ANÁLISIS DEL RETO

Juan Jose Cediel Borrero, 202212212, jj.cediel@uniandes.eedu.co.

Juan David Rojas Cañizales, 202120434, jd.rojasc12@uniandes.edu.co.

Juan Estaban Alvarez Garcia, 202212030, je.alvarezg1@uniandes.edu.co.

Nota: Para todas las pruebas realizadas se utilizó la misma máquina, siendo esta la que contaba con mejor hardware de los miembros del equipo.

Procesadores	**Intel(R) Core (TM) I5-10300H CPU @ 2.5GHz 2.50 GHz**
Memoria RAM (GB)	8 GB
Sistema Operativo	Windows 10 Home x64

Las únicas librerías usadas fueron datetime, prettytable, csv, time y la librería del curso.

Requerimiento 1

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Año inicial del periodo (con formato AAAA). Año final del periodo (con formato AAAA).
Salidas	El número total de películas presentes en el periodo y la información de las tres primeras y tres últimas películas de dicha lista.
Implementado (Sí/No)	Sí. Juan Esteban Álvarez

Análisis de complejidad

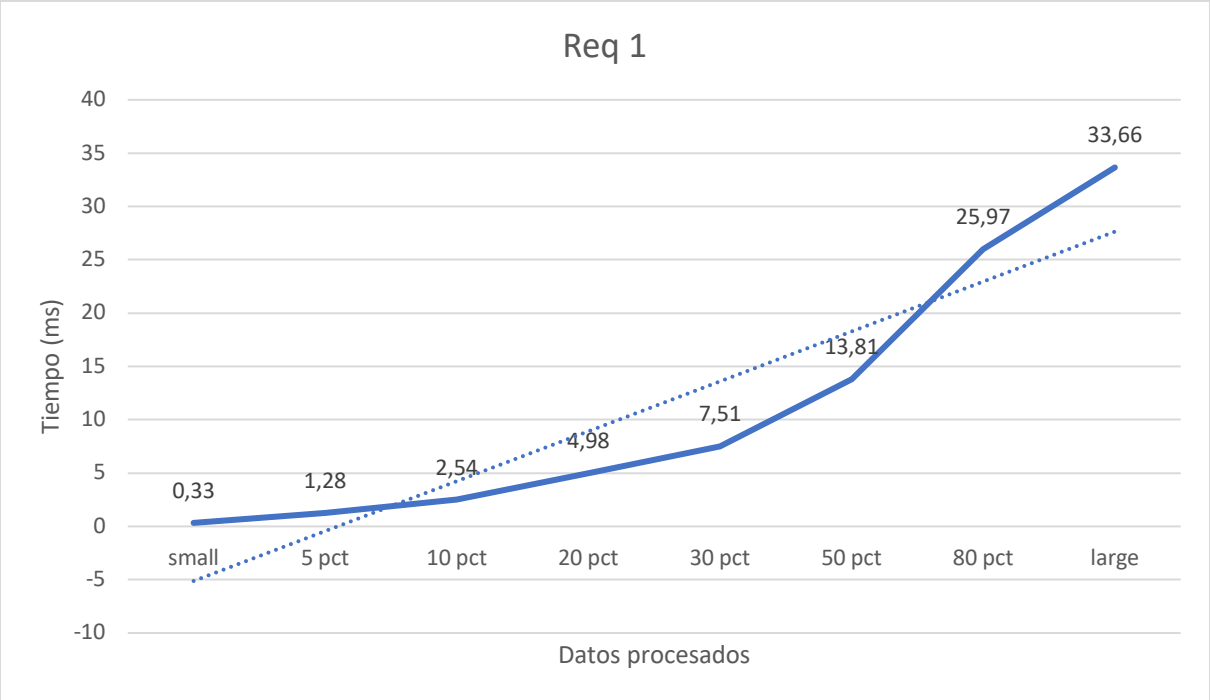
Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1) Validar los inputs del usuario	O(1)
Paso 2) Inicializar TAD que retornará la información al usuario	O(1)
Paso 3) Recorrer linealmente todos los videos del catálogo y añadir al TAD los videos que cumplen la condición.	O(N)
TOTAL	***O(N)***

Pruebas Realizadas (Prueba con “0000-2020”)

Muestra	Tiempo (s)
small	0,33
5 pct	1,28
10 pct	2,54
20 pct	4,98
30 pct	7,51
50 pct	13,81
80 pct	25,97
large	33,66

Graficas



Req	Input	Muestra	Output	Tiempo (ms)
1	0-2020	small	133 peliculas	0,33
1	0-2020	5 pct	716 peliculas	1,28
1	0-2020	10 pct	1439 peliculas	2,54
1	0-2020	20 pct	2938 peliculas	4,98
1	0-2020	30 pct	4450 peliculas	7,51
1	0-2020	50 pct	7458 peliculas	13,81
1	0-2020	80 pct	11898 peliculas	25,97
1	0-2020	large	14884 peliculas	33,66

Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Teóricamente la implementación tiene un orden lineal O(N), y experimentalmente la gráfica se mantiene cerca de la línea de tendencia sin dispersarse mucho, lo que coincide con la gráfica lineal esperada teóricamente.

Requerimiento 2

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Fecha inicial del periodo (con formato “%B %d, %Y"). Fecha final del periodo (con formato “%B %d, %Y").
Salidas	El número total de programas de TV presentes en el periodo y la información de los tres primeros y tres últimos programas de dicha lista.
Implementado (Sí/No)	Sí. Juan Esteban Álvarez

