Análisis reto 1 de Estructura de datos y algoritmos

Integrantes:

- Estudiante 1: Jerónimo López / 202320969 / j.lopezm234@uniandes.edu.co
- Estudiante 2: Julián David Ramos González /202414411/ jd.ramosg1@uniandes.edu.co
- Estudiante 3: Juan Esteban Piñeros Barrera / 202412232 / Je.pineros@uniandes.edu.co

Requerimientos:

- Requerimiento 1: Grupal
- Requerimiento 2: Grupal
- Requerimiento 3: Julián Ramos (Estudiante 2)
- Requerimiento 4: Jerónimo López (Estudiante 1)
- Requerimiento 5: Juan Esteban Piñeros (Estudiante 3)
- Requerimiento 6: Grupal
- Requerimiento 7: Grupal
- Requerimiento 8: Grupal

Requerimiento 1:

Complejidad temporal: Lo primero que cabe resaltar es que hay dos bucles en la función, pero estos no son anidados. El primero es “for i in range(ar.size(catalog['runtime'])):” el cual itera sobre todos los elementos de catálogo en su runtime. Su tamaño es n, donde n es la cantidad de elementos en el catálogo. Debido a que no hay operaciones extras aparte de asignaciones y comparaciones (O(1)) que se harán en todo el tamaño del catálogo, lo cual implica que este primer bucle tiene una complejidad de O(n). El segundo bucle “for pelicula in movies_true” tiene la idea de comparar todas las películas que cumplen la condición y ver cuál es la que se estrenó primero, en el peor de los casos debe recorrer toda la lista, por lo que la complejidad es de O(N). Ambos ciclos tienen una complejidad de O(n), pero estos no son anidados, por ende la función se puede ver como un O(n) + O(n) = O(n). Resultado final, la función del requerimiento tiene una complejidad de O(n)

Tabla de tiempo:

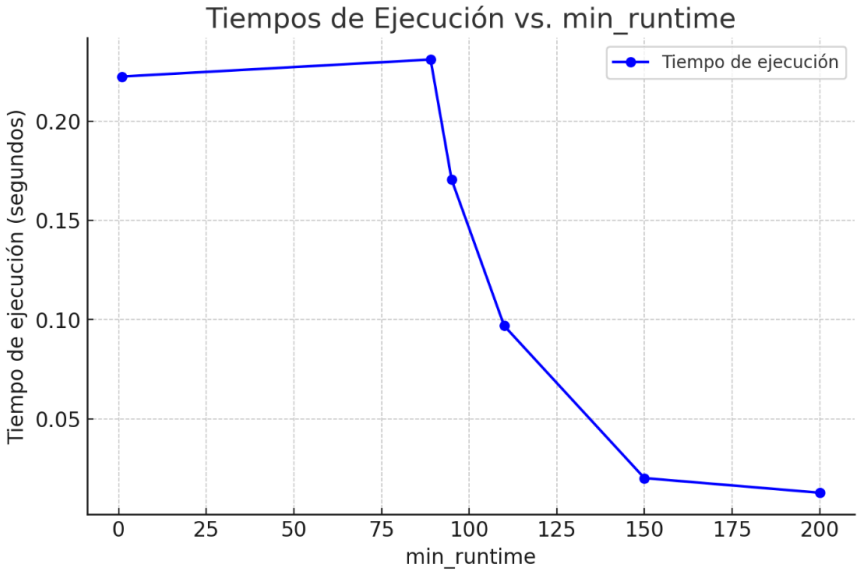


Tabla de comparación:

	min_runtime	Tiempo de ejecución (
1	1	0.222425
2	89	0.230985
3	95	0.170466
4	110	0.096832
5	150	0.020125
6	200	0.012752

Análisis final:

Podemos ver que la función si es lineal, lo que pasa y a lo que se debe su extraño comportamiento es que a medida que vamos ampliando el min_runtime, las películas que cumplen esta característica son menores, por lo que la segunda lista cada vez tiene menor volumen de datos y hace que el segundo bucle se ejecute menos veces (ya que disminuye el tamaño de n) por lo cual se acorta el tiempo

