser la complejidad. En primer lugar, tenemos un ordenamiento MergeSort para organizar los artistas por su año de nacimiento, que tiene una complejidad de NlogN en el peor caso. Como segundo indicio, tenemos un recorrido de toda la lista artistas, pata crear una nueva sublista con los artistas dentro del rango especificado, que tiene una complejidad de N (es un recorrido por toda la lista) en su peor caso. Por ende, considerando la mayor complejidad, se tiene que es NlogN.

Requerimiento 2:

Para este requerimiento, se puede partir de varias acciones que definen la complejidad de esta función. Empezando por los recorridos de las listas (obras y sorted_list), estos tienen una complejidad de O(N). Pero, al hacer un ordenamiento haciendo uso de MergeSort, para organizar la lista de obras por su fecha de adquisición, se tiene que al ser NLogN, la complejidad de esta función de ordenamiento en su peor caso, la complejidad final de la función es NLogN.

Requerimiento 3:

Para este requerimiento debemos tener en cuenta 5 funciones. Empezando por los recorridos de listas, que se utilizan en 4 de las 5 funciones, se tiene que la complejidad para cada una de esas funciones es de O(N), excepto para las funciones total_obras y lista_total_tecnicas, que realizan un recorrido dentro de un recorrido, por lo que son de complejidad $O(N^2)$. Sin embargo, hay una función, técnica_más_utilizada, que utiliza MergeSort, para el ordenamiento de una lista, teniendo la función una complejidad NlogN. Por lo que, en total, la complejidad de este requerimiento es $O(N)+O(N)+O(N^2)+ O(N^2)+O(NLogN)$.

Requerimiento 4:

Requerimiento 5:

Para este requerimiento, se hizo uso de 6 funciones. Para la primera función, ListaPorDepto(), se tiene que la complejidad es de O(NlogN), ya que utiliza la función de ordenamiento MergeSort, y esta es su complejidad en el peor caso. Para la segunda función, CalcularCostoEnvíoObra, tiene una complejidad O(k), al ser colo condicionales. Para la tercera función, CostoTodasObras, su complejidad es de O(N), ya que se realiza el recorrido cobre una lista. Para la cuarta función, ObrasMasAntiguas, se tiene un ordenamiento con MergeSort, que como ya se ha dicho, su complejidad es de O(NLogN). Para la quinta función, ObrasMasCaras, al igual que la anterior, hay un ordenamiento con MergeSort, entonces la complejidad también es O(NLogN). Por último, la complejidad de la última función, ArtistaEnObra, se tiene una complejidad de $O(N^2)$. Teniendo todo

esto en cuenta, se tiene que la mayor complejidad es de O(NLogN), pero la complejidad total es de O(NlogN)+ O(N)+O(K)+ O(NLogN)+ O(NLogN)+ O(N^2).

Ambientes de pruebas

	Máquina 1	**Máquina 2**
Procesadores	Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
Memoria RAM (GB)	8.00 GB	8,00 GB
Sistema Operativo	Windows 10 Pro 64-bits	Windows 10 Home Single Language 64-bits

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

Maquina 1

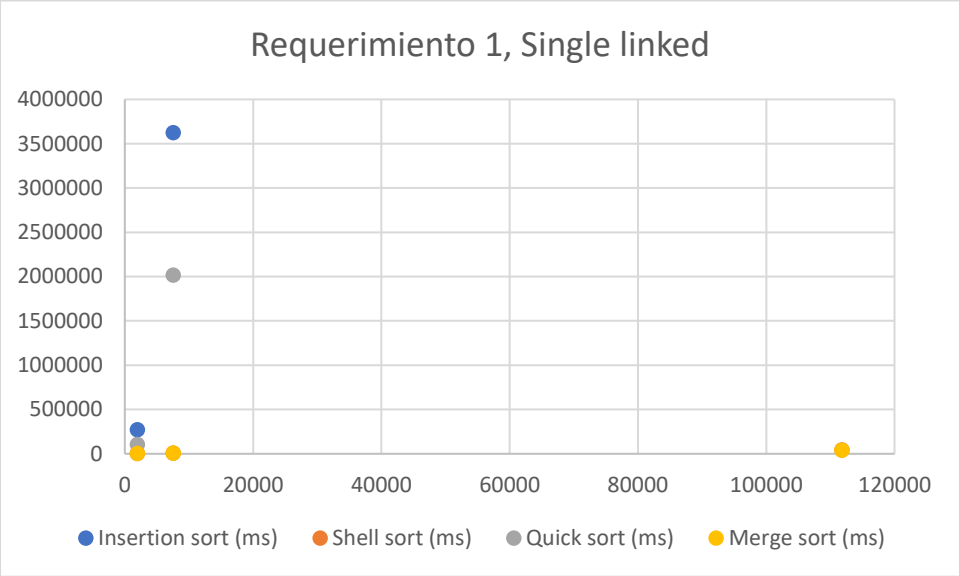
Para los cálculos de rendimiento en función de tiempo, fueron implementados algunos cambios en el código. En el view, se utilizó start_time y stop_time después de los input para encontrar el tiempo que tomó en completar con el requerimiento, este resultado se imprime en consola.

