Análisis de resultados Reto 1

François Morales – f.moraless - 202211168
Santiago Navarrete – s.navarretev - 202211202
Gabriela García – g.garcias2 – 202210869

Introducción

El Reto 1 se basa en el manejo de un gran volumen de datos con el fin de realizar requerimientos útiles de búsqueda y análisis de ellos. Es importante resaltar que el tipo de estructura usado para este reto fue ARRAY_LIST y, cuando así se requería, se utilizó únicamente la función de ordenamiento MERGE SORT, debido a su cualidad de estabilidad.

Por otro lado, con relación a los tiempos de ejecución, el reto se ejecutó en un solo computador cuyas características se describen a continuación:

- Procesador: AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz
- Memoria Ram: 16,0 GB (15,4 GB utilizable)
- Sistema Operativo: Sistema operativo de 64 bits, procesador x64 (Windows 11)

Requerimientos

Requerimiento 1 (Grupal)

- Descripción:

Entrada	catálogo, año inicial, año final
Salida	Lista de los contenidos lanzados en dado rango de años
Implementado	Si

- Complejidad: O(n+NlogN)

  Pasos:

  1. Se crea una variable con el catálogo de todas las producciones y se crea una lista vacía

  Declaración de Variables = O(2)

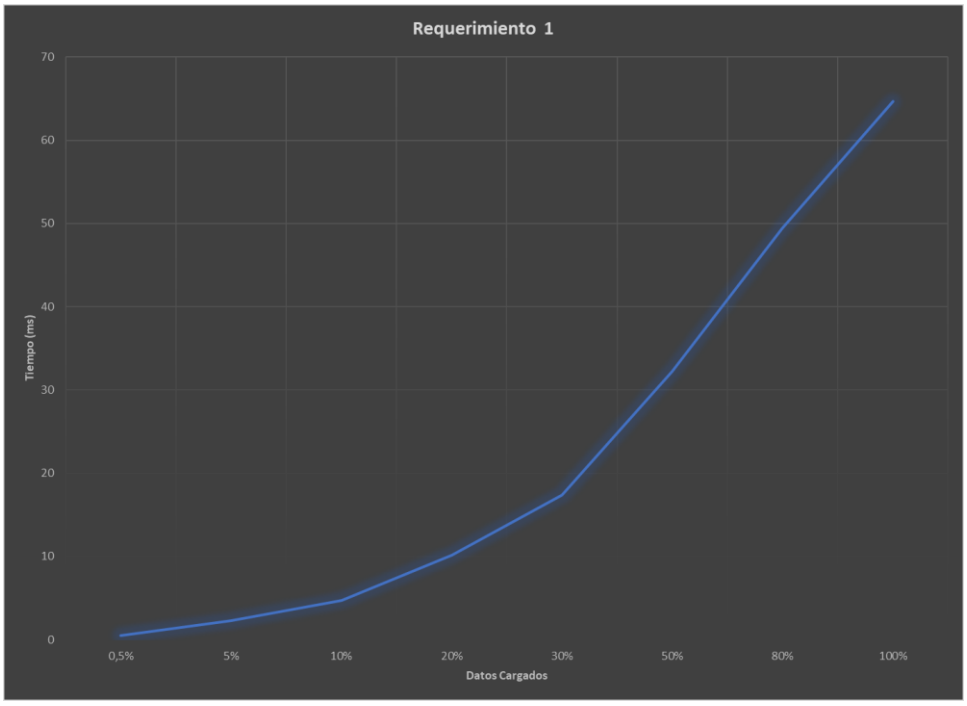
  2. Se realiza un ciclo para verificar que contenidos están dentro del rango de años

Ciclo=O(n)

3. Se ordena la lista del contenido encontrado por rango de años

Merge sort = O(Nlog(n))

- Tiempos de ejecución



Requerimiento 2 (Grupal)

- Descripción

Entrada	catálogo, fecha inicial, fecha final
Salida	Lista de los contenidos añadidos a las plataformas en dado rango de fechas
Implementado	Si

- Complejidad: O(n+1 +NlogN)

Pasos:

1. Se crea una variable con el catálogo de todas las producciones y se crea una lista vacía

Declaración de Variables = O(2)

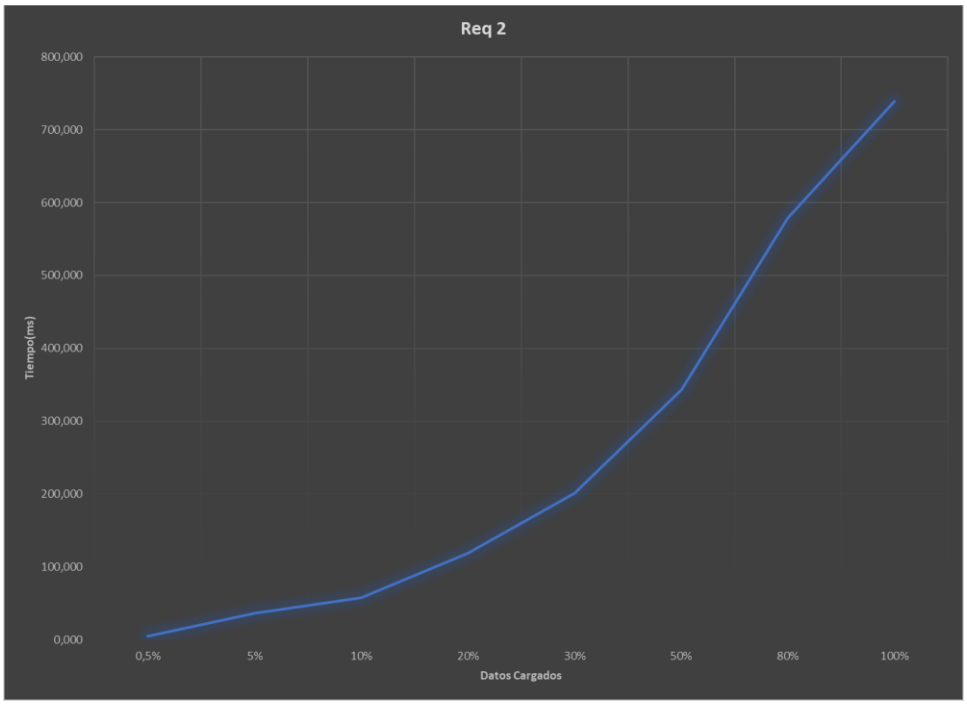
2. Se realiza un ciclo para verificar que contenidos están dentro del rango de fechas

Ciclo= O (n+1)

3. Se ordena la lista del contenido encontrado por rango de años

Merge sort = O (NLogN)

- Tiempos de ejecución



Requerimiento 3 (François)

- Descripción

Entrada	Catálogo, actor por buscar
Salida	Lista con la información del actor, numero de películas en donde aparece el actor, numero de shows en donde aparece el actor
Implementado	Si

- Complejidad: O (N + Nlog(N) + 2)

  Pasos

  1. Se declara una variable con todos los elementos del catálogo, una con el número de películas, una con el número de shows y otra que contenga y el tamaño del catálogo, luego se crea una lista vacía en donde se irá agregando la información del actor que se desea buscar.

  Declaración de variables = O (5)

2. Se crea un ciclo con el rango del tamaño de la lista y se toma elemento por elemento del arreglo

Declaración de variables = O (2)

Sentencia de asignación = O (2N)

Comparación mayor que = O (N+1)

Orden de crecimiento = O (N)

3. Se coje el elemento dado por el paso anterior y se mira si el actor se encuentra entre la información de la lista, si es así se agrega el elemento a la lista del primer paso, además se observa si es una película o un show de televisión.

