OBSERVACIONES DEL RETO 2

Integrantes individuales:

Adriana Sofia Rozo Cepeda Cod. 202211498 (as.rozo@uniandes.edu.co)
Diego Fernando Galván Cruz Cod. 202213709 (d.galvanc@uniandes.edu.co)

Requerimiento grupal básico:

1. Requerimiento 2: Examinar los programas de televisión agregados en un año.

Requerimientos individuales:

2. Requerimiento 3: Adriana Sofia Rozo Cepeda
3. Requerimiento 5: Diego Fernando Galván Cruz

Requerimientos avanzados:

4. Requerimiento 6: Encontrar el contenido con un director involucrado.
5. Requerimiento 7: Listar el TOP (N) de los géneros con más contenido (G)

Analisis de complejidad → O() + Tiempo de ejecución:

Espacio de la carga [KB]

Porcentaje de la muestra [pct]	Carga
5.00%	1941.65
10.00%	18060.32
50.00%	83491.2
80.00%	168278.14
100.00%	189268.1

Ambientes de pruebas:

	Máquina 1	Máquina 2
Procesadores	11th Gen Intel(R) Core(TM) i7-1195G7 @ 2.90GHz 2.92 GHz	**11th Gen Intel(R) Core (TM) i7-1165G7 @ 2.80GHz 2.80 GHz**
Memoria RAM (GB)	16,0 GB (15,7 GB usable)	16,0 GB (15,7 GB usable)
Sistema Operativo	Sistema operativo de 64 bits, procesador basado en x64	**Windows 11 Home Single Language. 64-bit operating system, x64-based processo**

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

Las gráficas fueron realizadas según los resultados de la máquina 1 y para el req 4 (6 del reto) la gráfica fue realizada con los resultados de la máquina 2.

Máquina 1:

Porcentaje de la muestra [pct]	Req 2	Req 3	Req 5	Req 6	Req 7
5.00%	3.6	4.42	5.09	5.35	25.26
10.00%	6.78	3.49	17.27	10.6	26.6
50.00%	6.55	4,52	23.98	23.1	28.27
80.00%	7.19	4.54	24.44	23.05	28.14
100.00%	7.19	4.74	25.06	23.11	28.74

Memoria consumida vs porcentaje muestra Máquina 1

Porcentaje de la Muestra [pct]	Req 2	Req 3	Req 5	Req 6	Req 7
5.00%	0.24	0.27	0.56	101.37	19.18
10.00%	0.43	0.19	1.51	326.12	119.09
50.00%	1.85	0.24	8.41	5192.64	327.35
80.00%	2.88	0.17	14.02	11133.91	427.35
100.00%	3.08	0.26	20.65	16221.36	470.25

Tiempo de ejecución vs porcentaje muestra Máquina 1

Máquina 2:

Porcentaje de la muestra [pct]	Req 2	Req 3	Req 5	Req 6	Req 7
5.00%	4.16	6.09	9.49	11.1	27.18
10.00%	6.48	3.62	17.91	10.78	27.35
50.00	8.66	0.19	22.21	6.4	27.48
80.00%	7.44	4.66	24.68	7.24	27.64
100.00%	7.44	4.89	25.23	6.4	27.46

Memoria consumida vs porcentaje muestra Máquina 2

Porcentaje de la Muestra [pct]	Req 2	Req 3	Req 5	Req 6	Req 7
5.00%	0.45	0.21	1.66	166.57	94.38
10.00%	0.36	0.14	3.26	436.66	240.3
50.00%	2.52	0.19	9.56	5450.4	351.1
80.00%	2.96	0.18	80.96	11429	445.8
100.00%	3.35	0.2	84.19	16526	575.1