Esta función fue probada con movies-small.csv

Requerimiento 2:

Primer bucle (for i in range (ar.size(catalog['original_language']))**):**

Este bucle recorre todos los elementos en el catálogo que están en original_language. Si n es el número de elementos en el catálogo, este bucle tiene una complejidad de O(n), ya que realiza iteraciones sobre todos los elementos. Dentro del bucle, las operaciones son básicamente asignaciones, comparaciones y adiciones a una lista (movies_true). Cada una de estas operaciones se ejecuta en tiempo constante, es decir, O(1).

Segundo bucle (for pelicula in movies_true**):**

Una vez se ha filtrado la lista de películas que cumplen la condición de idioma, este bucle recorre las películas filtradas para identificar cuál es la última (en términos de fecha de publicación). En el peor de los casos, todas las películas cumplen la condición, lo que implica que la lista movies_true tendrá también n elementos, y, por lo tanto, el segundo bucle también será O(n). Sin embargo, a medida que se restringe el número de películas que cumplen con el idioma, el tamaño de movies_true se reduce, lo que significa que este bucle se ejecuta menos veces.

Como los bucles no están anidados, se suman sus complejidades, resultando en O(n) + O(n), lo cual nos da una complejidad final de O(n), es decir, la función tiene una complejidad lineal.

Tabla de tiempo:

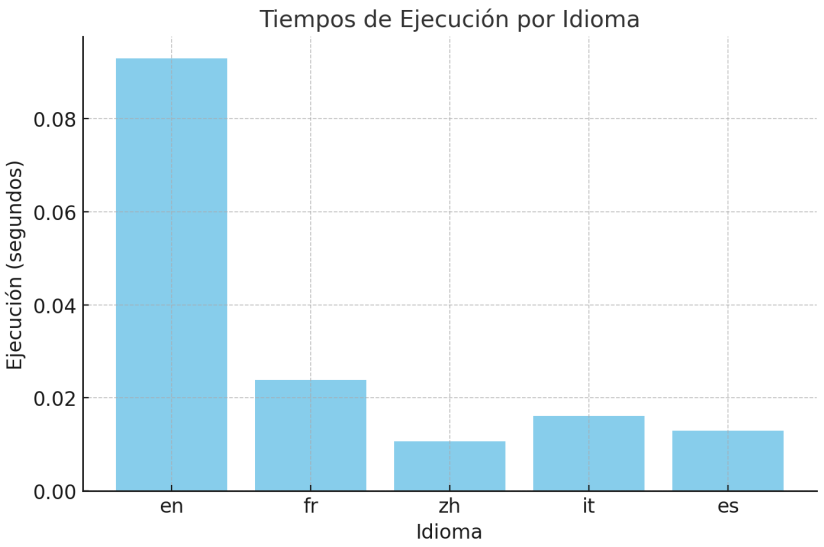


Tabla de comparación:

Idioma	Ejecución (segundos)
en	0.092973
fr	0.023898
zh	0.010663
it	0.016178
es	0.012964

Análisis final:

El comportamiento observado de que el tiempo de ejecución se acorta cuando se reduce la lista movies_true es esperado, ya que el segundo bucle solo necesita recorrer un subconjunto del total de elementos en el catálogo. Esto se manifiesta especialmente cuando el parámetro idioma_usuario es poco común en el catálogo, reduciendo el tamaño de la lista a comparar. En el gráfico se puede observar con claridad que el inglés, al ser el idioma más común, tiene un tiempo de ejecución superior, ya que requiere una mayor cantidad de datos en el bucle. Pero si hablamos, por ejemplo, de idiomas como español o italiano, el segundo bucle se ejecuta muchas menos veces, haciendo que el tiempo total de ejecución sea menor que cuando muchas películas cumplen con la condición de idioma.