Análisis de complejidad

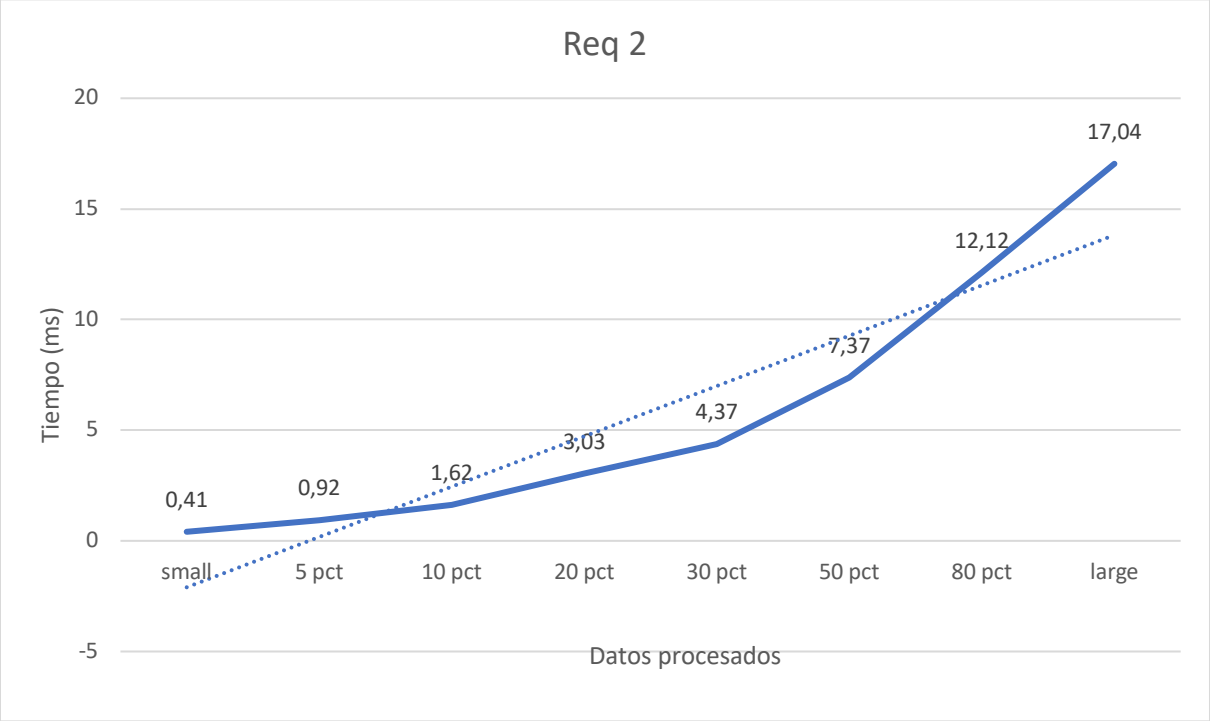
Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1) Validar los inputs del usuario	O(1)
Paso 2) Inicializar TAD que retornará la información al usuario	O(1)
Paso 3) Recorrer linealmente todos los videos del catálogo y añadir al TAD los videos que cumplen la condición.	O(N)
TOTAL	***O(N)***

Pruebas Realizadas (Prueba con “September 17, 2010 - September 17, 2020”)

Muestra	**Tiempo (s)**
small	0,41
5 pct	0,92
10 pct	1,62
20 pct	3,03
30 pct	4,37
50 pct	7,37
80 pct	12,12
large	17,04

Graficas



Req	Input	Muestra	Output	Tiempo (ms)
2	September 17, 2010 - September 17, 2020	small	41 TV_shows	0,41
2	September 17, 2010 - September 17, 2020	5 pct	177 TV_shows	0,92
2	September 17, 2010 - September 17, 2020	10 pct	337 TV_shows	1,62
2	September 17, 2010 - September 17, 2020	20 pct	639 TV_shows	3,03
2	September 17, 2010 - September 17, 2020	30 pct	944 TV_shows	4,37
2	September 17, 2010 - September 17, 2020	50 pct	1568 TV_shows	7,37
2	September 17, 2010 - September 17, 2020	80 pct	2510 TV_shows	12,12
2	September 17, 2010 - September 17, 2020	large	3169 TV_shows	17,04

Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Teóricamente la implementación tiene un orden lineal O(N), y experimentalmente la gráfica se mantiene cerca de la línea de tendencia sin dispersarse mucho, lo que coincide con la gráfica lineal esperada teóricamente.

Requerimiento 3

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Nombre del actor a buscar
Salidas	Cantidad de películas y cantidad de programas en donde participo el actor, además de los datos plataforma, titulo, directos, actores, duración, país y géneros de las primeras tres y ultimas tres.
Implementado (Sí/No)	Sí. Juan David Rojas Cañizales

Análisis de complejidad

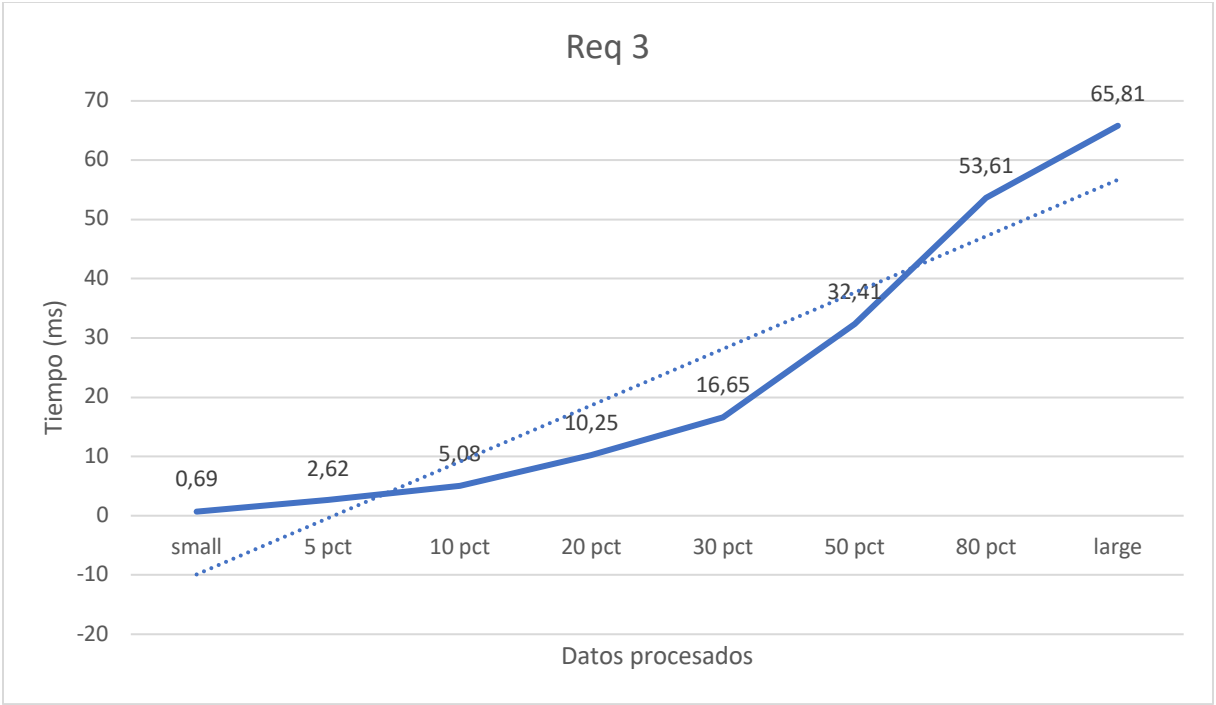
Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Llamado view -> controller	O(1)
Llamada controller -> model	O(1)
Iteracion sobre actores de cada video	O(n*m)
If el actor buscados esta en los actores del video	O(1)
If type es Movie o Tv_Show	O(1)
Agregar el actual video a la ultima pos de los videos del actor	O(1)
TOTAL	***O(n*m)***