Sentencia de asignación = O (N)

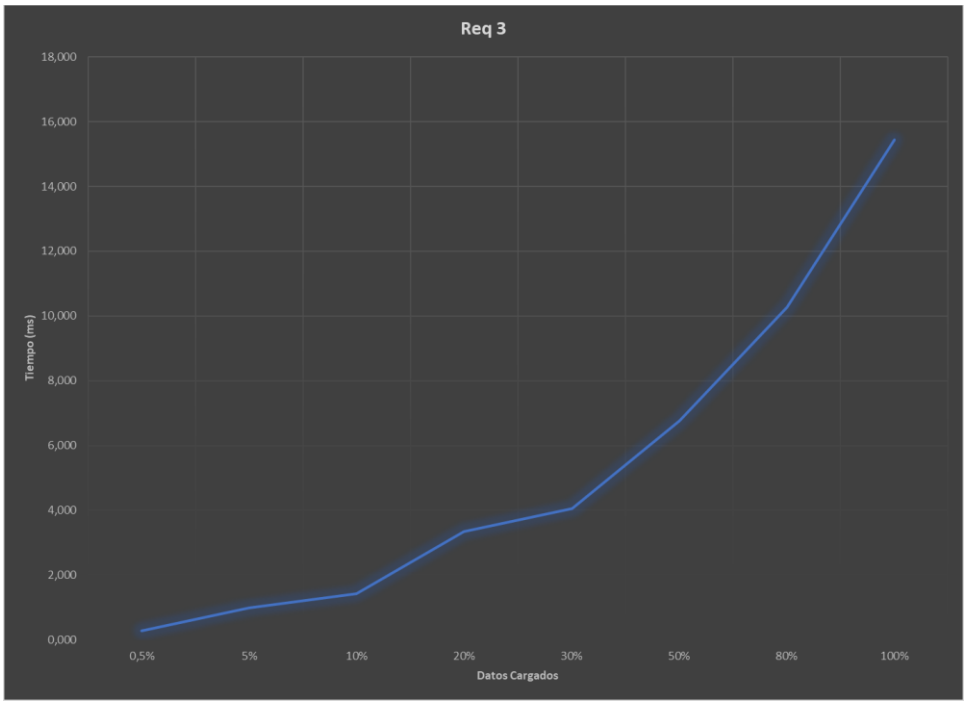
Comparación = O (2N)

Orden de crecimiento = O (N)

4. Finalmente se ordena la lista final con el algoritmo merge sort.

Orden de crecimiento = O (Nlog(N))

- Tiempos de ejecución



Requerimiento 4 (Gabriela)

- Descripción

Entrada	Catálogo, genero
Salida	num_movies (número de películas con el género deseado), num_shows (número de tv shows con el género deseado), lista de los contenidos de dado género
Implementado	Si

- Complejidad

  Numero de películas del género:

  Sentencia de asignación: O (3)

  Recorrido: O (n+1)

  Numero de shows del género:

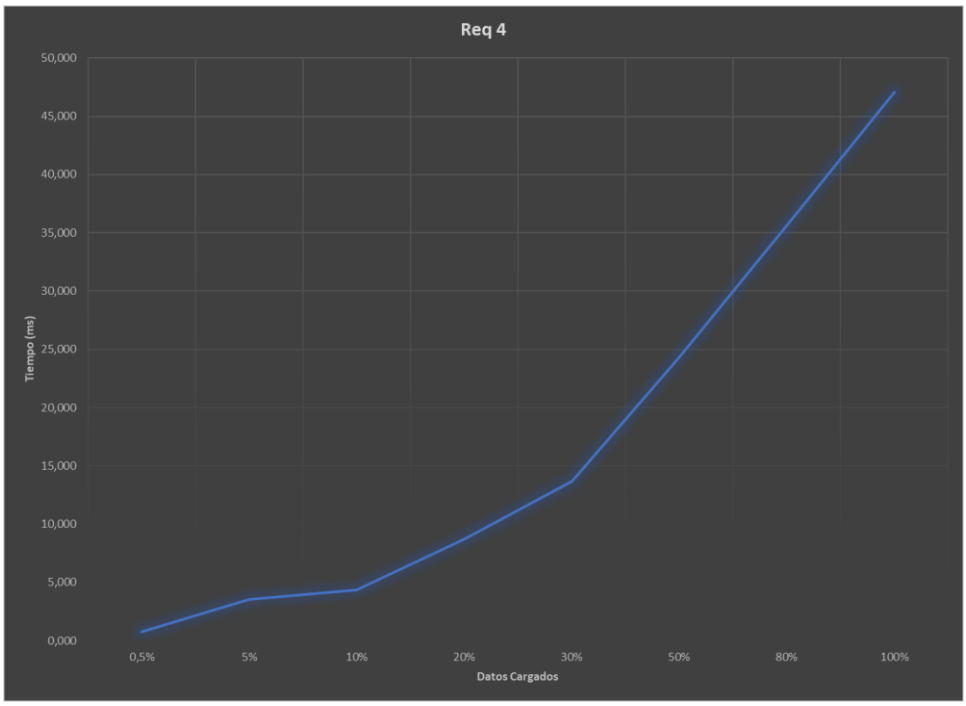
  Sentencia de asignación: O (3)