Tiempo de ejecución vs porcentaje muestra Máquina 2

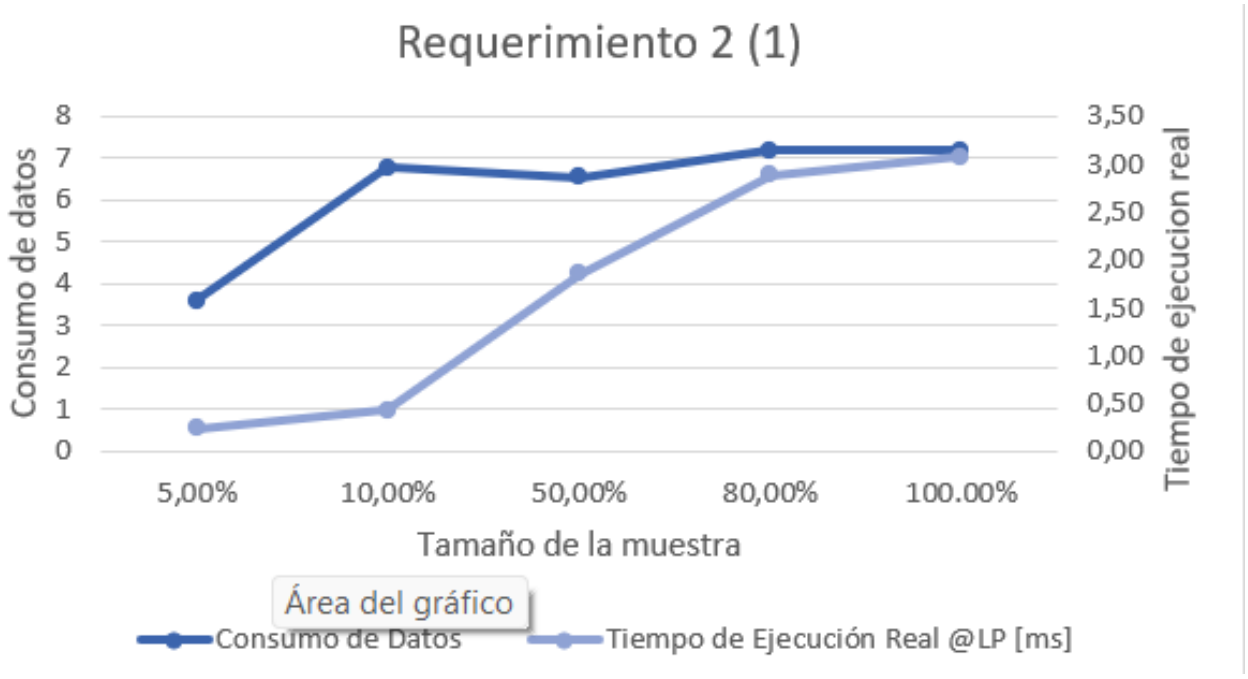
Requerimiento 1:

M = Tamaño Lista de lista_dates

O(Mlog(M))

En el requerimiento 1 consultamos en un mapa (‘films_per_date) donde cada llave es una fecha y su valor tiene un array con todos los los TV Shows que pertenecen a ese año. Laconsulta tiene una complejidad O(1) posteriormente se organizan los álbumesencontrados alfabéticamente, proceso que le corresponde una complejidad de lista_dates log(lista_dates). Por lo que la complejidad del requerimiento se resume en complejidad de organizar el contenido encontrado (m tamaño de los álbumes encontrados).

```
def TvShowsAdded(catalog, date):
    """
    Función principal del requerimiento 2
    """
    lista_dates = mp.get(catalog["model"]["films_per_date"], date)

    if lista_dates == None:
        return None, None
    lista_dates = mp.get(catalog["model"]["films_per_date"], date)["value"]
    num_TVShows = mp.size(lista_dates)
    lista_dates_ord = mgs.sort(lista_dates, cmpRequerimiento2)
    films = FirstAndLast(lista_dates_ord)


    return films, num_TVShows
```