Pruebas Realizadas (Prueba con “Chris Evans”)

Muestra	**Tiempo (s)**
small	0,69
5 pct	2,62
10 pct	5,08
20 pct	10,25
30 pct	16,65
50 pct	32,41
80 pct	53,61
large	65,81

Graficas



Req	Input	Muestra	Output	Tiempo (ms)
3	Chris Evans	small	1 Movie / 0 TV_shows	0,69
3	Chris Evans	5 pct	1 Movie / 0 TV_shows	2,62
3	Chris Evans	10 pct	2 Movie / 0 TV_shows	5,08
3	Chris Evans	20 pct	4 Movie / 0 TV_shows	10,25
3	Chris Evans	30 pct	7 Movie / 0 TV_shows	16,65
3	Chris Evans	50 pct	12 Movie / 0 TV_shows	32,41
3	Chris Evans	80 pct	17 Movie / 0 TV_shows	53,61
3	Chris Evans	large	20 Movie / 0 TV_shows	65,81

Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Se evidencia que la ejecución para encontrar los videos en que los participo un actor tiene un crecimiento O(n*m) donde n es la cantidad de videos y m es la cantidad de actores que participan en cada video, sabiendo que n es mucho más grande que m. La gráfica muestra un crecimiento teórico lineal y el crecimiento real da muestra de ello.

Requerimiento 4

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Géneros
Salidas	El número total de películas de ese género. El número total de programas de ese género. Los tres primeros y tres últimos registros de la lista filtrada por la categoría ingresada por parámetro.
Implementado (Sí/No)	Sí. Juan Jose Cediel

Análisis de complejidad

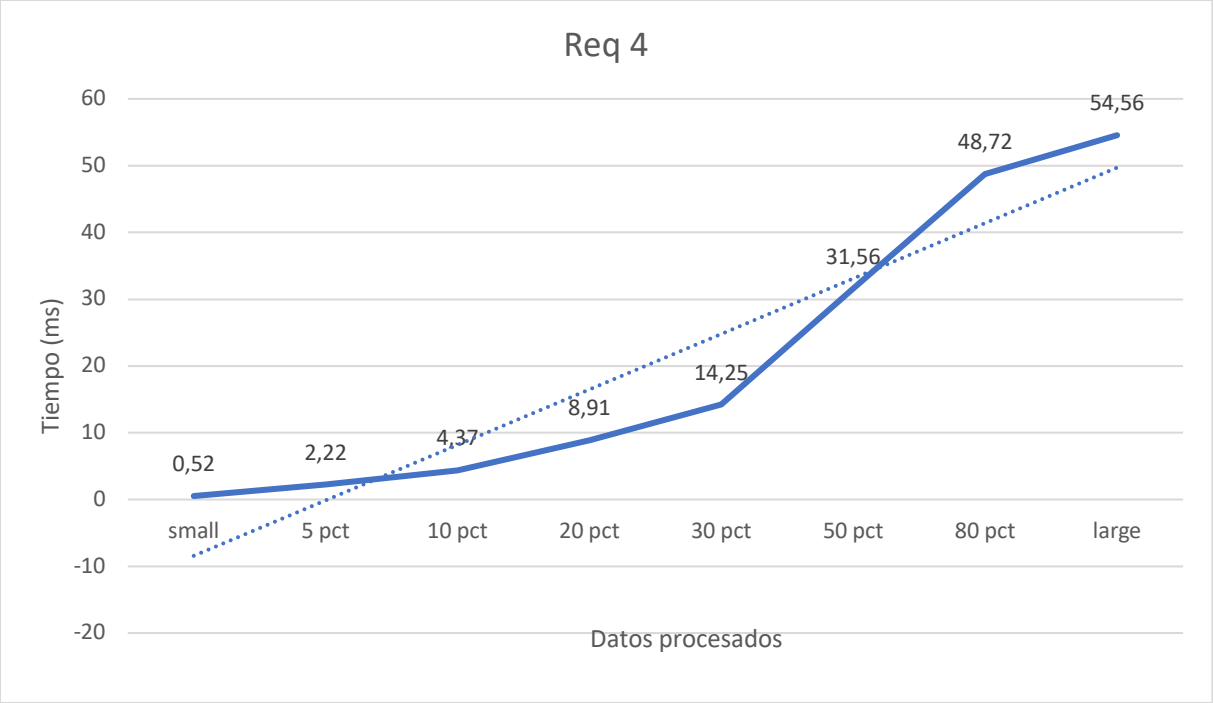
Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Validar los inputs del usuario	O(1)
Paso 2 : Inicializar TAD que retornará la información al usuario	O(1)
Paso 3: Recorrer linealmente todos los videos del catálogo y añadir al TAD los videos que cumplen la condición.	O(N)
Paso 4: Extraer por cada video del catalogo su categoría y verificar si es igual a la ingresada por parámetro.	O(C)
TOTAL	$\boldsymbol{O\,(N * C)}$

Pruebas Realizadas (Prueba con Drama)

Entrada	**Tiempo (s)**
small	0,52
5 pct	2,22
10 pct	4,37
20 pct	8,91
30 pct	14,25
50 pct	31,56
80 pct	48,72
large	54,56

Graficas



Req	Input	Datos procesados	Tiempo (ms)
4	Drama	small	0,52
4	Drama	5 pct	2,22
4	Drama	10 pct	4,37
4	Drama	20 pct	8,91
4	Drama	30 pct	14,25
4	Drama	50 pct	31,56
4	Drama	80 pct	48,72
4	Drama Spielberg	large	54,56

Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

Teóricamente la implementación tiene un orden lineal O(N), y experimentalmente la gráfica se mantiene cerca de la línea de tendencia sin dispersarse mucho, lo que coincide con la gráfica lineal esperada teóricamente.

