Informe Reto 1 – Grupo 04

Sergio Cañar – 202020383 – s.canar@uniandes.edu.co
Juan Martín Vázquez 202113214 – j.vasquezc@uniandes.edu.co
Juan Bernardo Parra 202021772 – j.parrah@uniandes.edu.co

Las pruebas de tiempos de ejecución fueron realizadas por Juan Bernardo Parra, la máquina empleada tenía las siguientes especificaciones:

	Máquina Utilizada
Procesadores	Intel(R) Core(TM) i7- 1065G7 CPU @ 1.30GHz 1.50 GHz
Memoria RAM (GB)	12.0 GB (11.7 GB utilizable)
Sistema Operativo	Windows 11 Home Single Language – 64 bits

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

Los requerimientos individuales fueron realizados por:

- Requerimiento 3 – Sergio Cañar
- Requerimiento 4 - Juan Martín Vásquez
- Requerimiento 5 – Juan Bernardo Parra

Requerimiento 1

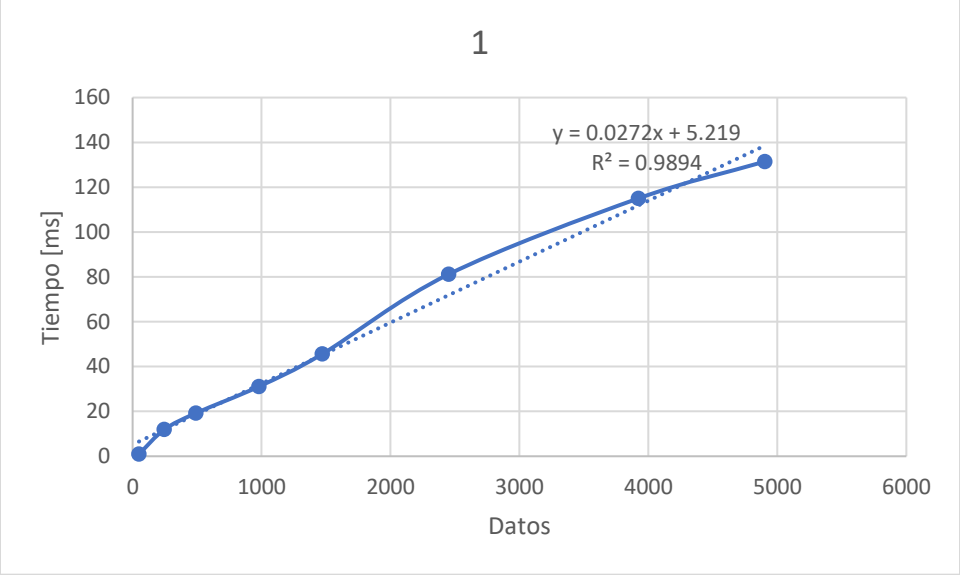
1. Análisis de complejidad – Teórico
   Debido a que en el requerimiento 1 únicamente se realiza un Merge Sort, su complejidad es de $O(n \log n)$.

2. Análisis de complejidad – Práctico

Si usa un arreglo como estructura de datos, se obtienen los siguientes resultados:

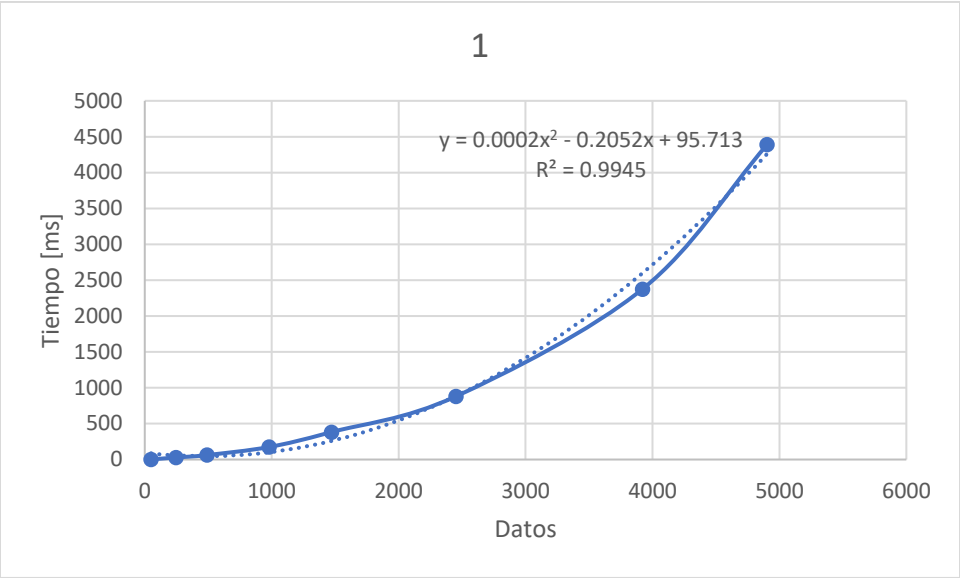
1 - Arreglo		
Tiempo	Número de datos	Tiempo [ms]
small	49	0.97
5%	245	11.87
10%	490	19.21
20%	980	31.12
30%	1470	45.58
50%	2451	81.182
80%	3922	115.01