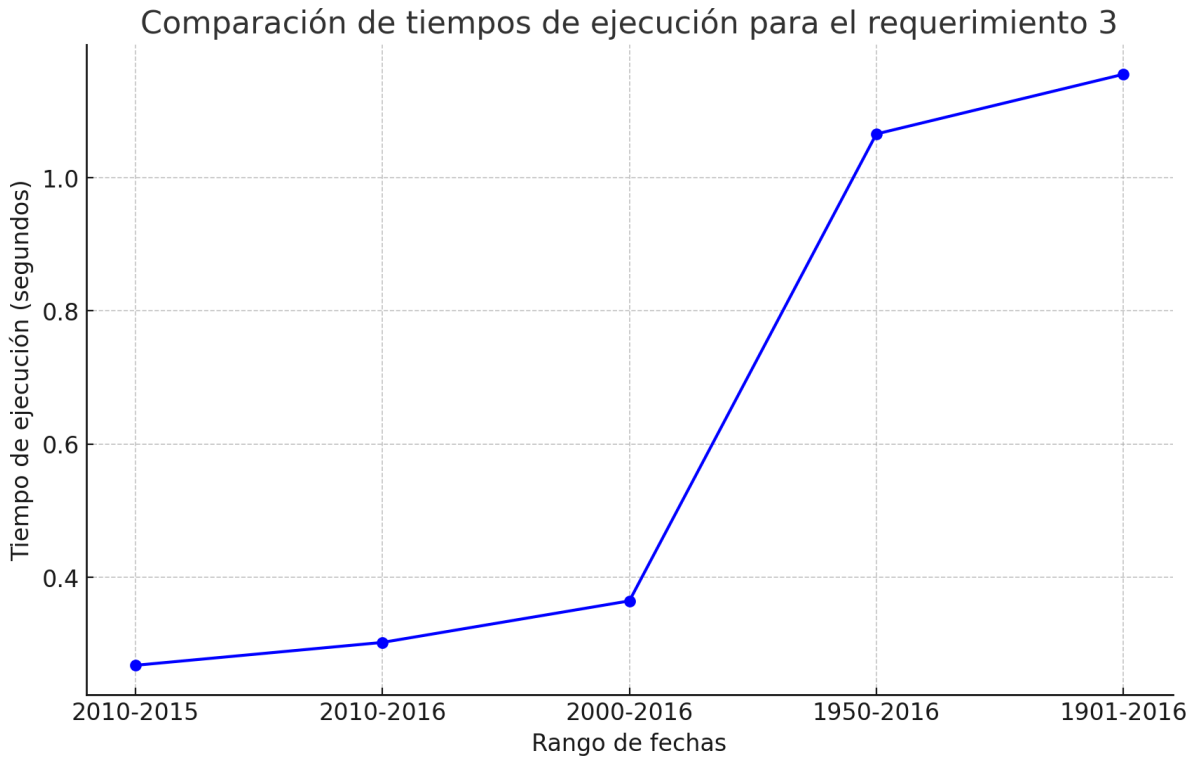
Esta función fue probada con movies-small.csv

Requerimiento 3:

Análisis de complejidad: Se debe recorrer toda la lista de películas para verificar el idioma y las fechas, lo que implica una complejidad de O(N) en el peor de los casos, donde N es el número total de películas. Cada película es comparada con el idioma y el rango de fechas especificado, lo cual toma tiempo constante para cada comparación, lo que no afecta la complejidad, manteniéndose en O(N).

Por lo tanto, la complejidad total del requerimiento es O(N), ya que se recorre la lista una sola vez para aplicar los filtros.