Requerimiento 5

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Fecha inicial del periodo (con formato “%B %d, %Y"). Fecha final del periodo (con formato “%B %d, %Y").
Salidas	El número total de programas de TV presentes en el periodo y la información de los tres primeros y tres últimos programas de dicha lista.
Implementado (Sí/No)	Sí. Juan Esteban Álvarez

Análisis de complejidad

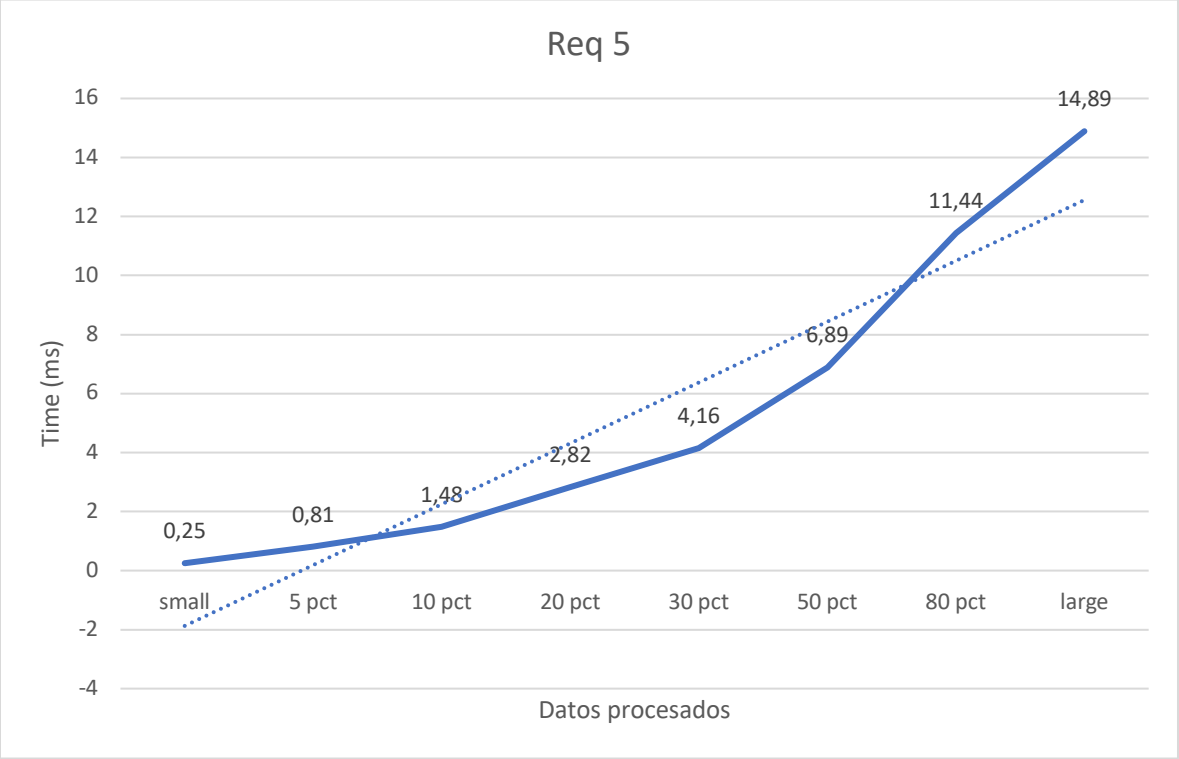
Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1) Validar los inputs del usuario	O(1)
Paso 2) Inicializar TAD que retornará la información al usuario	O(1)
Paso 3) Recorrer linealmente todos los videos del catálogo y añadir al TAD los videos que cumplen la condición.	O(N)
TOTAL	***O(N)***

Pruebas Realizadas (Prueba con “United States”)

Muestra	**Tiempo (s)**
small	0,25
5 pct	0,81
10 pct	1,48
20 pct	2,82
30 pct	4,16
50 pct	6,89
80 pct	11,44
large	14,89

Graficas



Req	Input	Muestra	Output	Tiempo (ms)
5	United States	small	30 Movie / 18 TV_shows	0,25
5	United States	5 pct	168 Movie / 74 TV_shows	0,81
5	United States	10 pct	337 Movie / 180 TV_shows	1,48
5	United States	20 pct	686 Movie / 334 TV_shows	2,82
5	United States	30 pct	1027 Movie / 502 TV_shows	4,16
5	United States	50 pct	1730 Movie / 822 TV_shows	6,89
5	United States	80 pct	2742 Movie / 1265 TV_shows	11,44
5	United States	large	3390 Movie / 1607 TV_shows	14,89

Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Teóricamente la implementación tiene un orden lineal O(N), y experimentalmente la gráfica se mantiene cerca de la línea de tendencia sin dispersarse mucho, lo que coincide con la gráfica lineal esperada teóricamente.

