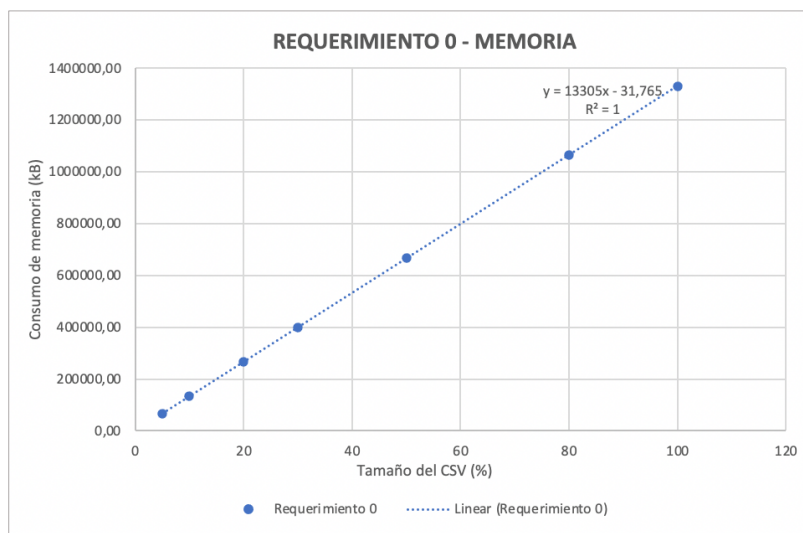
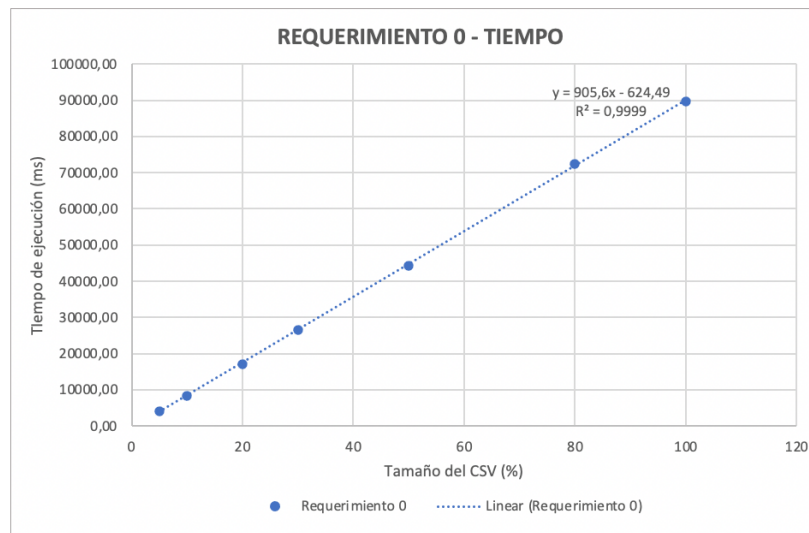


## Documento de análisis Reto 2

Nombres integrantes	Requerimiento	Código	Correo
Mariana Díaz Arenas	Requerimiento 2	202020993	<a href="mailto:m.diaza2@uniandes.edu.co">m.diaza2@uniandes.edu.co</a>
Nicole Murillo Fonseca	Requerimiento 3	202025521	<a href="mailto:n.murillof@uniandes.edu.co">n.murillof@uniandes.edu.co</a>