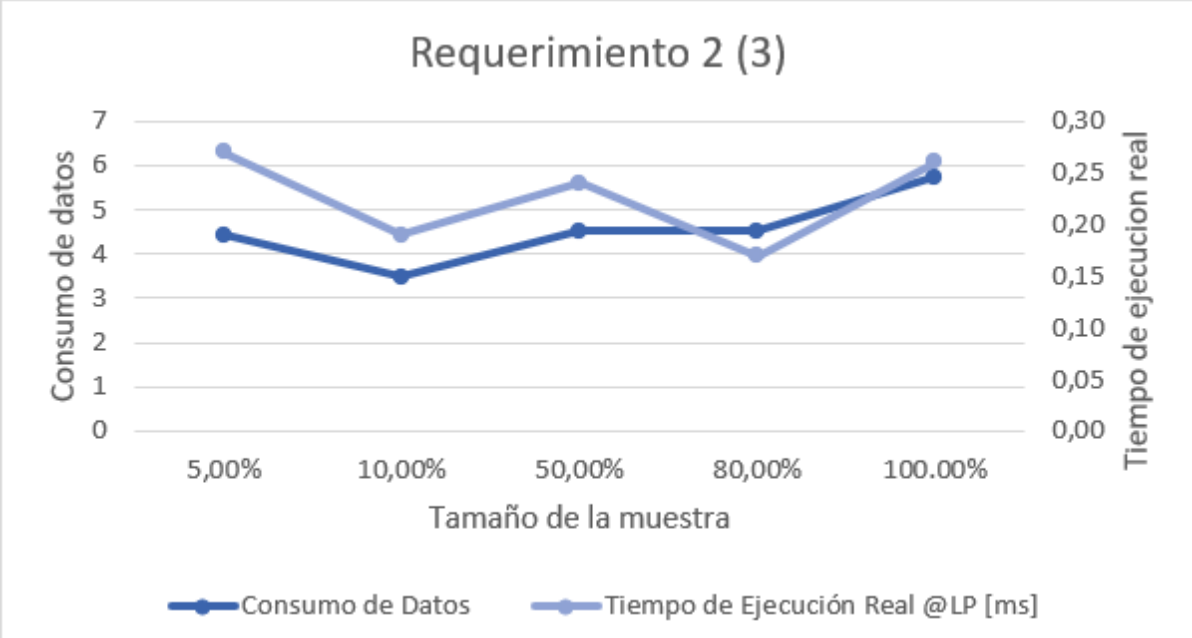
Requerimiento 2:

E = Tamaño Lista de exist_films

O(Elog(E))

En el requerimiento 5 consultamos en un mapa ('film_per_actor') donde cada llave corresponde al nombre de un actor y su valor tiene un array con todo el contenido en el que participa el actor. La consulta tiene una complejidad O(1) posteriormente seextrae diversa información del diccionario acciones con complejidades O(1). Después organizamos dicho contenido con el cmpRequerimiento3. Este proceso le corresponde una complejidad de exist_films log(exist_films). La complejidad del requerimiento se resume en complejidad de organizar el contenido encontrado.

```
def ContentByActor(catalog, Actor_Name):
    resp_films = None
    exist_films = mp.get(catalog["film_per_actor"], Actor_Name)
    if exist_films:
        resp_films = me.getValue(exist_films)
        types = CountContentbyTypeR3(resp_films)
        resp_films = mgs.sort(resp_films,cmpRequerimiento3)

    return resp_films, types
```



Requerimiento 3:

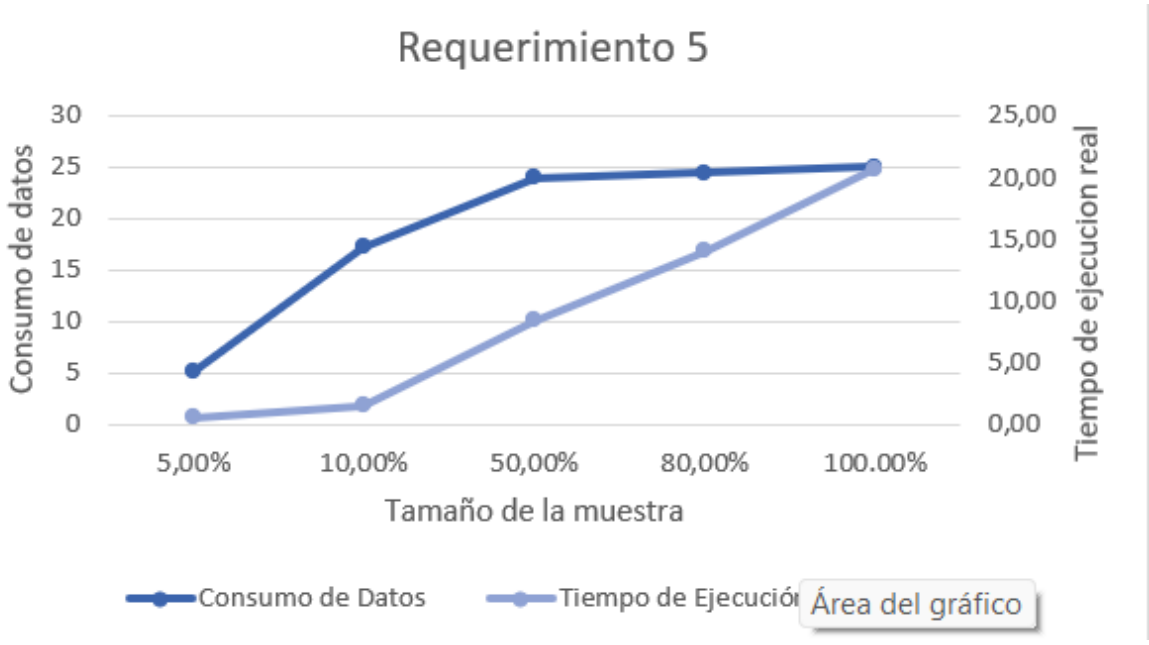
C = Tamaño Lista de exist_films

O(Clog(C))

```
def ContentCountry(catalog, country):
    resp_films = None
    exist_films = mp.get(catalog["film_per_country"], country)
    if exist_films:
        resp_films = me.getValue(exist_fi
        types = CountContentbyTypeR5(resp
        resp_films = mgs.sort(resp_films,cmpRequerimiento5)

    return resp_films, types
```

(function) cmpRequerimiento5: (film1: Any, film2: Any) -> bool

En el requerimiento 3 consultamos en un mapa ('film_per_country') donde cada llave corresponde al nombre de un país y su valor tiene un ar. La consulta tiene una complejidad O(1) posteriormente se extrae diversa información del diccionario (acciones con complejidades O(1)). Después organizamos dicho contenido con el cmpRequerimiento5. Este proceso le corresponde una complejidad de exist_films log(exist_films). La complejidad del requerimiento se resume en complejidad de organizar el contenido encontrado.



Requerimiento 6: búsqueda por director

C = Tamaño Lista de Contenido por dic

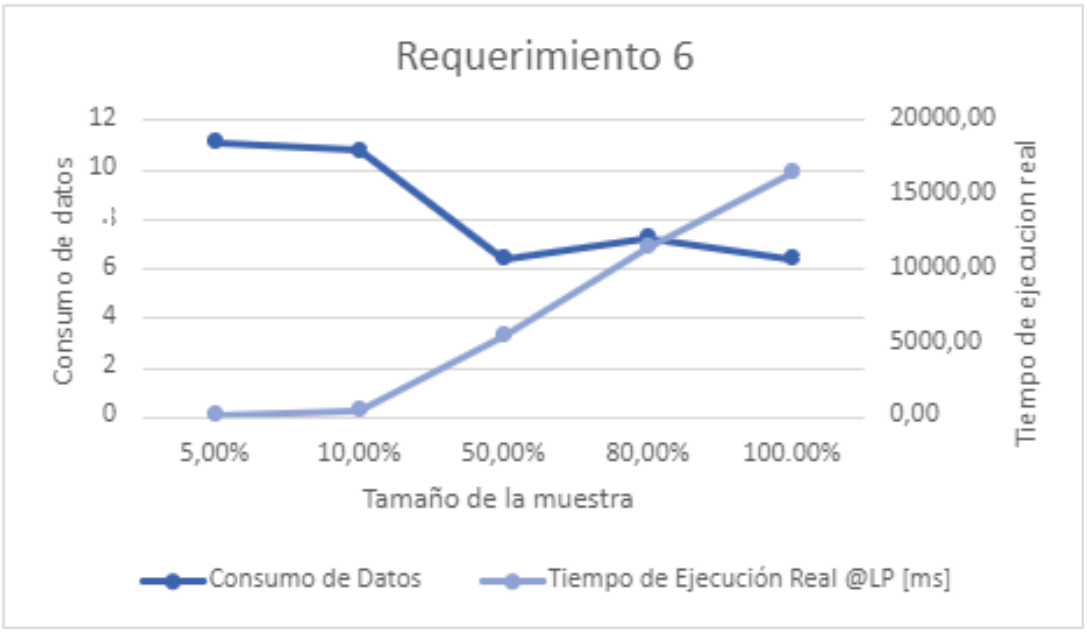
(Contenido por director)