Tabla de tiempo:



Realizado con el idioma fr en la base de datos small

Tabla de comparación:

Rango de fechas	Número de películas	Tiempo de ejecución (segundos)
2010-10-10 a 2015-10-10	140	0.267881
2010-10-10 a 2016-10-10	167	0.302219
2000-10-10 a 2016-10-10	368	0.364614
1950-10-10 a 2016-10-10	701	1.065716
1901-10-10 a 2016-10-10	753	1.155280

Análisis final:

El análisis de los resultados muestra que los tiempos de ejecución del requerimiento 3 crecen de manera proporcional al número de películas filtradas, lo que concuerda con la complejidad teórica de O(N). A medida que aumenta el rango de fechas y el número de películas, el tiempo de procesamiento también incrementa de forma lineal. Las pequeñas variaciones en los tiempos se deben a factores externos del sistema, pero en general, la implementación es eficiente y no presenta problemas de rendimiento significativos.

Esta función fue probada con movies-small.csv

Requerimiento 4:

Inicialización de fechas y tamaño (fecha_inicial_dt**,** fecha_final_dt**, y** tamanio**):**

Convertir las fechas a objetos datetime toma tiempo constante O(1) y calcular el tamaño del catálogo (tamanio) también se obtiene en tiempo constante, O(1).

Primer bucle (for i in range(tamanio)**):**
Este bucle recorre todos los elementos del catálogo. Si n es el número de películas en el catálogo, entonces este bucle tiene una complejidad O(n). Dentro del bucle, se ejecutan varias operaciones como la conversión de la fecha de lanzamiento de la película a un objeto datetime (O(1)) y comparaciones de fechas (O(1)). También se verifica si el estado de la película coincide con el estado especificado (O(1)).
Cuando una película cumple con las condiciones, se suma su duración y se crea un diccionario con la información de la película. Esta creación del diccionario y la asignación de los valores individuales es una operación constante O(1). Cada vez que una película cumple con la condición, se añade a la lista peliculas["peli"], lo cual tiene una complejidad O(1) para cada inserción.
Ordenamiento con ordenar_peliculas_por_fecha_insertion**:**
La función ordena la lista de películas filtradas usando insertion sort. La complejidad de este algoritmo de ordenamiento es $O(n^2)$, donde k es el número de películas que cumplen las condiciones del estado y las fechas. En el peor caso, todas las películas en el catálogo cumplen las condiciones, por lo que el costo del ordenamiento sería $O(n^2)$.
Cálculo del promedio de duración:
El cálculo del promedio de duración se hace después del bucle principal, y su complejidad es constante O(1), ya que solo implica una división.
Selección de las primeras y últimas películas:
Si hay más de 20 películas en la lista filtrada, se seleccionan las primeras 5 y las últimas 5 películas. Seleccionar estos elementos tiene una complejidad O(1), ya que no se está recorriendo la lista completa.

El bucle principal es O(n) y el ordenamiento tiene una complejidad de $O(n^2)$. En el peor caso, cuando todas las películas cumplen las condiciones, el costo total sería $O(n) + O(n^2)$, lo que resulta en una complejidad final de $O(n^2)$ en el peor de los casos.

Tabla de tiempo:

Tabla de comparación:

Fecha inicial	Fecha final	Estado	Ejecución (segundos)
1960-01-01	1970-01-01	Released	0.607532
1970-01-01	1980-01-01	Released	1.039397
1980-01-01	1990-01-01	Released	1.327906
1990-01-01	2000-01-01	Released	2.672090
2000-01-01	2010-01-01	Released	10.252207
2010-01-01	2018-01-01	Released	13.094810

Análisis final:

Analizando los resultados obtenidos podemos darnos cuenta de que las películas son fechas más antiguas, por lo cual el tiempo de ejecución se reduce drásticamente. Por el contrario, fechas más recientes cuentan con un mayor número de peliculas registradas en la base de datos, por lo cual el tiempo de ejecución aumenta en $O(n^2)$. Aquel principal componente que le añade tiempo a la función es la búsqueda por Insertion Sort, pero al ser estable y al no producir errores vale la pena usarla sacrificando un poco de tiempo.