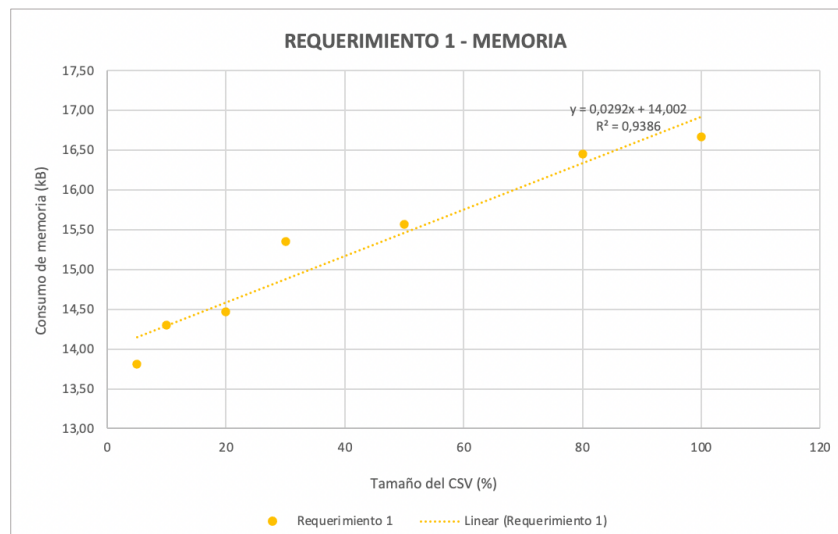
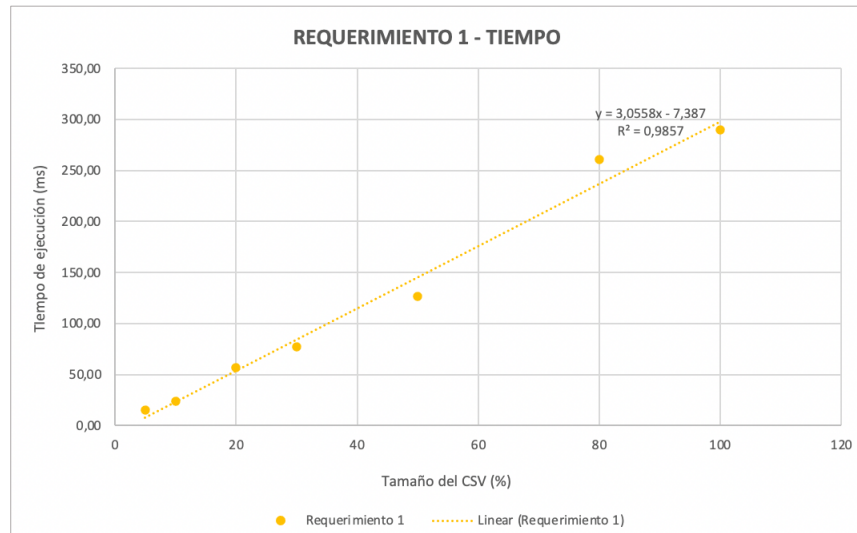
### Análisis de la complejidad de las funciones del Reto 2:

#### Gráfica requerimiento 0: Cargar información del catálogo



El propósito del requerimiento 0 era cargar los datos del documento csv, por tanto, tiene sentido que el tiempo de carga del algoritmo y el consumo de memoria dependieran completamente del número de datos que contenía el archivo ingresado. En este orden de ideas, al obtener una gráfica con una representación lineal para el consumo de datos y con un orden de crecimiento lineal  $O(n)$  para el tiempo, es posible suponer que la carga de datos fue exitosa en la inicialización del catálogo.