O(N^2)

En el requerimiento 6 consultamos en un mapa (`FilmsbyDirector) donde cada llave corresponde al nombre de un director y su valor es una lista con la información de las correspondientes peliculas. La consulta tiene una complejidad O(1) posteriormente. se consulta en 4 mapas internos donde cada llave corresponde al código de un país y su valor es una lista con todas las canciones en ese país. Esta consulta también tiene una complejidad O(1). Después organizamos las canciones por su popularidad y en su defecto la duración. Este proceso le corresponde una complejidad de cancionesEncontradas log(cancionesEncontradas). Por lo que la complejidad del requerimiento se resume en complejidad de organizar las canciones encontradas.

```
def ContentByDirector(catalog, director):
    contenidopordic = mp.get(catalog["films_per_director"], director)
    contenidopordic = me.getValue(contenidopordic)
    if mp.isEmpty(contenidopordic) == True:
        print("No hay contenido asociado a ese director")

    cantidad_total_DIC = lt.size(contenidopordic) #Se obtiene el contenido TOTAL por director

    contenidopordic =   mgs.sort(contenidopordic,cmpRequerimiento6) #Se organiza la función

    types= CountContentbyTypeR6(contenidopordic,director) #Se obtiene el contenido TOTAL por TV shows y movies

    numero_cont_genero={} #Diccionario para saber la cantidad de de contenido por genero}
    cont_total= 0 #Total contenido
    for contenido in lt.iterator(contenidopordic):
        genre = contenido["listed_in"]

        if ',' in genre:
            genero_cont = genre.split(",")
            for i in genero_cont:
                if i in numero_cont_genero:
                    numero_cont_genero[i]+=1
                else:
                    numero_cont_genero[i]=1
            cont_total += 1
        else:
            if genre in numero_cont_genero:
                    numero_cont_genero[genre]+=1
                    cont_total += 1
            else:
                numero_cont_genero[genre]=1
                cont_total += 1
    keys=mp.keySet(catalog['films_per_director'])
    large=mp.size(catalog['films_per_director'])
```