Esta función fue probada con movies-small.csv

Requerimiento 5:

Complejidad temporal: El primer bucle es *for* i *in* range(0, tamanio) es un ciclo que va a iterar en toda la lista, lo cual nos da una complejidad de O(n), todo lo que sucede dentro de este bucle son comparaciones y asignaciones, las cuales tienen una complejidad de O(1), lo cual resulta que la primera parte de este código es un O(n), fuera del bucle suceden comparaciones para organizar los datos de salida, y algunos cálculos matemáticos, los cuales tienen una complejidad de O(1), por lo cual la mayor complejidad es de O(n) lo cual resulta en que la complejidad temporal es de **O(n)**

Tabla de tiempo:

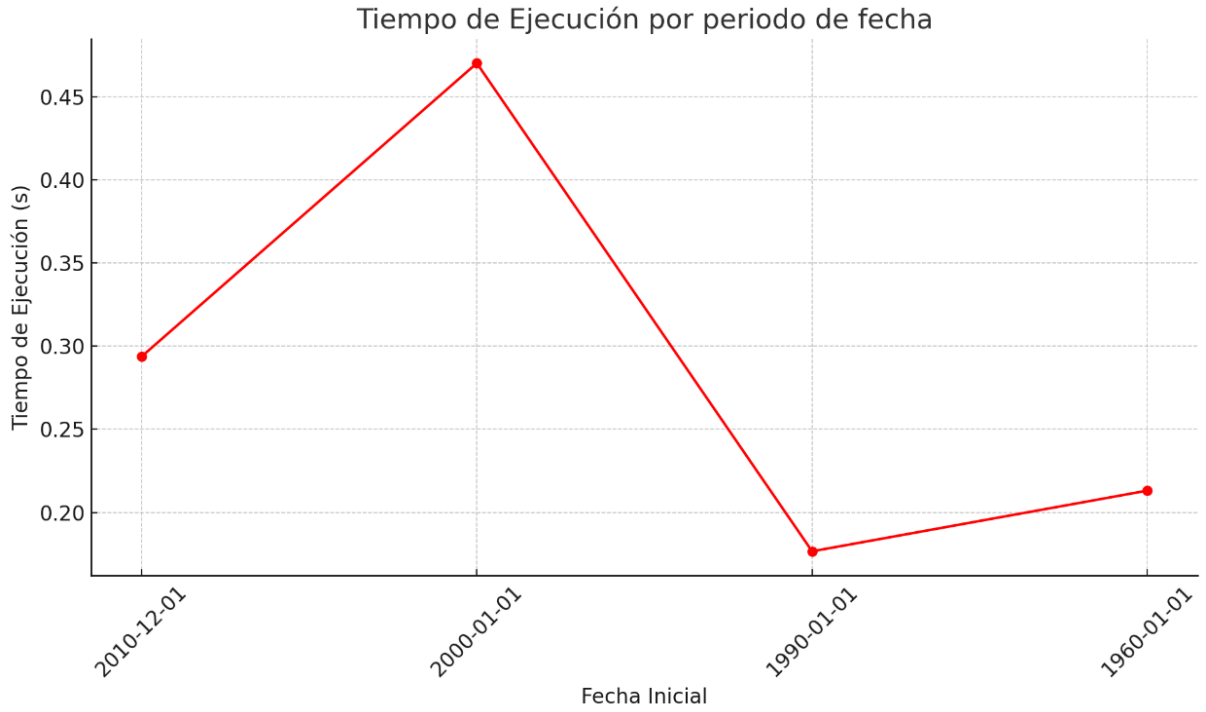


Tabla de comparación:

Rango de Fechas	Duración (min)	Tiempo de Ejecución (s)	Eficiencia (s/min)
2010-12-01 a 2017-06-25	90 - 150	0.293688	0.0021
2000-01-01 a 2017-01-01	80 - 160	0.470180	0.0029
1990-01-01 a 2017-01-01	60 - 200	0.176590	0.0009
1960-01-01 a 2017-01-01	1 - 300	0.213099	0.0007