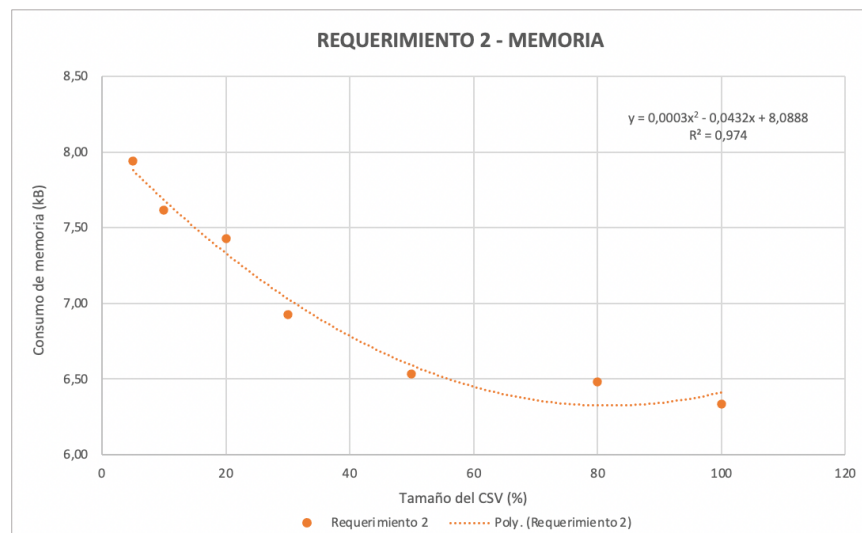
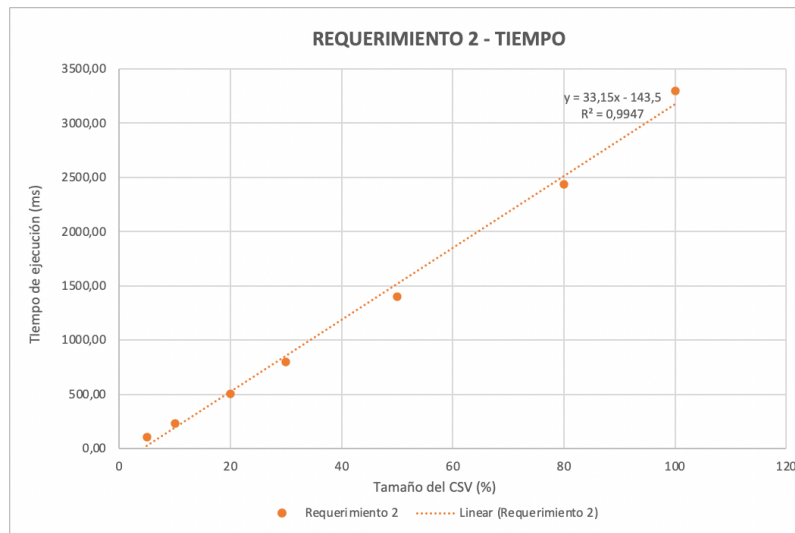
**Gráfica requerimiento 1:** Videos n con más *views* en un país determinado dada una categoría específica



Mediante la gráfica es posible observar que el orden de crecimiento es lineal para el consumo de datos y  $O(n)$  para el tiempo, ya que, se evidencia como el tiempo que toma el algoritmo y el consumo de memoria dependían del tamaño del csv, es decir, del número de datos que se estuvieran cargando. Ahora bien, para este requerimiento se usaron mapas creados previamente en la inicialización del catalogo para las categorías

y los países; sin embargo, las gráficas demuestran que el tiempo y consumo de datos aumentaban proporcionalmente según el tamaño del archivo csv, ya que, se hizo uso de la función `sort()`; por tanto, entre más grande era el documento a cargar, más elementos tenía que organizar esta función, más tiempo tomaba el programa y más memoria consumía.