100%	4903	131.45



Por el contrario, si se usa una Single Linked List como estructura de datos, los resultados son los siguientes:

1 - Single Linked		
Tiempo	Número de datos	Tiempo [ms]
small	49	2.036
5%	245	27.741
10%	490	62.171
20%	980	174.92
30%	1470	382.5
50%	2451	880.2
80%	3922	2377
100%	4903	4390



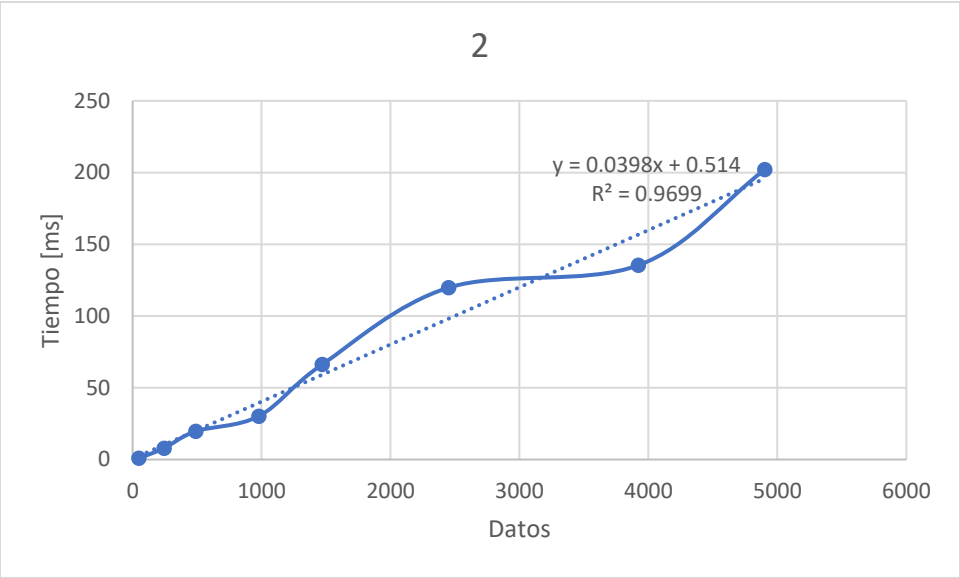
3. Análisis
   Como se puede observar, cuando los datos se organizan como un arreglo la gráfica de los datos obtenidos experimentalmente tiene una forma parecida a una función $n \log n$, como sugiere la teoría. Sin embargo, cuando los datos se almacenan como una Single Linked, podemos ver que la curva tiende a ser cuadrática. Sin embargo, y si bien es un dato que se excluye de la notación Big O, el factor que acompaña a la variable cuadrática es de 0.0002, lo que "reduce" el crecimiento de los datos en comparación a una cuadrática con factor 1.

Requerimiento 2

1. Análisis de complejidad – Teórico
   Al igual que en el requerimiento 1, el requerimiento 2 utiliza únicamente un Merge Sort. Por esta razón, su complejidad esperada en el peor caso es de $O(n \log n)$.
2. Análisis de complejidad – Práctico

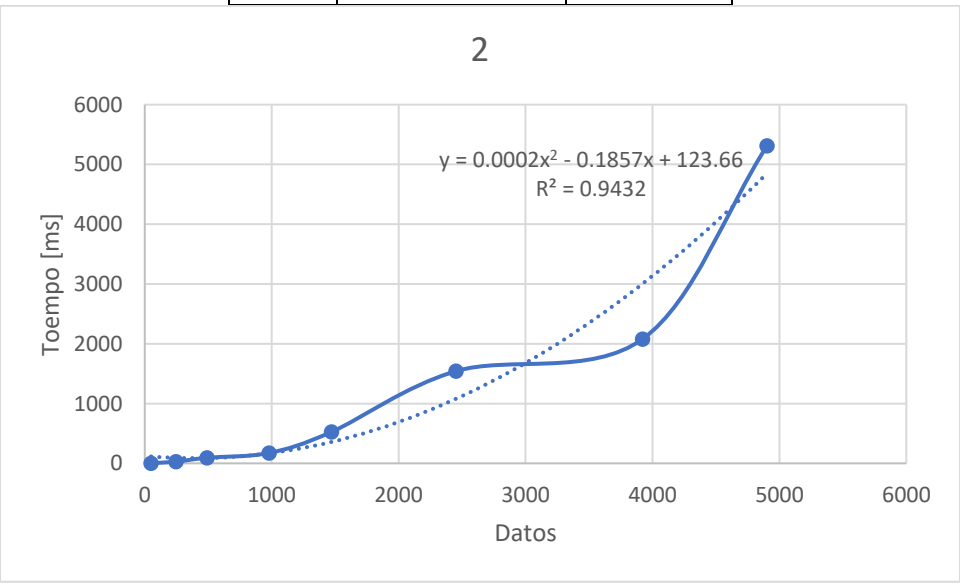
Si usa un arreglo como estructura de datos, se obtienen los siguientes resultados:

2 - Arreglo		
Tiempo	Número de datos	Tiempo [ms]
small	49	0.877
5%	245	7.933
10%	490	19.68
20%	980	30.291
30%	1470	66.23
50%	2451	119.83
80%	3922	135.37
100%	4903	202.076



Por el contrario, si se usa una Single Linked List como estructura de datos, los resultados son los siguientes:

2 - Single Linked		
Tiempo	Número de datos	Tiempo [ms]
small	49	3.33
5%	245	27.65
10%	490	92.83
20%	980	173.66
30%	1470	524.1
50%	2451	1541.4
80%	3922	2079
100%	4903	5314



3. Análisis
   Como se puede observar en la primera gráfica, cuando los datos se almacenan como un arreglo, el requerimiento 2 tiene una complejidad de $O(n)$. Sin embargo, al igual que en el requerimiento 1, cuando los datos son almacenados en una Single Linked, la complejidad es cuadrática. Si bien esto se aleja de la complejidad teórica, hay que mencionar que el factor que acompaña a la variable cuadrática es tan solo de 0.0002, lo que ayuda a "aplanar" la curva de rendimiento.

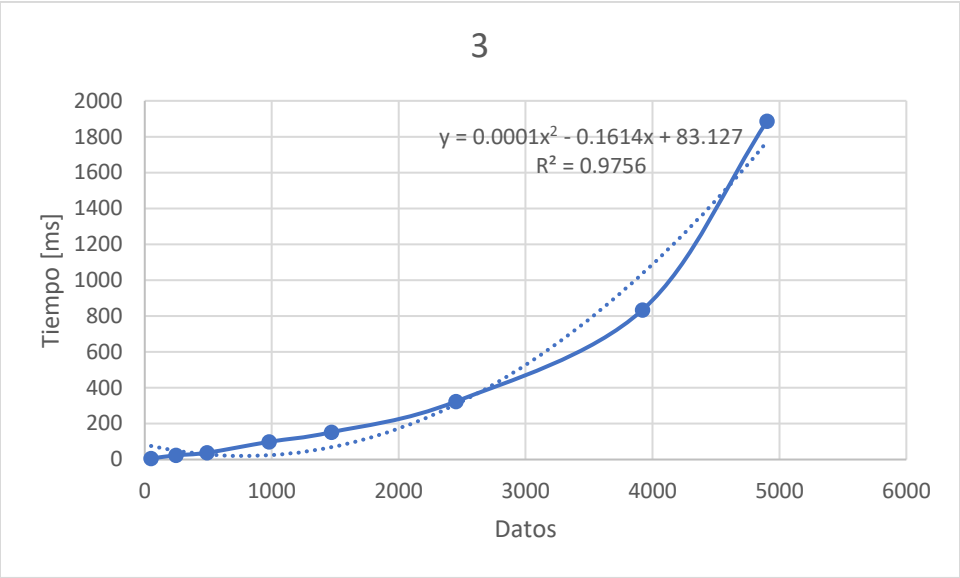
Requerimiento 3 – Sergio Cañar

1. Análisis de complejidad – Teórico
   El código de solución del requerimiento 3 en el model.py contiene múltiples ciclos *for* que iteran sobre distintas listas. Sin embargo, solo uno de estos *for* itera sobre el archivo de Excel con los datos (por lo que tiene complejidad O(N)). El resto de ciclos *for* itera sobre listas auxiliares creadas a partir del primer for. Además, dentro de los ciclos se encuentran únicamente funciones como addLast, las cuales tienen complejidad de $O(1)$ tanto en Array como en Single Linked List. Con este análisis, la complejidad del requerimiento 3 sería de $O(n)$. Sin embargo, al inicio del requerimiento se hace un ordenamiento con QuickSort, el cual tiene una complejidad de $O(n^2)$, la cual corresponderá también a la complejidad del requerimiento 3.

2. Análisis de complejidad – Práctico

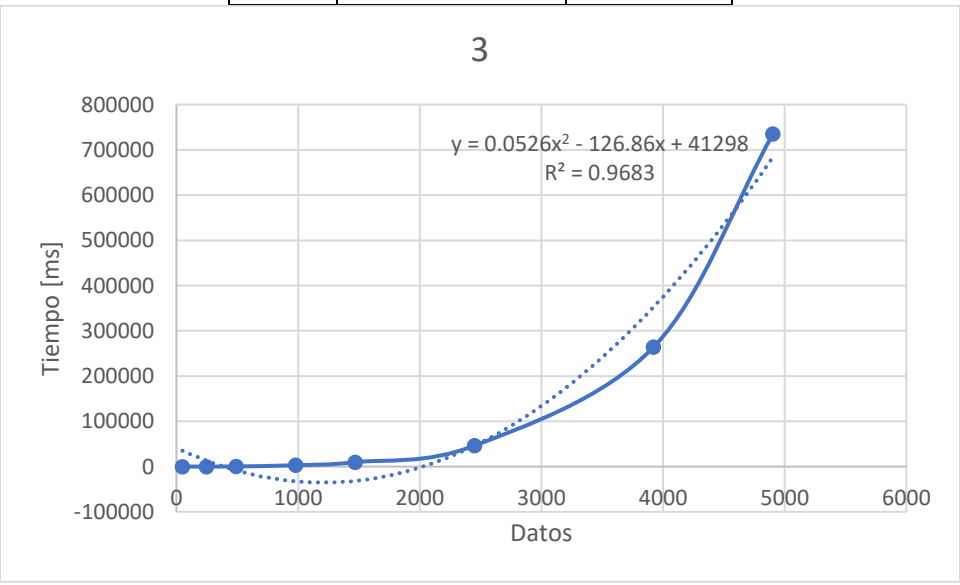
Si usa un arreglo como estructura de datos, se obtienen los siguientes resultados:

3 - Arreglo		
Tiempo	Número de datos	Tiempo [ms]
small	49	5.846
5%	245	23.34
10%	490	37.89
20%	980	98.67
30%	1470	152.34
50%	2451	323.8
80%	3922	834.5
100%	4903	1887.4



Por el contrario, si se usa una Single Linked List como estructura de datos, los resultados son los siguientes:

3 - Single Linked		
Tiempo	Número de datos	Tiempo [ms]
small	49	8.16
5%	245	84.23
10%	490	395.34
20%	980	3012.4
30%	1470	9822
50%	2451	46392
80%	3922	264078.575
100%	4903	735079



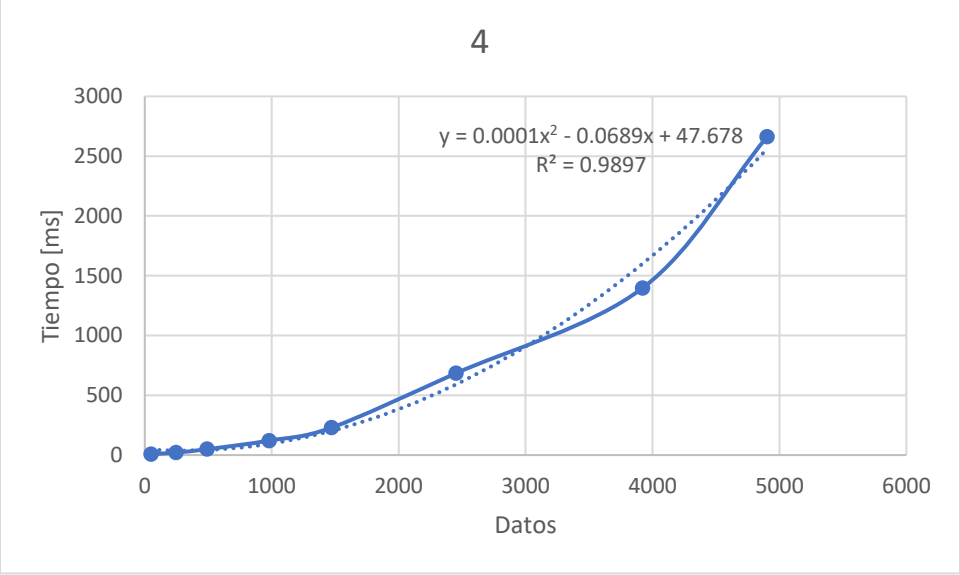
3. Análisis
   Las gráficas muestran que los resultados prácticos coinciden con lo enunciado teóricamente, pues el requerimiento demuestra tener complejidad $O(n^2)$.

Requerimiento 4 – Juan Martín Vásquez

1. Análisis de complejidad – Teórico
   Similar a lo que sucede en el requerimiento 3, la complejidad del requerimiento 4 es de $O(n^2)$. Esto se debe a que se realiza un QuickSort, cuyo peor caso tiene una complejidad cuadrática, y el resto de los ciclos tiene una complejidad máxima de $O(n)$. Por esta razón, y dado que la notación Big O conserva únicamente el término con mayor complejidad, podemos afirmar que la complejidad teórica del requerimiento es de $O(n^2)$
2. Análisis de complejidad – Práctico