  Recorrido: O (n+1)

  Creación de lista con el género filtrado y listas de primeros 3 y últimos 3

  1) O (2)
  2) O (2N) size
  3) O (N+1)
  4) Merge: O (NlogN)
  5) O (N)
  6) O (N)

- Tiempo de ejecución:



Requerimiento 5 (Santiago)

- Descripción

Entrada	catálogo, país
Salida	Lista de los contenidos realizados en dado país
Implementado	Si

- Complejidad: O (n + 2NlogN)

  Pasos:

  1. Se crea una variable con el catálogo de todas las producciones y se crean dos listas vacías, una de tipo películas y otra de tipo programas de televisión

  Declaración de Variables = O (3)

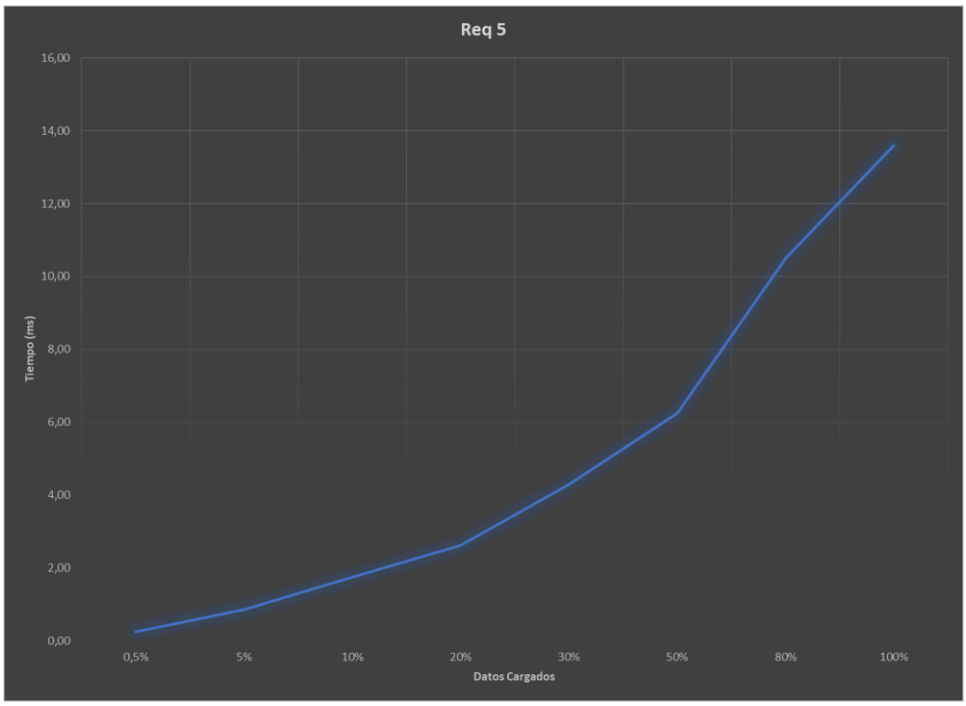
  2. Se realiza un ciclo para verificar que contenidos son del país que entró por parámetro

  Ciclo= O (n)

  3. Se ordenan las dos listas del contenido encontrado por país (movies y tv shows)

  Merge sort = O (2 NLogN)

- Tiempos de ejecución



Requerimiento 6 (Grupal)