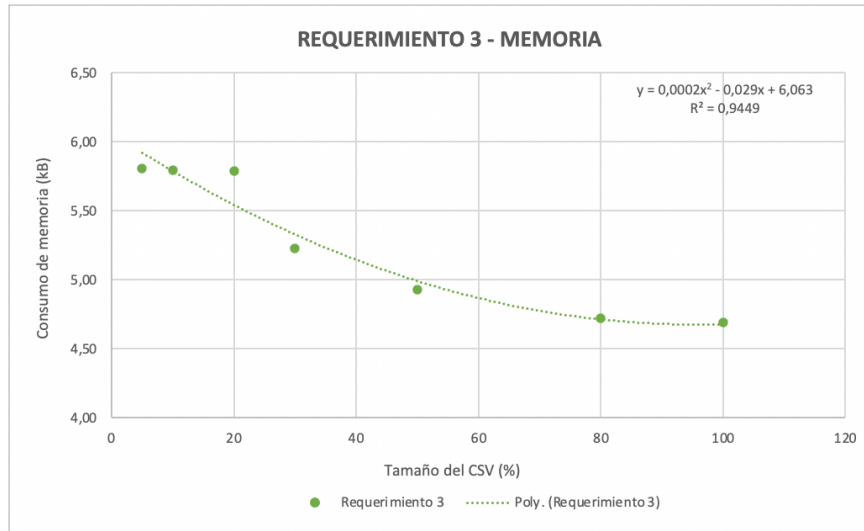
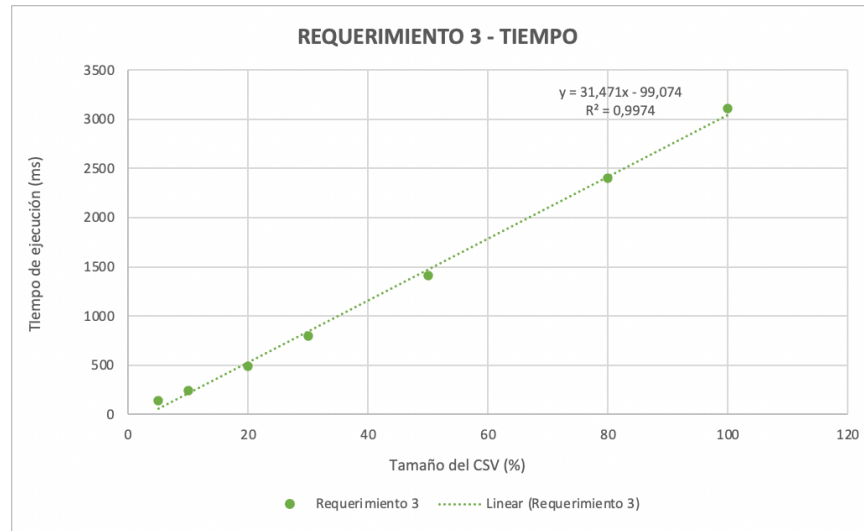
### Gráfica requerimiento 2: Video *trending* por más días en un país determinado



En este caso, solo se hizo uso del mapa de países; en general, para el requerimiento 2 del reto 2 usamos de nuevo la función `sort()` para organizar los videos por títulos y otra iteración sencilla, comparando los id de los videos, para poder contar el número de días que un video estuvo *trending* y así encontrar el video más frecuente. Es debido a las iteraciones que, se evidencia un orden de crecimiento lineal  $O(n)$  para el tiempo. Ahora bien, el consumo de memoria tuvo una tendencia cuadrática; un posible motivo de esto es que,