Requerimiento 6

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Director
Salidas	En tablas: El número total de películas y programas dirigidos por ese director. El número total de películas y programas por cada género (listed_in). El número total de películas y programas por plataforma.
Implementado (Sí/No)	Sí. Juan Jose Cediel

Análisis de complejidad

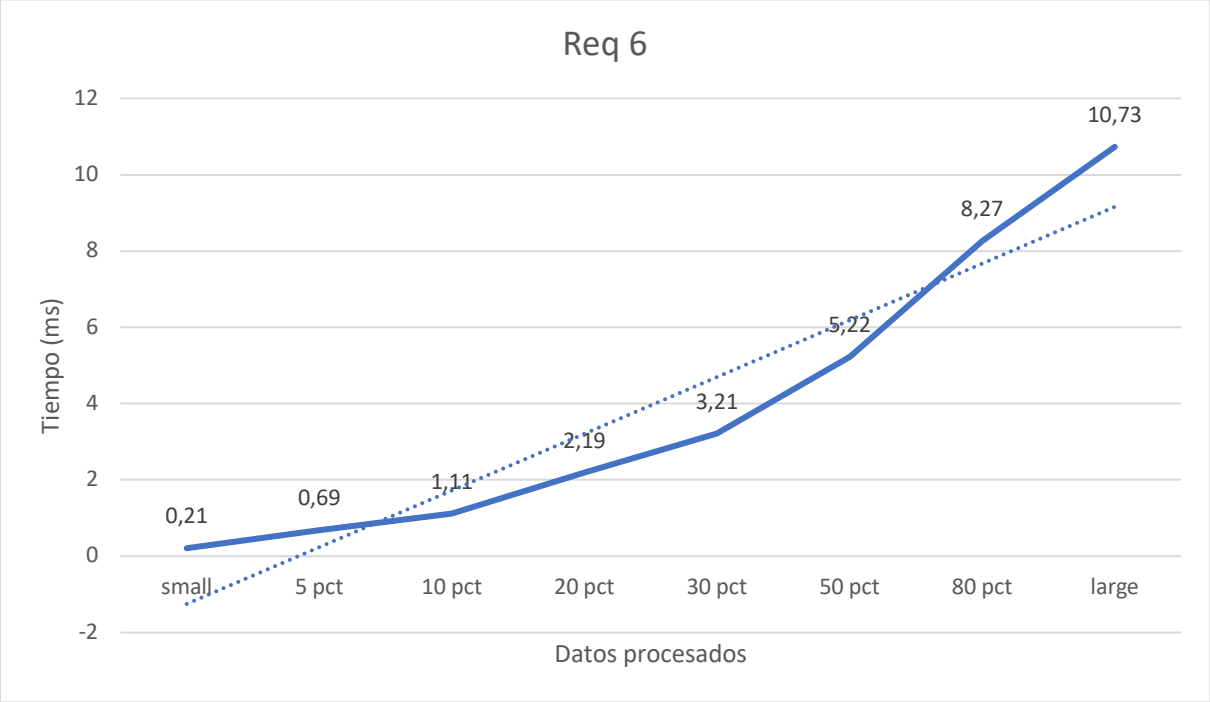
Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Paso 1: Validar los inputs del usuario	O(1)
Paso 2: Inicializar TAD que retornará la información al usuario	O(1)
Paso 3: Recorrer linealmente todos los videos del catálogo y añadir al TAD los videos que cumplen la condición.	O(N)
Paso 3.1: Recorrer linealmente las categorías del video y añadirlas a aux géneros.	O(C)
Paso 4: Agregar como una lista las categorías en dataDirector	O(L)
TOTAL	***O(N*C)+ O(L)***

Pruebas Realizadas (Prueba Steven Spielberg)

Entrada	**Tiempo (ms)**
small	0,21
5 pct	0,69
10 pct	1,11
20 pct	2,19
30 pct	3,21
50 pct	5,22
80 pct	8,27
large	10,73

Graficas



Req	Input	Datos procesados	Tiempo (ms)
6	Steven Spielberg	small	0,21
6	Steven Spielberg	5 pct	0,69
6	Steven Spielberg	10 pct	1,11
6	Steven Spielberg	20 pct	2,19
6	Steven Spielberg	30 pct	3,21
6	Steven Spielberg	50 pct	5,22
6	Steven Spielberg	80 pct	8,27
6	Steven Spielberg	large	10,73

Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

Teóricamente la implementación tiene un orden lineal O(N), y experimentalmente la gráfica se mantiene cerca de la línea de tendencia sin dispersarse mucho, lo que coincide con la gráfica lineal esperada teóricamente.

Requerimiento 7

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Entero que representa el top de géneros a listar
Salidas	Rank, géneros, total de videos, cantidad de movies, cantidad de tv shows, cantidad de videos en Amazon Prime, cantidad de videos en Disney Plus, cantidad de videos en Hulu y cantidad de videos en Netflix.
Implementado (Sí/No)	Sí. Juan David Rojas Cañizales

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Carga Inicial

Pasos	**Complejidad Absoluta**	**Complejidad Acumulada**
Iteracion sobre los videos	O(n)	
Iteracion sobre las categorias de cada video	O(m)	O(n*m)
Iteracion con isPresent para saber si la categoria ya esta en la lista categorias del catalog	O(z)	O(n*m*z)
Si la categoria no se encuentra se inserta en la ultima posicion	O(1)	
Operación GetElement sobre la lista de categorias del catalog	O(1)	
If type es Movie o Tv_Show	O(1)	
If la Plataforma del video es Amazon_Prime, Disney_Plus, Hulu y Netflix	O(1) -> O(4)	
Merge sort sobre la lista categorías del catalog para organizarla de mayor numero de videos por categoría a menor numero de videos por categoría.	O(n*log(n))	
TOTAL	**O(n*m*z) + O(n*log(n))**	

Ejecución por Usuario

Pasos	**Complejidad**
Llamado view -> controller	O(1)
Llamada controller -> model	O(1)
Sublist de la lista con categorias organizadas de categoria con mayor numero de videos al menor numero de videos	O(n)
TOTAL	***O(n)***