- Descripción

Entrada	catálogo, director
Salida	Cantidad de Movies y TV Shows dirigidas por director, Cantidad de producciones por género dirigidas por director, Cantidad de producciones por plataforma dirigidas por director, lista de las películas ordenadas en orden cronológico dirigidas por director
Implementado	Si

- Complejidad:

  Sentencia de asignación: O(3)

  O(N) size

  O(NLogN) mergesort

  Creación del diccionario numero_generos_autordic

  Sentencia de asignación: O(3)

O(N+2)

O(n^2+2)

diccionario a lista:

O(N)

Creación de plataformasdic:

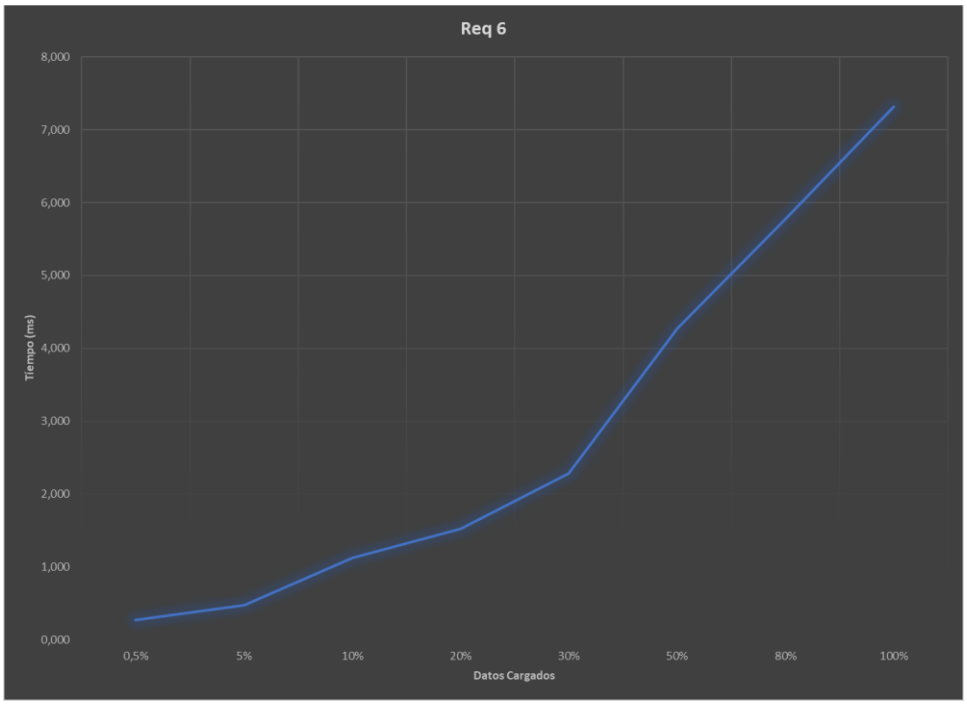
O(1)

O(N+1)

diccionario a lista:

O(N)

- Tiempos de ejecución



Requerimiento 7 (Grupal)

- Descripción

Entrada	Catálogo, numero top
Salida	Lista del top dado de géneros con su información como: cantidad por plataforma, cantidad entre cada tipo de producción (Movie, Tv Show)

Implementado	Si

- Complejidad: O (N^2 + Nlog(N) + 1)

  Pasos:

  1. Primero que todo se asigna una variable al catálogo con la información de las películas y series de televisión y segundo se crea un diccionario vacío en donde si irán agregando los datos del TOP(N) de géneros solicitados.

  Declaración de variables = O (2)

  Sentencia de asignación = O (2)

  2. Se crea un ciclo con el rango de la longitud de la lista que contiene la información del catálogo y dentro de este se declaran cinco variables para separar los datos y posteriormente compararlos: en la primera variable se toma cada elemento de la lista dependiendo su posición, la segunda variable es el servicio de streaming del elemento, la tercera su tipo, la cuarta su género y en la quinta se separa individualmente cada genero del elemento.

  Declaración de variables = O (5)

  Sentencia de asignación = O (5N)

  Comparación mayor que = O (N+1)

  Orden de crecimiento = O (N)

  3. Para el tercer paso se crea un ciclo con el rango del arreglo con los géneros separados, y por la posición se mira si el género del elemento está o no en el diccionario, si no está se crea cada llave con su valor y si está se suman los valores.