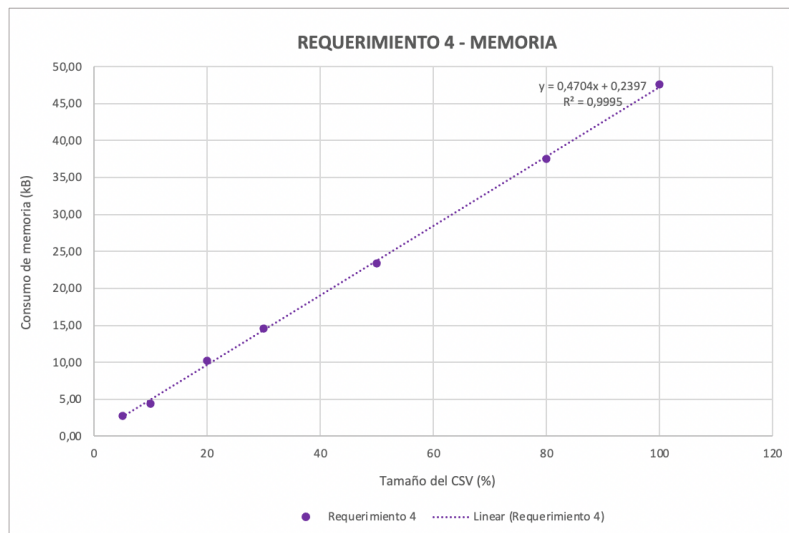
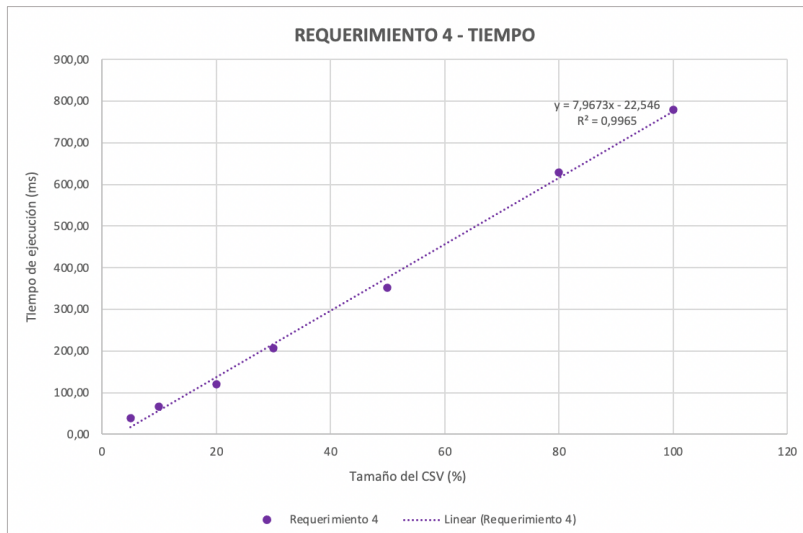
después de cargar el catálogo se podía utilizar menos memoria porque al buscar solo se iba a traer la lista que le correspondía a la llave en la basa de datos, el tiempo aumentó con respecto al tamaño del archivo, pero como se pudo observar la memoria se redujo.

### Gráfica requerimiento 3: Video *trending* por más días para una categoría específica



Para el requerimiento 3, se hizo un procedimiento muy similar al que se describió en el requerimiento 2, pero en este caso se hizo uso del mapa por categorías. Debido a que también se usó la función `sort()` y las mismas iteraciones sencillas para encontrar el video más *trending*, el orden de crecimiento del tiempo de ejecución fue lineal  $O(n)$ . Así mismo, al igual que en el requerimiento 2, el consumo de memoria también tuvo una tendencia cuadrática por el mismo motivo.

**Gráfica requerimiento 4:** Videos n con más *likes* en un país determinado y con un *tag* específico



En este requerimiento, más que en los anteriores, se observa claramente el ajuste lineal de las gráficas al aumentar el tamaño del archivo csv tanto para el tiempo de ejecución como para el consumo de memoria, por tanto, podemos inferir que el orden de crecimiento es, de nuevo para el tiempo, lineal  $O(n)$ . Las dos posibles razones de esto son, el uso de la función `sort()` para ordenar los videos por *likes* y una iteración sencilla para encontrar los videos que tuvieran ese *tag*. Decidimos no usar un mapa para los *tags*, porque consideramos que, al ser tan variados, la carga inicial iba a tomar demasiado tiempo y, en general, el algoritmo iba a ser menos eficiente.

Observaciones generales de la resolución del Reto 2 en relación con la solución dada para el Reto 1:

Tabla de tiempo de ejecución y consumo de memoria para cada requerimiento en el Reto 2:

Requerimiento 0			Requerimiento 1		
Tamaño CSV [%]	Tiempo de Ejecución [ms]	Consumo de memoria [kB]	Tamaño CSV [%]	Tiempo de Ejecución [ms]	Consumo de memoria [kB]
5	4141,65	66699,89	5	14,88	13,81
10	8380,59	133142,92	10	24,25	14,30
20	17192,25	265840,21	20	56,51	14,47
30	26679,29	398593,57	30	77,02	15,35
50	44319,38	665719,88	50	126,60	15,57
80	72465,94	1064423,91	80	260,39	16,45
100	89600,37	1330399,16	100	290,12	16,67
Requerimiento 2			Requerimiento 3		
Tamaño CSV [%]	Tiempo de Ejecución [ms]	Consumo de memoria [kB]	Tamaño CSV [%]	Tiempo de Ejecución [ms]	Consumo de memoria [kB]
5	107,61	7,94	5	142,35	5,81
10	232,73	7,62	10	239,70	5,80
20	502,91	7,43	20	493,37	5,79
30	798,71	6,92	30	795,67	5,23
50	1400,30	6,53	50	1407,21	4,93
80	2435,28	6,48	80	2400,03	4,72
100	3297,17	6,33	100	3112,17	4,69
Requerimiento 4					
Tamaño CSV [%]	Tiempo de Ejecución [ms]	Consumo de memoria [kB]	Factor de carga:	0.5	
5	39,17	2,72	Tipo de mapa:	"PROBING"	
10	66,13	4,41			
20	120,42	10,20			
30	207,08	14,57			
50	351,55	23,41			
80	629,48	37,56			
100	778,71	47,57			