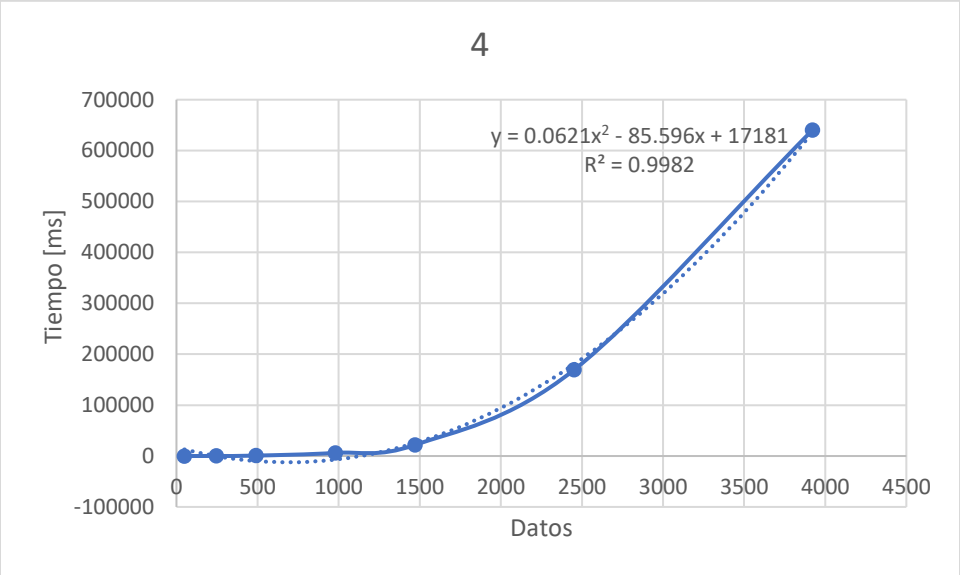
Si usa un arreglo como estructura de datos, se obtienen los siguientes resultados:

4 - Arreglo		
Tiempo	Número de datos	Tiempo [ms]
small	49	8.32
5%	245	20.98
10%	490	49
20%	980	121
30%	1470	229
50%	2451	684
80%	3922	1397
100%	4903	2663



Por el contrario, si se usa una Single Linked List como estructura de datos, los resultados son los siguientes:

4 - Single Linked		
Tiempo	Número de datos	Tiempo [ms]
small	49	11
5%	245	152
10%	490	820
20%	980	6219
30%	1470	21810.632
50%	2451	169178
80%	3922	640549
100%	4903	



3. Análisis

En este caso se evidencia como la gráfica corresponde a la complejidad algorítmica esperada puesto que se ajusta a una curva que corresponde a la complejidad temporal $O(n^2)$. Igualmente, se observa cómo en las pruebas realizadas con listas enlazadas hay un mayor tiempo de ejecución, lo anterior, se debe principalmente a la composición de esta estructura de datos.

Requerimiento 5 – Juan Bernardo Parra

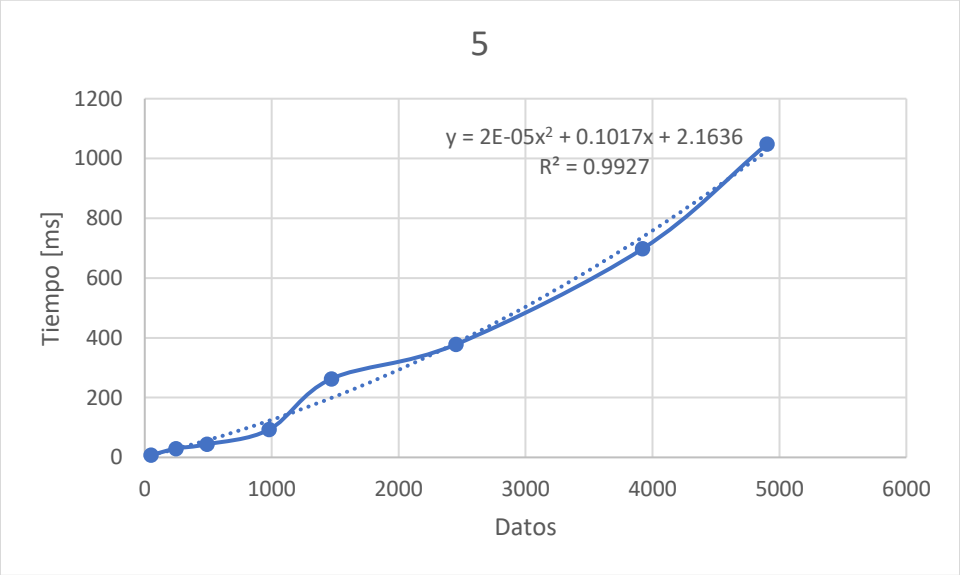
1. Análisis de complejidad – Teórico
   Si bien se usan múltiples ciclos para este requerimiento, ninguno supera una complejidad $O(N)$, puesto que en estos ciclos no existen operaciones de complejidad considerable. Por ejemplo, para el primer for del código en model, se itera sobre el archivo de Excel. Esto tiene como complejidad $O(n)$, puesto que se recorren todos los datos. Sin embargo, dentro del ciclo se hacen únicamente operaciones como addLast que tienen complejidades constantes. En este ciclo también se incluye un isPresent, pero este se realiza sobre una lista de tamaño 10, por lo que no tiene una complejidad considerable. Sin embargo, al inicio del código se realiza un Merge Sort

para organizar los datos, el cual tiene una complejidad de $O(n \log n)$, por lo que esta será la complejidad del algoritmo.