Pruebas Realizadas de Carga Inicial

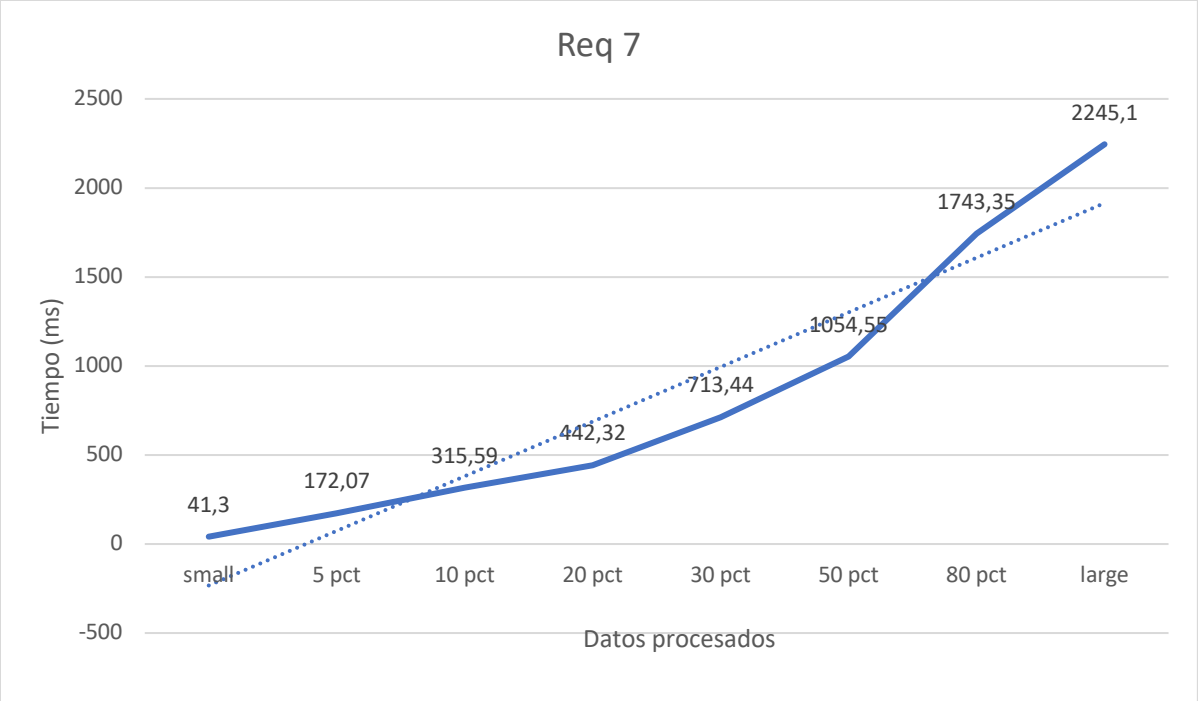
Muestra	Tiempo (ms)
small	41,3
5 pct	172,07
10 pct	315,59
20 pct	442,32
30 pct	713,44
50 pct	1054,55
80 pct	1743,35
large	2245,1

Pruebas Realizadas (Prueba con “100”)

Muestra	Tiempo (s)
small	0,05
5 pct	0,06
10 pct	0,05
20 pct	0,06
30 pct	0,06
50 pct	0,06
80 pct	0,07
large	0,07

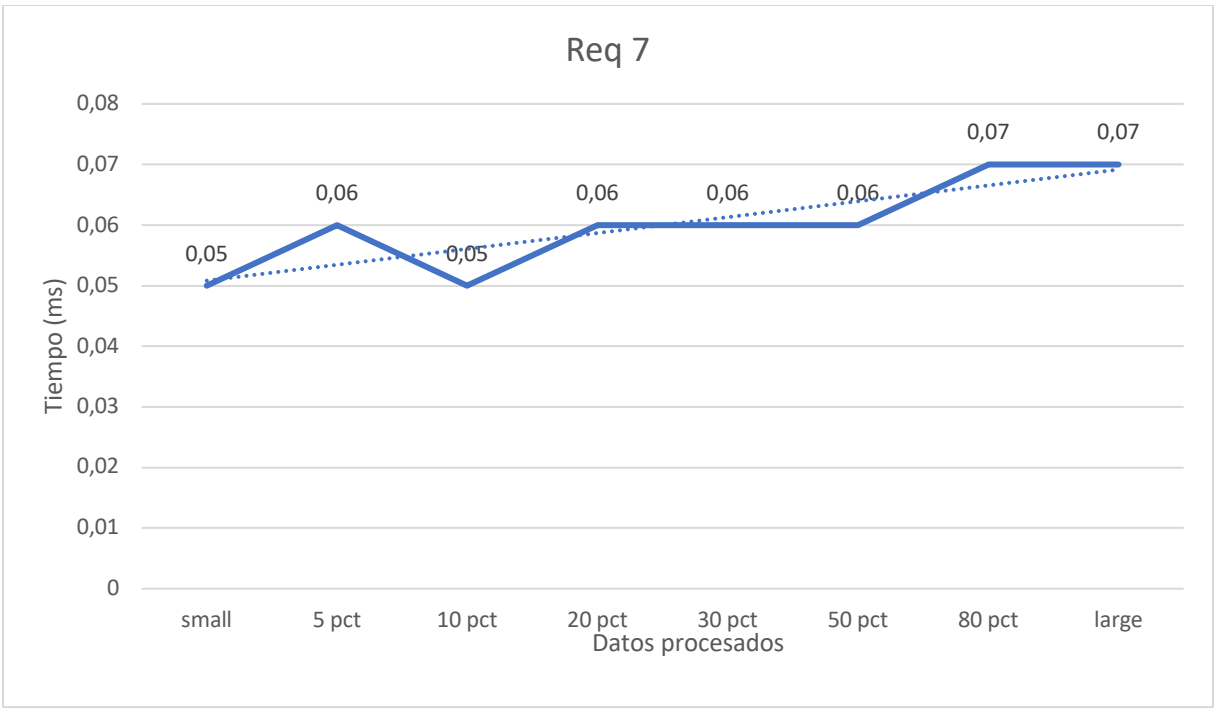
Graficas

Grafica Carga Inicial



Req	Muestra	Tiempo (ms)
7	small	41,3
7	5 pct	172,07
7	10 pct	315,59
7	20 pct	442,32
7	30 pct	713,44
7	50 pct	1054,55
7	80 pct	1743,35
7	large	2245,1

Grafica Ejecucion Por Usuario



Req	Input	Muestra	Output	Tiempo (ms)
7	100	small	Top 100 / Categorias	0,05
7	100	5 pct	Top 100 / Categorias	0,06
7	100	10 pct	Top 100 / Categorias	0,05
7	100	20 pct	Top 100 / Categorias	0,06
7	100	30 pct	Top 100 / Categorias	0,06
7	100	50 pct	Top 100 / Categorias	0,06
7	100	80 pct	Top 100 / Categorias	0,07
7	100	large	Top 100 / Categorias	0,07

Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Se evidencia que el orden de crecimiento para el obtener el ranking de categorías organizado desde la categoría con mayor cantidad de videos a la con menor cantidad de videos, tiene un crecimiento de O(n*m*z) donde n es la cantidad de videos, m es la cantidad de categorías que tiene cada video y z es la cantidad de categorías ya inicializadas en la lista categorías del catalog. Sumado al orden de crecimiento O(n*log(n)) que se da al realizar un merge sort sobre la lista categorías del catalog que nos permite ordenar de mayor a menor las categorías. En la implementación se prefiero este camino dado que podemos garantizar que m < z < n por lo que no se llega hasta un orden de crecimiento cuadrático y se mantienen los tiempos de ejecución bajos. Para la implementación de ejecución por usuario se ejecuta la función sublist, que en el peor de los casos es O(n). Para ambos casos se demuestra un comportamiento real muy conforme al comportamiento teórico.

Requerimiento 8

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Entero que representa el top de actores a listar
Salidas	Rank, nombre, cantidad de videos en Amazon Prime, cantidad de videos en Disney Plus, cantidad de videos en Hulu, cantidad de videos en Netflix, cantidad de movies, cantidad de tv shows, nombre de los directores con los que ha trabajado y nombre de los actores con los que ha trabajado.
Implementado (Sí/No)	Sí. Juan David Rojas Cañizales

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Carga Inicial

Pasos	**Complejidad Absoluta**	**Complejidad Acumulada**
Iteracion sobre los videos	O(n)	
Iteracion sobre los actores de cada video	O(m)	O(n*m)
If el actor esta o no registrado en un dict	O(1)	
If type es Movie o Tv_Show	O(1)	
If la Plataforma del video es Amazon_Prime, Disney_Plus, Hulu y Netflix	O(1) -> O(4)	
Agregar el actual video a la ultima pos de los videos del actor	O(1)	
Iteracion sobre los actores registrados en el dict	O(x) X cant de actores	
Iteracion sobre cada video por cada actor	O(y) Y cant videos en los que actuo el actor	O(x*y)
isPresent sobre cada director de cada video en la lista de directores que tiene cada actor	O(z) Z cant de directores que trabajaron con ese actor	O(x*y*z)
If existe o no el director en la lista de directores del actor	O(1)	
Si el director ya existía solo se busca entonces se hace un getElement sobre ARRAY_LIST	O(1)	
isPresent sobre cada miembro del cast de cada video en la lista de compañeros que tiene cada actor	O(d) d cant de actores que trabajaron con el actor	O(x*y*z*d)
If existe o no el actor en la lista de compañeros del actor	O(1)	

Si el compañero ya existía solo se busca entonces se hace un getElement sobre ARRAY_LIST	O(1)	
Merge sort sobre la lista de directores del actor	O(z*logz)	O(x*y*z*d*(z*logz))
Merge sort sobre la lista de compañeros del actor	O(d*logd)	O(x*y*z*d*(z*logz) * (d*logd))
Iteracion por cada actor del dict para insertarlo en un lista de tipo ARRAY_LIST	O(x) X cant de actores	O(x*y*z*d*(z*logz) * (d*logd)*x)
Merge sort sobre la lista actores del catalog	O(x*logx) X cant de actores	O(x*y*z*d*(z*logz) * (d*logd)*x*(x*logx)
TOTAL	**O(x*y*z*d*(z*logz) * (d*logd)*x*(x*logx)) + O(n*m)**	

Ejecución por Usuario

Pasos	**Complejidad**
Llamado view -> controller	O(1)
Llamada controller -> model	O(1)
Sublist de la lista con actores organizados por mayor numero de actuaciones al menor numero de actuaciones	O(n)

Pruebas Realizadas de Carga Inicial

Entrada	**Tiempo (s)**
small	265,97
5 pct	1778,34
10 pct	4330,54
20 pct	7820,18
30 pct	12981,43
50 pct	24628,67
80 pct	47403,95
large	66051,14

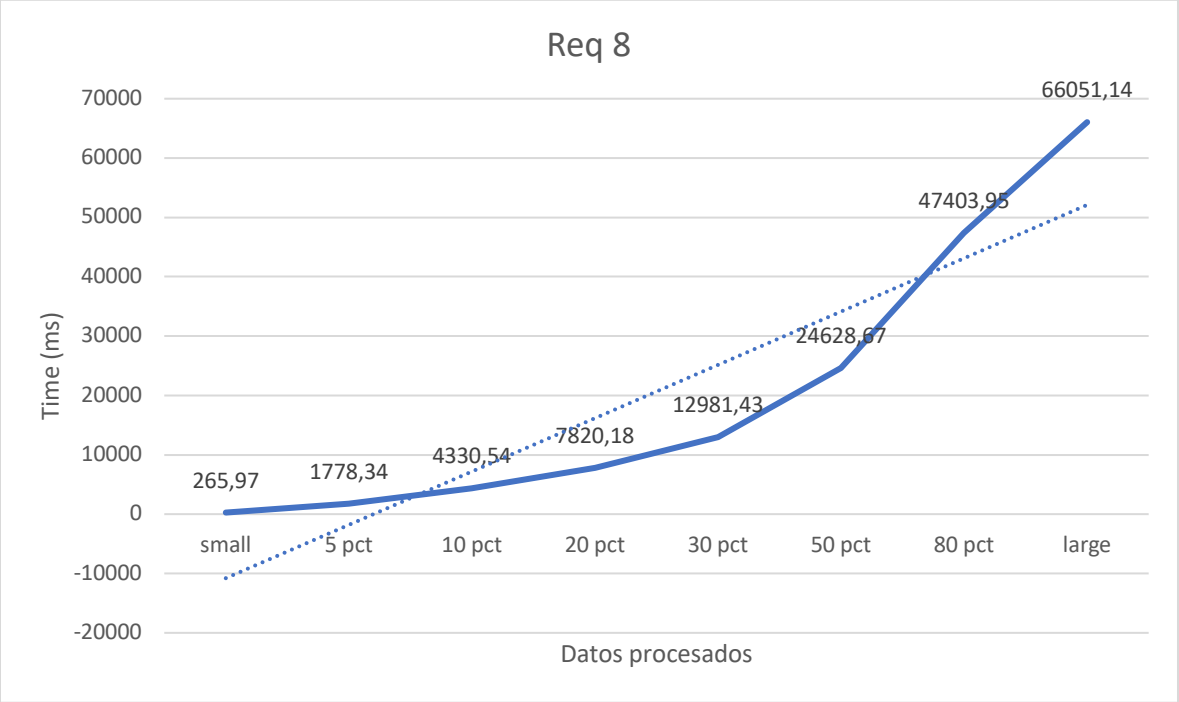
Pruebas Realizadas (Prueba con “100”)