Tabla de tiempos para cada requerimiento en el Reto 1:

Requerimiento 0		Requerimiento 1	
Tamaño CSV [%]	Tiempo [ms]	Tamaño CSV [%]	Tiempo [ms]
5	725,18	5	48,14
10	1493,48	10	100,71
20	3001,09	20	188,77
30	4285,83	30	264,29
50	7305,78	50	437,59
80	12120,75	80	824,65
100	15702,78	100	970,77
Requerimiento 2		Requerimiento 3	
Tamaño CSV [%]	Tiempo [ms]	Tamaño CSV [%]	Tiempo [ms]
5	122,17	5	115,43
10	265,01	10	263,88
20	544,49	20	520,12
30	845,83	30	838,71
50	1459,95	50	1451,08
80	2464,65	80	2451,61
100	3332,71	100	3169,35
Requerimiento 4			
Tamaño CSV [%]	Tiempo [ms]		
5	89,41		
10	182,12		
20	368,38		
30	546,17		
50	839,87		
80	1506,15		
100	2273,89		

En general, si se observan y comparan los tiempos de ejecución del Reto 1 y del Reto 2, se puede evidenciar el uso de tablas de Hash o mapas. Para empezar, el tiempo de carga del catálogo fue significativamente mayor en el Reto 2; el motivo más claro de esto, fue que ahora el catalogo se inicializaba tanto con una TAD lista como con 4 mapas diferentes. A pesar de que usamos *Linear Probing* con un factor de carga de 0.5 (el factor ideal), esto no evitó que el tiempo de inicialización del catálogo fuera casi 6 veces mayor que en el Reto 1. En el Reto 1 cargar el catalogo tomó 15702,78 ms con el archivo csv al 100%, mientras que en el Reto 2, tomó 89600,37 ms.

Ahora bien, gracias a que el catalogo ya tenía diferentes mapas con las respectivas categorías, países y categoría/países como llaves, los requerimientos fueron temporalmente más eficaces. Para el requerimiento 1, por ejemplo, hicimos un mapa que tenía como llaves el país y la categoría, de tal manera que solo usamos la función `sort()` para ordenar los videos según sus *likes* y el mapa mencionado previamente. Lo anterior, se puede observar en las tablas de tiempos, pues para el Reto 1 el requerimiento 1 tuvo un tiempo de 970,77 ms con el archivo completo, mientras que en el Reto 2, este mismo requerimiento, tomó solo 290,12 ms.

Respecto a los requerimientos 2 y 3, el cambio en términos del tiempo no fue tan evidente, debido a que solo se usó un mapa en cada función; respectivamente, uno con los países de llaves y uno con las categorías de llaves. Así pues, usando el archivo completo, en el Reto 1 (requerimiento 2) el tiempo total fue de 3332,71 ms mientras que en el Reto 2 fue de 3297,17 ms; por su lado, en el requerimiento 3 (Reto 1) el tiempo total fue de 3169,35 ms mientras que en el Reto 2 fue de 3112,17 ms.

Finalmente, en el requerimiento 4 el cambio fue bastante notable, pues al parecer usar el mapa de países redujo el tiempo del Reto 1 de 2273,89 ms a 778,71 ms en el Reto 2 usando el 100% del csv.