Análisis final:

Se observa que el tiempo de ejecución aumenta de manera constante o proporcional a cambios en las duraciones o fechas, eso podría indicar una tendencia lineal. Por lo cual puede implicar que el análisis inicial está bien realizado y la función si es O(n)

Esta función fue probada con movies-small.csv

Requerimiento 6:

Complejidad temporal: Lo primero que se puede ver es que hay dos ciclos, uno de estos anidados (lo cual nos puede dar una clara pista de qué tipo de función es). El bucle externo recorre un rango de años desde anio_inicial hasta anio_final, lo cual implica que la range será la distancia de anio_final - anio_inial, lo cual nos da un rango de N, el bucle interno recorre todas las películas para ver si cumplen las condiciones, lo cual implica que el primer bucle recorre todo el catálogo para ver si la condición de estar en el rango de años se cumple, la otra busca que las otras características se cumplan en el tamaño de todo el catálogo, aunque lo que sucede adentro son operaciones O(1), como hay un ciclo anidado, quiere decir que la complejidad es de O(n^2)

Tabla de tiempo:

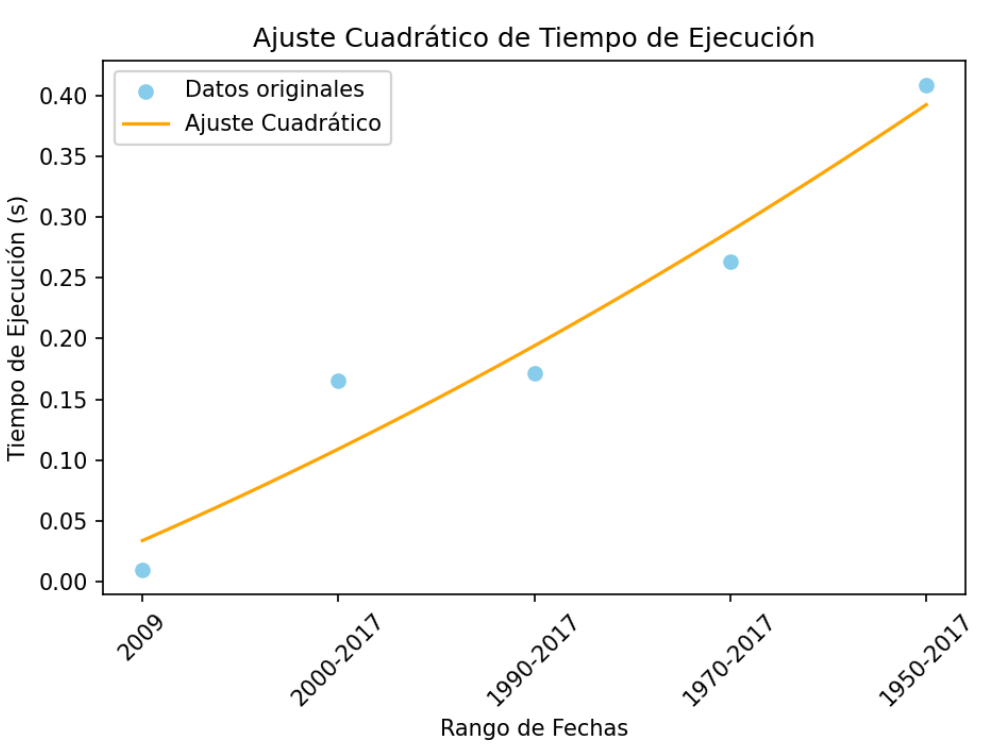


Tabla de comparación:

Idioma	Fecha Inicial	Fecha Final	Tiempo de Ejecución (s)
en	2009	2009	0.009694
en	2000	2017	0.164948
en	1990	2017	0.171137
en	1970	2017	0.263139
en	1950	2017	0.408851

Análisis final: Podemos ver que la función efectivamente es cuadrática, sobre todo la gráfica de la función nos permite ver cómo la gráfica crece de manera cuadrática (o eso parece) por su naturaleza, por ende queda comprobado que la naturaleza de la función efectivamente es O(n^2)
Esta función fue probada con movies-small.csv

Requerimiento 7:

Primero, para filtrar las películas dentro de un rango de años, es necesario recorrer todas las películas disponibles. Este paso tiene una complejidad de O(N), donde N es el número total de películas en el catálogo.