  Declaración de variables = O (10)

  Sentencia de asignación = O (12N)

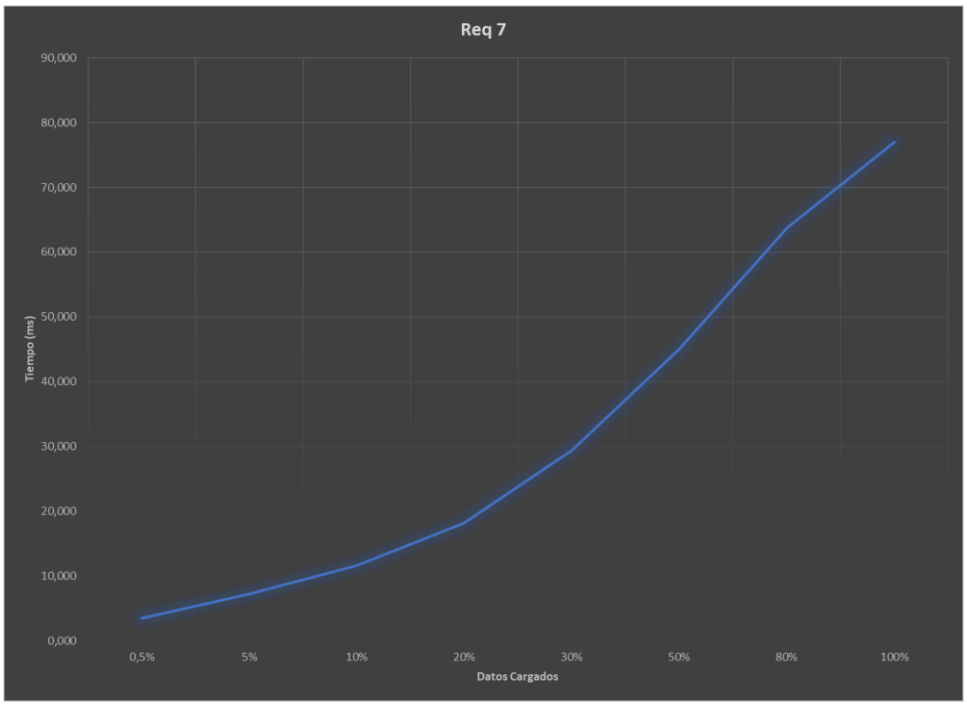
  Comparación mayor que = O (N+1)

  Orden de crecimiento = O (N+1)

  4. En el último paso se ordenan los datos de mayor a menor usando el algoritmo marge sort.

  Orden de crecimiento = O (Nlog(N))

- Tiempos de ejecución



Requerimiento 8 (Bono Grupal)

- Descripción

Entrada	catálogo, top pedido
Salida	Lista del top dado de actores con su información como: Colaboraciones, cantidad de producciones de cada tipo (Movies, TV Show) por plataforma
Implementado	Si

- Complejidad: O(N log N + M log M+ 2n+ N*Top*lista + colaboraciones*Top)

Pasos

1. Se crea una variable con la lista de todas las producciones y se ordena con criterio de la fecha en que se añadió en la plataforma

   Asignación = O(1)

   Ordenamiento Merge = O(NlogN)
2. Se crea un diccionario para guardar las cantidades según el actor, la información se agrega a una lista y se ordena para encontrar los actores con más participación

Asignación =O(1)