2. Análisis de complejidad – Práctico

Si usa un arreglo como estructura de datos, se obtienen los siguientes resultados:

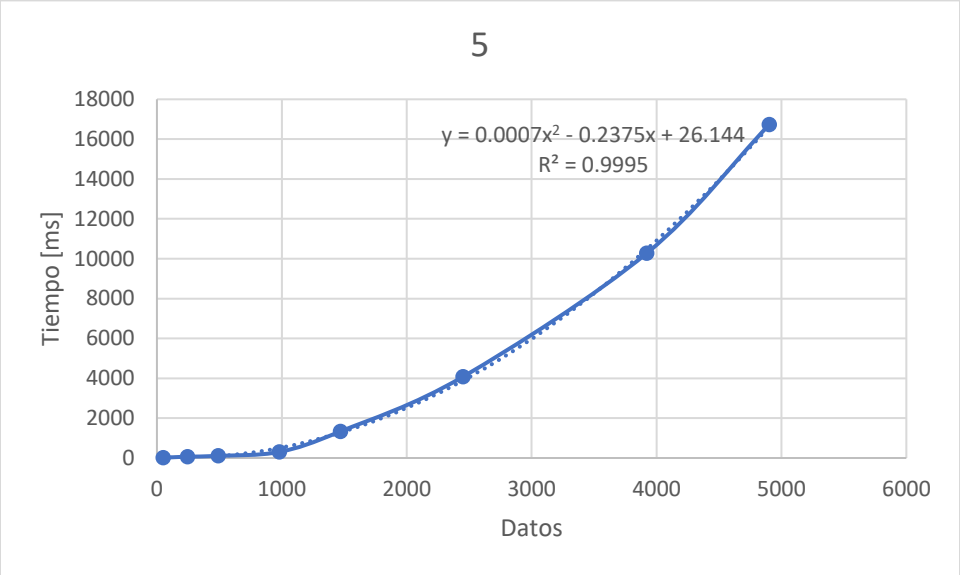
5 - Arreglo		
Tiempo	Número de datos	Tiempo [ms]
small	49	7.663
5%	245	29.34
10%	490	43.83
20%	980	93.18
30%	1470	262.8
50%	2451	377.94
80%	3922	697.88
100%	4903	1048.32



Por el contrario, si se usa una Single Linked List como estructura de datos, los resultados son los siguientes:

5 - Single Linked		
Tiempo	Número de datos	Tiempo [ms]
small	49	11.623
5%	245	67.12
10%	490	105.32
20%	980	304.55
30%	1470	1335
50%	2451	4080

80%	3922	10277
100%	4903	16735.383



3. Análisis
   Si bien las gráficas se ajustan a comportamientos de tendencia cuadráticos, al igual que en otros requerimientos los factores que acompañan a la variable de mayor grado son considerablemente pequeños (pues en este caso son 0.00005 y 0.0007 para arreglo y lista enlazada respectivamente). Esto hace que su crecimiento se "aplane" y sea más cercano al enunciado teóricamente.

Requerimiento 6

1. Análisis de complejidad – Teórico

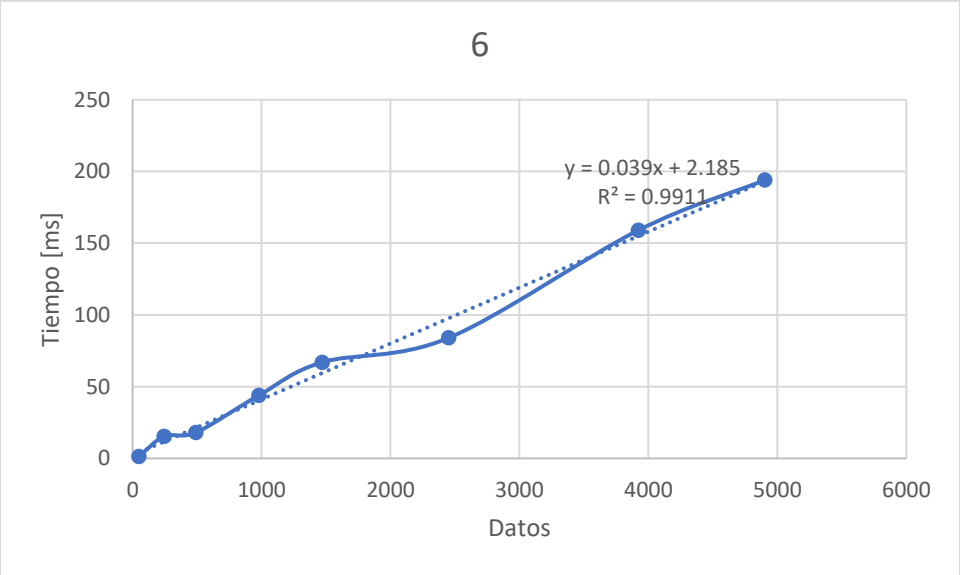
La complejidad del requerimiento 6 fue de O(k*Q^2*S). k corresponde a la cantidad de Código sector económico únicos en el año que entra por parámetro, Q hace referencia a la cantidad de entradas de los datos que tienen un Código sector económico común y por último S que hace referencia a la cantidad de subsectores que tenga algún Q. Es difícil establecer la complejidad de esta función, pues las variables propuestas anteriormente varían grandemente, por ejemplo, para algunos sectores es posible tener 1, 2 o hasta 3 subsectores, no obstante al observar como estas son fracciones de N datos se puede considerar que tiene un comportamiento tendiendo hacia lo lineal. Caba resaltar que existe un trozo de código que dependiendo del caso podría dar otra complejidad a este algoritmo, cuando se organizan todos los datos N en una lista de listas dado el código del sector económico esta tiene una complejidad O(N).