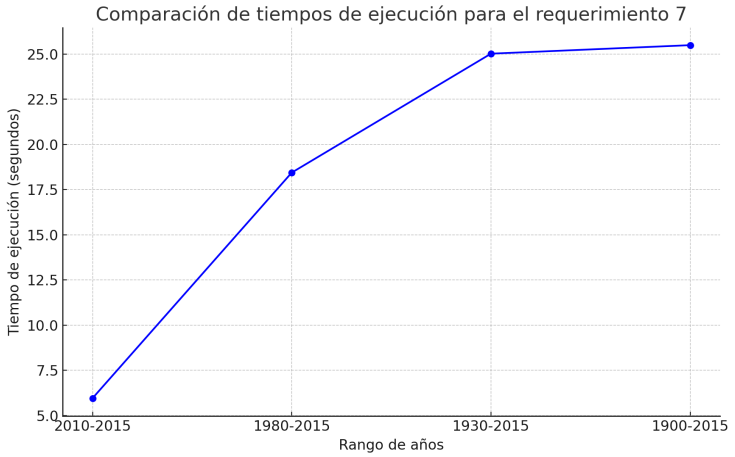
Además, para cada película, se debe verificar si alguna de sus compañías productoras coincide con la compañía buscada. Si cada película tiene un promedio de M compañías, se necesitará realizar M comparaciones por película.

En el peor de los casos, si es necesario revisar todas las películas y cada una tiene múltiples compañías productoras, la complejidad total será O(N * M), donde:

- N es el número total de películas,
- M es el número promedio de compañías productoras por película.

Si, en cambio, cada película tiene solo una compañía productora o se optimiza el proceso de búsqueda (usando estructuras de datos como conjuntos o diccionarios), la complejidad sería **O(N)**, ya que solo se necesitaría recorrer una vez la lista de películas y hacer una verificación rápida.

Tabla de tiempo:



Probado con la compañía Warner Bros., ejecutado en la base de datos small

Tabla de comparación:

Rango de años	Total de películas	Tiempo de ejecución (segundos)
2010-2015	7	5.951634
1980-2015	25	18.436520
1930-2015	38	25.017973
1900-2015	50	25.488973

Análisis final:

El análisis del requerimiento 7 muestra que los tiempos de ejecución crecen proporcionalmente al número de películas y al rango de años consultados, lo cual es consistente con la complejidad teórica de O(N * M). A medida que el número de películas aumenta, los tiempos de ejecución también incrementan, pero se estabilizan a partir de cierto punto.