Ciclo = O(2 n) ya que hay 2 ciclos seguidos

Orden Merge = (M log M) ya que se organiza una lista de menor tamaño

3. Se crea un diccionario para añadir la información según el top dado

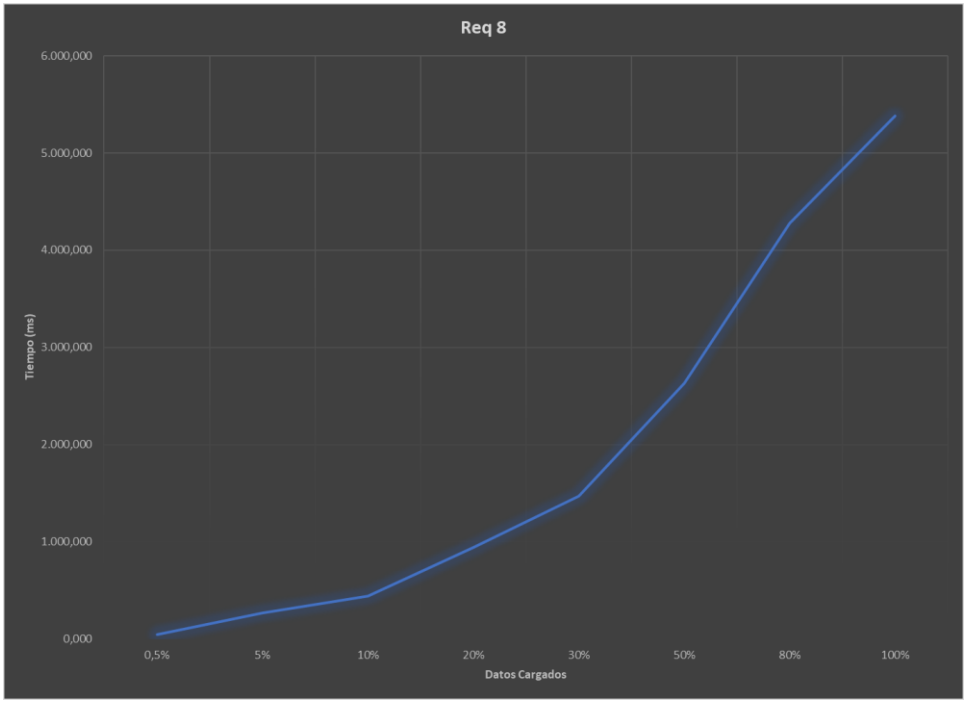
Primer Ciclo = O(Top) Veces que sea el top

Segundo Ciclo = O(N*Top) Revisa cuales tiene el mismo nombre que los del Top

Otros Ciclo = O(N*Top*lista) Contador de los géneros, colaboraciones, plataformas y tipos de contenido en los que se presenta cada actor

While = O(colaboraciones*Top) Verifica que colaboraciones están repetidas y las elimina

- Tiempos de ejecución



Conclusiones

- El modelo de trabajo MVC permite la fácil implementación y modificación de la aplicación en general, adicionalmente, permite encontrar y solucionar errores de manera más fácil y eficiente
- Las complejidades teóricas coinciden con los tiempos de ejecución, por ejemplo, la complejidad teórica más grande se da en el requerimiento 8 (BONO) y así lo demuestran sus resultados con respecto a los tiempos de ejecución.

- La estructura de datos más eficiente para la realización de los requerimientos fue ARRAY_LIST debido al resultado de sus tiempos de ejecución.
- La función de ordenamiento más eficiente en este Reto fue "Merge Sort", ya que, al ser una función estable, no requiere que se lleve a cabo un análisis de como vienen los datos de entrada. Adicionalmente, obtuvo los mejores tiempos de ejecución para cada requerimiento.

Anexos

Tabla de tiempos (medidos en ms)

Datos Cargados	Req 1	Req 2	Req 3	Req 4	Req 5	Req 6	Req 7	Req 8
0,5%	0,492	4,771	0,265	0,774	0,24	0,272	3,432	39,369
5%	2,32	36,339	0,98	3,53	0,864	0,474	7,231	266,449
10%	4,757	57,26	1,418	4,363	1,738	1,124	11,645	437,117
20%	10,167	118,842	3,348	8,712	2,62	1,526	18,138	941,77
30%	17,374	200,992	4,053	13,706	4,28	2,283	29,379	1470,115
50%	32,195	343,052	6,757	24,367	6,243	4,279	45,055	2632,446
80%	49,379	578,27	10,258	35,652	10,506	5,782	63,747	4278,878
100%	64,64	739,416	15,432	47,072	13,588	7,319	76,993	5382,068