Para cada caso, se definía si se debía usar array_list o single-linked al momento de cargar los datos.

Para cada caso, se cambió manualmente el tipo de ordenamiento usado en todas las funciones (en todas las veces que aparecía) en el modelo. En la versión entregada del código se implementó MergeSort, pues según los resultados, tiende a ser la más eficiente (lo cual, por fortuna, concuerda con la teoría).

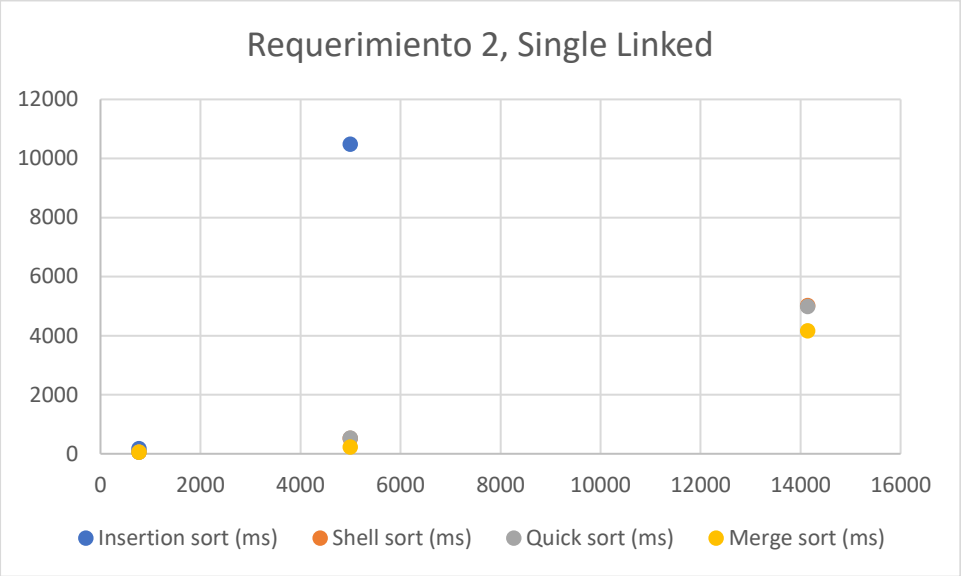
Requerimiento 1:

	Tamaño ArrayList	Insertion sort (ms)	Shell sort (ms)	Quick sort (ms)	Merge sort (ms)
Small	1948	1421,875	46,875	640,625	46,875
5pct	7572	12046,875	153,26	6740,568	140.625
80 pct	111781	106703,125	504,68	65321,89	421,875
	Tamaño Single-linked	Insertion sort (ms)	Shell sort (ms)	Quick sort (ms)	Merge sort (ms)
Small	1948	269984,375	800,46	101681,3	718,75
5pct	7572	3621498,245	5310,632	2014687,634	4718,75
80pct	111781		40532,61		38187,5



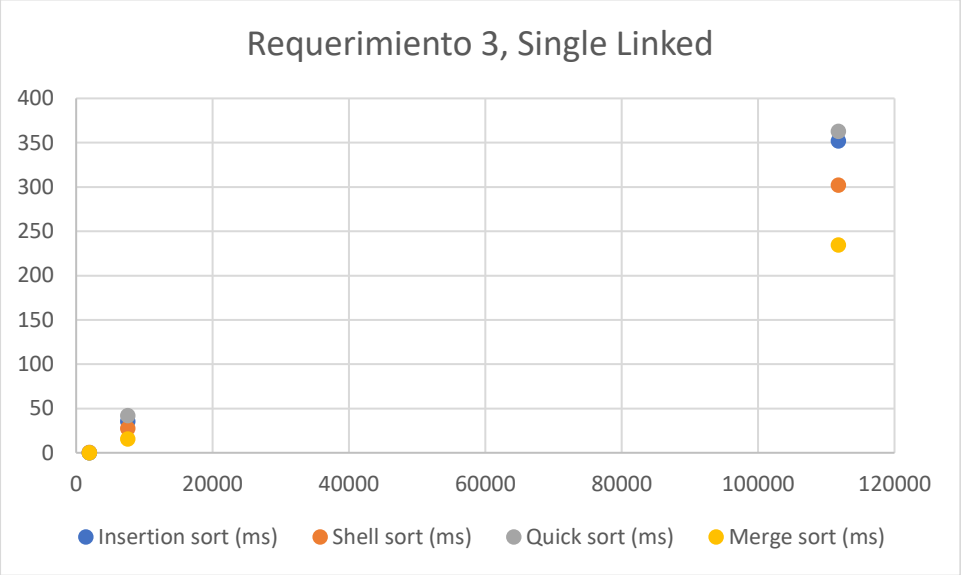
Requerimiento 2:

	Tamaño ArrayList	Insertion sort (ms)	Shell sort (ms)	Quick sort (ms)	Merge sort (ms)
Small	768	187,5	31,25	31,25	31,25
5pct	4996	10473,5	217,86	315,67	187,5
80pct	14143	1750637,96	4982,964	5315,864	4156,25
	Tamaño Single-linked	Insertion sort (ms)	Shell sort (ms)	Quick sort (ms)	Merge sort (ms)
Small	768	171,875	50,783	51,935	46.875
5pct	4996	10473,5	513,957	531,86	218,75
80pct	14143		5017,95	4978,365	4156,25



Requerimiento 3:

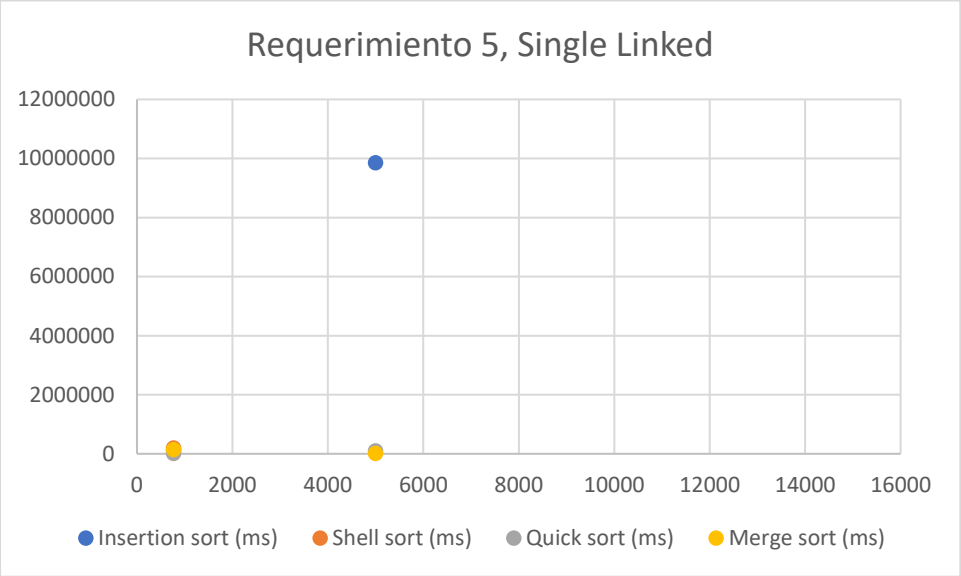
	Tamaño ArrayList	Insertion sort (ms)	Shell sort (ms)	Quick sort (ms)	Merge sort (ms)
Small	1948	0,12	0,15	0,12	0,12
5pct	7572	31,25	30,42	38,95	31,25
80pct	111781	300,27	287,96	310,95	265,625
	Tamaño Single-linked	Insertion sort (ms)	Shell sort (ms)	Quick sort (ms)	Merge sort (ms)
Small	1948	0,12	0,2	0,15	0,15
5pct	7572	35,38	27,498	41,861	15,625
80pct	111781	351,96	301,943	362,82	234,375



Requerimiento 5:

	Tamaño ArrayList	Insertion sort (ms)	Shell sort (ms)	Quick sort (ms)	Merge sort (ms)
Small	768	312,5	421,875	140,625	46.875
5pct	4996	35875	4972,316	1007,962	406,25
80pct	14143	399215,91	50319,963	10106,95	7781,25
	Tamaño Single-linked	Insertion sort (ms)	Shell sort (ms)	Quick sort (ms)	Merge sort (ms)
Small	768	100093,75	195306,951	873,52	140.625
5pct	4996	9853107,68		95317,931	10000
80pct	14143				


Requerimiento 5, Array List
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
0
2000
4000
6000
8000
10000
12000
14000
16000
Insertion sort (ms)
Shell sort (ms)
Quick sort (ms)
Merge sort (ms)


Requerimiento 5, Single Linked
12000000
10000000
8000000
6000000
4000000
2000000
0
0
2000
4000
6000
8000
10000
12000
14000
16000
Insertion sort (ms)
Shell sort (ms)
Quick sort (ms)
Merge sort (ms)