2. Análisis de complejidad – Práctico

Para simplificar el análisis, se contabilizó el tiempo del proceso solo con un año (2021).

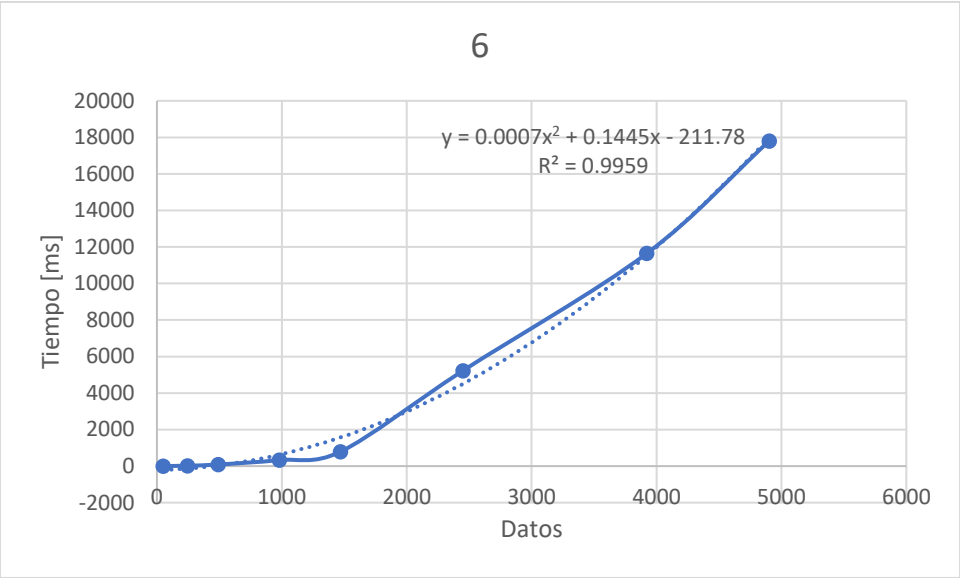
Si usa un arreglo como estructura de datos, se obtienen los siguientes resultados:

6 - Arreglo		
Tiempo	Número de datos	Tiempo [ms]
small	49	1.34
5%	245	15.43
10%	490	18
20%	980	44
30%	1470	67
50%	2451	84
80%	3922	159
100%	4903	194



Por el contrario, si se usa una Single Linked List como estructura de datos, los resultados son los siguientes:

6 - Single Linked		
Tiempo	Número de datos	Tiempo [ms]
small	49	3
5%	245	25
10%	490	88
20%	980	332
30%	1470	791
50%	2451	5229
80%	3922	11646
100%	4903	17808



3. Análisis

Lo planteado en el análisis teórico se puede comprobar observando el comportamiento de la función, pues esta tiene una tendencia lineal como se había predicho. Se puede apreciar como en la regresión realizada para los datos de ARRAY LIST estos siguen mejor lo predicho que la regresión a parir de los datos de SINGLE LINKED esto muy probablemente se deba a que al dividir los datos en una lista de listas por sector económico iterar sobre SINGLE LINKED sea mas costoso, no obstante se puede observar como el termino cuadrático tiene un coeficiente muy bajo lo que indica que tiene un comportamiento casi lineal.

Requerimiento 7

1. Análisis de complejidad – Teórico

En este requerimiento la complejidad temporal del algoritmo está dada por el uso del algoritmo de ordenamiento *quickSort* cuya complejidad temporal es $O(nlog(n))$ en el mejor caso y $O(n^2)$ en el peor caso. Además, es de consideración las iteraciones que se realizan en para identificar la posición de los años iniciales y finales ingresados como parámetro de este requerimiento cuya complejidad es $O(n)$. Por lo que a nivel teórico la complejidad esperada de este algoritmo es $On(nlog(n))$ o $O(n^2)$ en el peor de este.