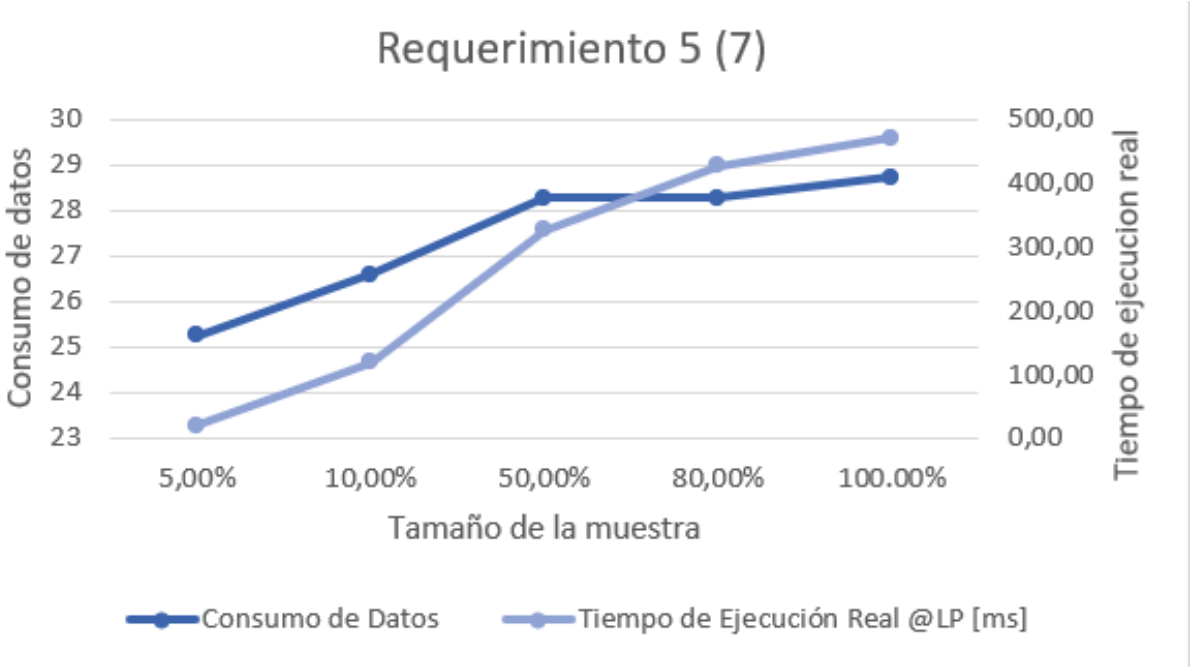
```
for contenido in range(1,large):
    id= lt.getElement(keys,contenido)
    numero_cont_platform = {'amazon':0,
        'netflix':0,
        'hulu':0,
        'disney':0} #Diccionsrio para contenido por plataforma
    cont = mp.get(catalog['films_per_director'],id)['value']['first']
    while cont:
        if mp.contains(catalog['id_amazon'],cont['info']['show_id']):
            numero_cont_platform['amazon'] +=1
        elif mp.contains(catalog['id_netflix'],cont['info']['show_id']):
            numero_cont_platform['netflix'] +=1
        elif mp.contains(catalog['id_hulu'],cont['info']['show_id']):
            numero_cont_platform['hulu'] +=1
        elif mp.contains(catalog['id_disney'],cont['info']['show_id']):
            numero_cont_platform['disney'] +=1
        cont = cont['next']
#Convertir diccionario a lista de tuplas
list_genre = [(k,v) for k,v in numero_cont_genero.items()]

respuesta_general = (cantidad_total_DIC, types, list_genre, cont_total, contenidopordic, numero_cont_platform)

return respuesta_general
```



Requerimiento 7:

O(N^2)

En el requerimiento 7 consultamos en un mapa ('films_per_genres') donde cada llave corresponde al nombre de un género y su valor tiene un array con todo el contenido que pertenecen a ese género. Las consultas tienen una complejidad O(1) (línea 431 y 432) y posteriormente se extrae diversa información del diccionario acciones con complejidades O(1). Se recorre hasta la longitud de la lista y dentro del ciclo consultamos el genre con la función lt.getElement(). Posterior a ello creamos un diccionario en el que sus llaves corresponden a lo requerido en la instrucción del req 7. A su vez en este ciclo obtenemos la primera película de la consulta hecha (línea 446).
Dentro del ciclo de While contamos el tipo de contenido (movie o TvShow), así como también mediante métodos de consulta obtenemos la cantidad de contenido al que corresponde cada plataforma dado un top.
Finalmente se organiza la función con mgs.sort() usando la cmp del requerimiento 7 (en nuestro caso, el 5).



```python
def topGenres (catalog,top):
    genresTop= lt.newList('SINGLE_LINKED')
    keys=mp.keySet(catalog['film_per_genres'])
    size=mp.size(catalog['film_per_genres'])


    for gen in range(1,size):
        genre=lt.getElement(keys,gen)
        g={
            'genre':genre,
            'media':mp.get(catalog['film_per_genres'],genre)['value'],
            'num_movies':0,
            'num_shows':0,
            'amazon':0,
            'netflix':0,
            'hulu':0,
            'disney':0
        }

        movie = mp.get(catalog['film_per_genres'],genre)['value']['first']
        while movie:
            if movie['info']['type']=='Movie':
                g['num_movies']+=1
            else:
                g['num_shows']+=1

            if mp.contains(catalog['id_amazon'],movie['info']['show_id']):
                g['amazon'] +=1
            elif mp.contains(catalog['id_netflix'],movie['info']['show_id']):
                g['netflix'] +=1
            elif mp.contains(catalog['id_hulu'],movie['info']['show_id']):
                g['hulu'] +=1
            elif mp.contains(catalog['id_disney'],movie['info']['show_id']):
                g['disney'] +=1
            movie = movie['next']


        lt.addLast(genresTop,g)

    genresTop= mgs.sort(genresTop,cmpRequerimiento7)
    rank=lt.newList('SINGLE_LINKED')
    size = lt.size(genresTop)
    for i in range(1,top + 1):
        if(i == size): break
        add=lt.getElement(genresTop,i)
        lt.addLast(rank,add)

    return rank
```