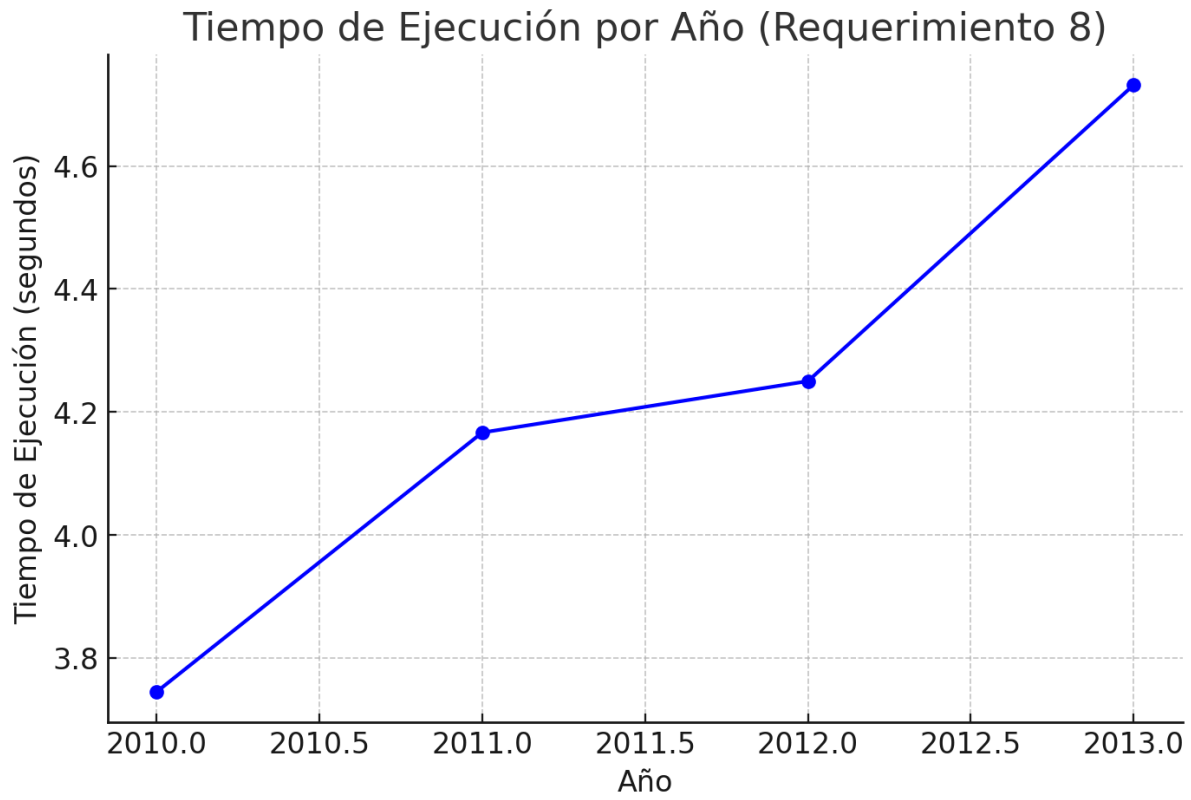
Esta función fue probada con movies-small.csv

Requerimiento 8 (Bono):

El análisis de complejidad se puede dividir en las siguientes partes:

1. Recorrido de las películas: La función recorre todas las películas una vez para filtrar las que cumplen con los criterios de año y género. Esto tiene una complejidad de O(N), donde N es el número total de películas en el dataset.
2. Verificación de géneros: En cada iteración, se verifica si la película pertenece al género especificado. Dado que cada película tiene una lista pequeña y fija de géneros (generalmente entre 1 y 3 géneros), esta verificación tiene un costo de O(1) para cada película. Aunque existe un pequeño coste adicional por la verificación, no cambia la complejidad total del algoritmo, que sigue siendo O(N).
3. Cálculo de estadísticas: Una vez filtradas las películas, se calculan las métricas como el promedio de duración, puntuación y se seleccionan la mejor y peor película. Esto también se hace en una sola pasada por las películas filtradas, lo que añade una complejidad de O(F), donde F es el número de películas filtradas. Como $F \leq N$, el impacto es mínimo y la complejidad sigue siendo O(N).

Tabla de tiempo:



Prueba hecha con la base de datos médium

Tabla de comparación:

	Año	Total Películas	Puntuación Promedio	Duración Promedio (n	Tiempo de Ejecución (
1	2010	114	5.94	86.87	3.744725
2	2011	151	5.7	85.75	4.166547
3	2012	134	5.29	88.49	4.250093
4	2013	160	5.84	81.61	4.731687

Análisis final:

Los resultados obtenidos coinciden con el análisis de complejidad **O(N)**. A medida que aumenta el número de películas procesadas, el tiempo de ejecución crece linealmente, lo que confirma la eficiencia del algoritmo.

Esta función fue probada con movies-small.csv