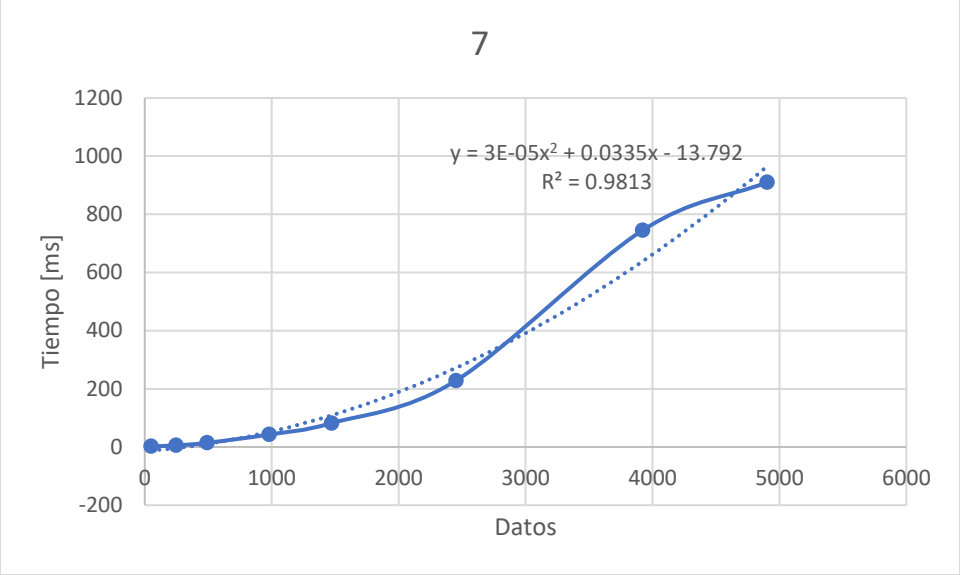
2. Análisis de complejidad – Práctico

Para estandarizar el proceso, todas las mediciones fueron hechas desde el año 2020 hasta el 2021 con el top 3.

Si usa un arreglo como estructura de datos, se obtienen los siguientes resultados:

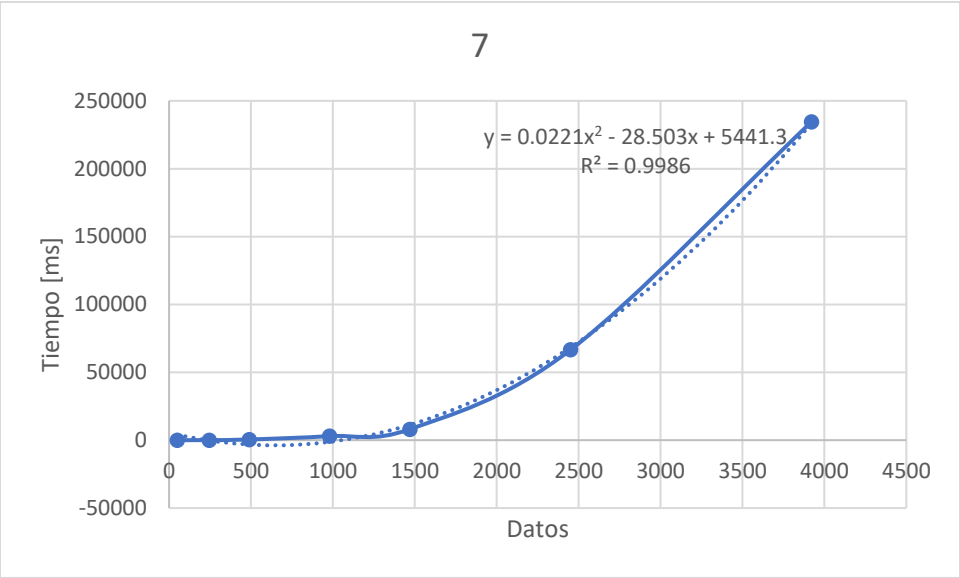
7 - Arreglo		
Tiempo	Número de datos	Tiempo [ms]
small	49	2.47

5%	245	5.5
10%	490	14.39
20%	980	42.9
30%	1470	82
50%	2451	228
80%	3922	745
100%	4903	910



Por el contrario, si se usa una Single Linked List como estructura de datos, los resultados son los siguientes:

7 - Single Linked		
Tiempo	Número de datos	Tiempo [ms]
small	49	2.44
5%	245	65
10%	490	513
20%	980	3013
30%	1470	8116
50%	2451	66637
80%	3922	234645
100%	4903	



3. Análisis

Considerando los resultados de las pruebas realizadas para este requerimiento corresponden al análisis teórico realizado puesto que la curva que mejor representa los tiempos de ejecución del requerimiento es la de $O(n^2)$ cómo se puede observar en la regresión realizada, especialmente para la gráfica que corresponde a la ejecución de tiempo con la lista enlazada. Por otro lado, con respecto a la toma de datos con los arreglos se puede afirmar que esta estructura es mucho más idónea para este requerimiento debido a que si bien tienen una complejidad temporal igual con respecto a la lista enlazada, se aprecia que tiene una mejor tolerancia a una mayor cantidad de datos que la lista enlazada.

Análisis general del reto.

En primer lugar, al analizar los resultados obtenidos, teniendo en cuenta la estructura de datos implementada junto con los tiempos de complejidad de los algoritmos de ordenamiento seleccionado y el tiempo de ejecución de los requerimientos se logró concluir que *merge sort* es uno de los mejores algoritmos de ordenamiento gracias a su garantía en su complejidad que siempre es O(nlogn). Adicionalmente, se logró observar en rasgos generales que la carga de datos junto con el tiempo de ejecución de los algoritmos era mucho más demorada al momento de implementar una lista encadenada a una lista conformada por un arreglo.