Análisis con Reto 1:

Requerimiento	Tiempo (Large)		Complejidad	
	Reto 1	**Reto 2**	**Reto 1**	**Reto 2**
Req 1 (Rec 2 Reto)	44.12	3.08	O(nlog(n))	M = Tamaño Lista de list_dates O(Mlog(M))
Req 2 (Rec 3 Reto)	44.82	0.26	***O(nlog(n))*** **+** O(n^2)	K = Tamaño Lista de exist_films O(Klog(K))
Req 3 (Rec 5 Reto)	40.59	20.65	***O(nlog(n))*** **+** O(n^2)+ O(n ^3)	C = Tamaño Lista de exist_films O(Clog(C))
Req 4 (Rec 6 Reto)	19.43	11133.91	***O(nlog(n))+*** O(n^2)+ O(n ^3)	O(N^2)
Req 5 (Rec 7 Reto)	12968. 19	470.25	***O(nlog(n))+*** O(n^2)+ O(n ^3)	O(N^2)

NOTA: En el caso del reto 2 los Sorts no implican un aumento considerable, puesto que los datos que se encuentran en las listas son considerablemente pequeños, por lo que no afectan la velocidad de las consultas. Por otra parte, el uso de tracemalloc para revisar el uso de la memoria implica un aumento relevante en la ejecución de los requerimientos, por lo que los datos de tiempos recogidos tuvieron que ser realizados sin la toma de memoria, por tanto, son fieles a la realidad.

COMPARACIÓN CON EL RETO 1

En el requerimiento 1 del Reto 1 se buscaba en un rango de años la consulta era binaria para la búsqueda de la posición de la posición 1 y lineal, por diversos problemas en la implementación para la búsqueda de la posición del final del rango después se hacía una sublista de esos datos.

Por otra parte, en el reto 2 en este requerimiento, solo hay la consulta de un año con complejidad O(1) y el Sort de los álbumes que allí pertenecen, siendo en este caso un requerimiento con menor complejidad. En el caso del requerimiento 2 del Reto 1 la complejidad era O(n) o en su defecto la cantidad de artistas que deseábamos listar, esto gracias a que en la carga los datos ya se encontraban organizados y solo toca extraerlos. Mientras que en el Reto 2 hacemos una consulta O(1) y un Sort de los Artistas encontrados, encontrando en esta casi allí mayor complejidad.

En el requerimiento 3 del Reto 1 la complejidad es similar al requerimiento anterior, puesto que la complejidad se basa en la cantidad de datos a listar. De igual manera, para el caso del reto 2 la complejidad se encuentra en el Sort de los datos encontrados dentro de la consulta O(1) en el mapa.

En el requerimiento 4 del Reto 1 la complejidad está en la búsqueda con complejidad log(n) del artista y posteriormente la búsqueda lineal con complejidad O(cancionesDelArtista) de las canciones de este artista para encontrar las que pertenezcan al país de distribución deseado. En el caso de requerimiento 2, tan la búsqueda de las canciones y del artista es O(1) gracias a la complejidad de las tablas de hash, pero se hace un Sort de este pequeño grupo de datos, siendo este el que le aumenta la complejidad.

En el requerimiento 5 del Reto 1 la complejidad está en la búsqueda con complejidad log(n) del artista para después extraer sus datos internos de los álbumes, lo que corresponde a O(1). Para el caso del reto 2 los álbumes se encuentran a una complejidad de O(1) y posteriormente se recorren linealmente para extraer la información de interés, siendo esta la complejidad se define este requerimiento. En el requerimiento 6 del Reto 1 la complejidad es similar a la del requerimiento 1 donde se tiene que buscar un rango con complejidad logarítmica para una posición y lineal para la otra. En el caso del reto 2 para este requerimiento el comportamiento es similar a la del requerimiento 4 con dos consultas O(1) y finalmente un Sort de Los datos encontrados