Entrada	**Tiempo (s)**
small	0,05
5 pct	0,06
10 pct	0,05
20 pct	0,08
30 pct	0,06
50 pct	0,08
80 pct	0,06

large	0,06

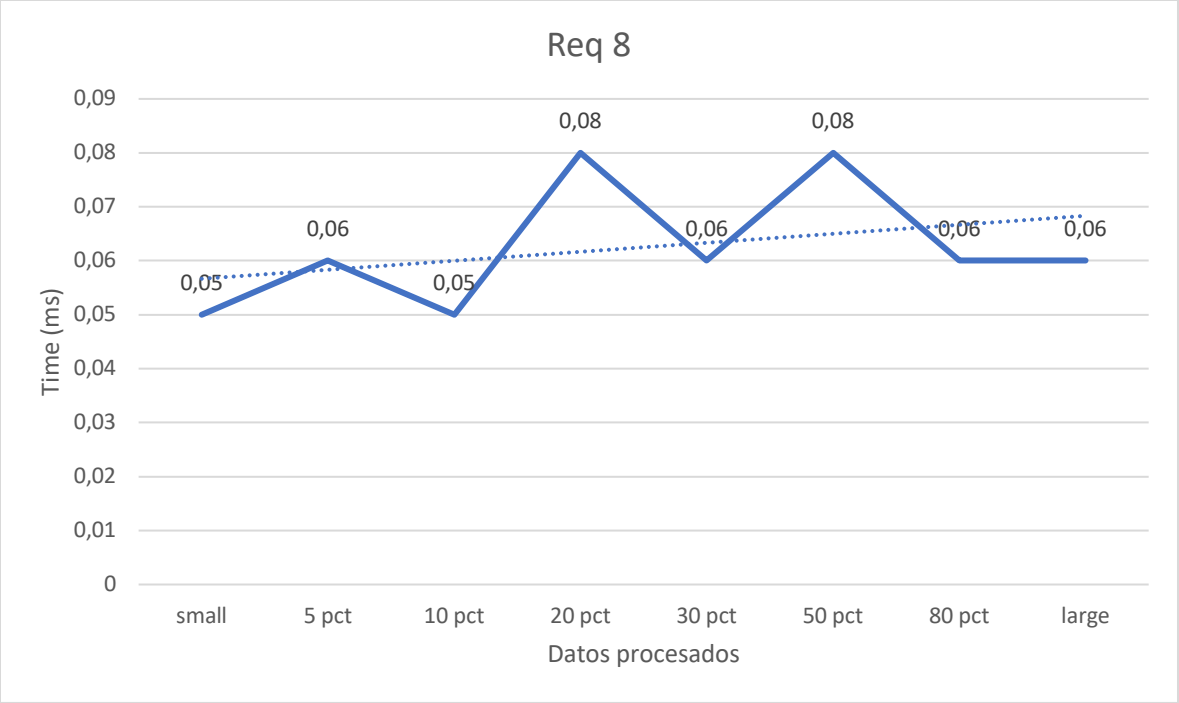
Graficas

Grafica Carga Inicial



Req	Muestra	Tiempo (ms)
7	small	265,97
7	5 pct	1778,34
7	10 pct	4330,54
7	20 pct	7820,18
7	30 pct	12981,43
7	50 pct	24628,67
7	80 pct	47403,95
7	large	66051,14

Grafica Ejecución Por Usuario



Req	Input	Muestra	Output	Tiempo (ms)
8	100	small	Top 100 / Actores	0,05
8	100	5 pct	Top 100 / Actores	0,06
8	100	10 pct	Top 100 / Actores	0,05
8	100	20 pct	Top 100 / Actores	0,08
8	100	30 pct	Top 100 / Actores	0,06
8	100	50 pct	Top 100 / Actores	0,08
8	100	80 pct	Top 100 / Actores	0,06
8	100	large	Top 100 / Actores	0,06

Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Se evidencia que el orden de crecimiento para el obtener el ranking de actores organizado desde el actor con mayor cantidad de videos al con menor cantidad de videos, tiene un crecimiento de O(n*m) donde n es la cantidad de videos, m es la cantidad de actores que tiene cada video. Sumado al orden de crecimiento O(x*y*z*d*(z*log(z))*(d*log(d))*x*(x*log(x)) donde x es la cantidad de actores que fueron obtenidos en la primer iteración (O(n*m)) y es la cantidad de videos en los que participo cada actor, z es la cantidad de directores con los que trabajo cada actor, d es la cantidad de actores que trabajaron con ese actor, z*log(z) es el orden de crecimiento del merge sort aplicado a la lista de directores del actor, d*log(d) es el orden de crecimiento del merge sort aplicado a la lista de compañeros actores con los que ha trabajado el actor y por ultimo x*logx es el merge sort aplicado a la lista de actores para poder ordenarlos de mayor a menor. Esta implementación, si bien parece complicada busca de forma

eficiente generar ciclor for lineales, es decir sabemos que antemano que x,y,z,d son variables que distan mucho de valor, donde x es la de mayor valor por mucha diferencia, esto nos permite reducir al mínimo las expresiones con exponente que elevan los tiempos de ejecución. Para la ejecución por usuario se ejecuta la función sublist, que en el peor de los casos es O(n) donde n es la cantidad de actores registrados. Para ambos casos se demuestra un comportamiento real muy conforme al comportamiento teórico.