**ANÁLISIS DEL RETO**

Giuliana Volpi Muñoz 202123986 g.volpi

Viviana Lara Peña, 202122798, mv.lara

Santiago Najar, 202021647, s.najar

# **Premisa: Todas las pruebas fueron corridas con la base de datos de tamaño “large” en el dispositivo con las siguientes características:**

|  |  |
| --- | --- |
| Procesadores | 11th Gen Intel(R) Core (TM) i7-1165G7 2.80GHz |
| Memoria RAM (GB) | 16.0 GB |
| Sistema Operativo | Windows 11 Pro |

# **Carga de datos**

|  |  |
| --- | --- |
| **TIEMPO DE EJECUCIÓN** | **MEMORIA UTILIZADA** |
|  |  |

# **Requerimiento <1> Examinar películas en año determinado**

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Un año por el que se desea hacer la consulta, y el catálogo creado al cargar los datos |
| **Salidas** | Devuelve la tupla resultante del get donde la llave es el año ingresado por parámetro y el valor es la lista de diccionarios que fueron estrenados en dicho año. Adicionalmente se devuelven los tiempos y la memoria empleada. |
| **Implementado (Sí/No)** | Si se implementó, se hizo grupal. |

## **Análisis de complejidad**

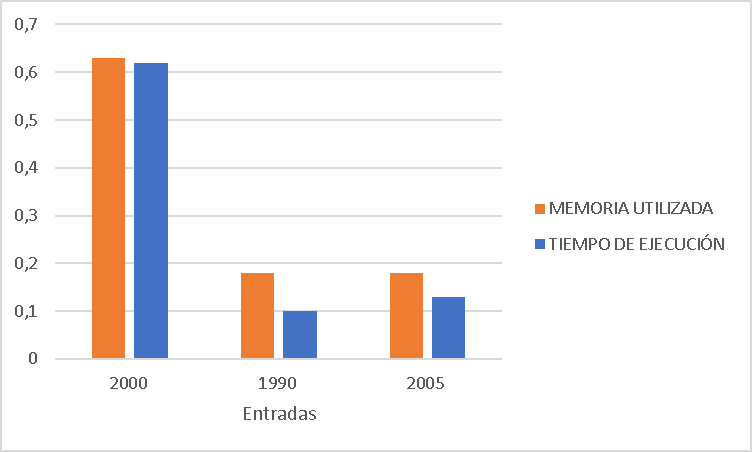
Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Con el contains se revisa si existe el año ingresado por parámetro en el catálogo. | O (1) |
| Con el get se guardan en formato de tupla las películas que fueron estrenadas en el año | O (1) |
| For (view) que recorre la lista de la respuesta actualizando los contadores de películas. | O(N) |
| ***TOTAL*** | ***O (N)*** |

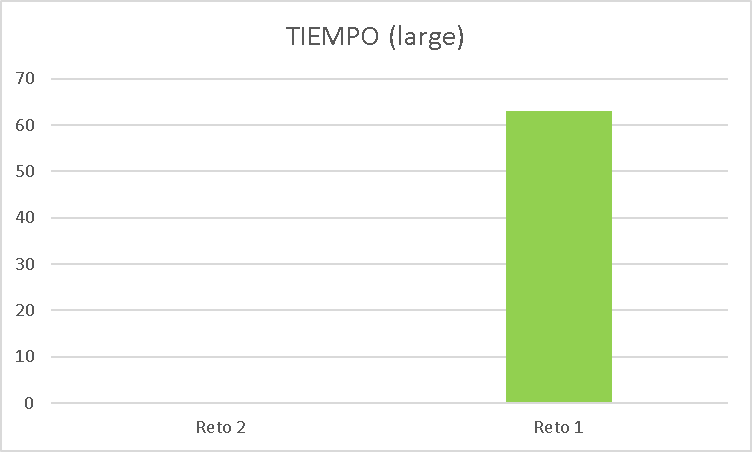
### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **TIEMPO DE EJECUCIÓN** | **MEMORIA UTILIZADA** |
| 2000 | 0.62 | 0.63 |
| 1990 | 0.10 | 0.18 |
| 2005 | 0.13 | 0.18 |



# **Comparación con Reto 1**



* En la gráfica se pueden comparar los tiempos de ejecución de las pruebas realizadas tanto para el reto 1 como para el reto 2 en este requerimiento, concluyendo que la diferencia entre las dos es amplia y el reto 2 es mucho más eficiente.
* Tanto en el reto 1 como en el reto 2, la complejidad para este requerimiento es de O(N) con la única diferencia de que en el reto 2 se hace uso de menos lineas de código.

# **Requerimiento <2> Programas de televisión agregados en un año**

Plantilla para el documentar y analizar cada uno de los requerimientos.

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | El usuario ingresa por parámetro el año por el que desea realizar la búsqueda y la función recibe el catálogo creado también. |
| **Salidas** | Devuelve la tupla resultante del get donde la llave es el año ingresado por parámetro y el valor es la lista de diccionarios que fueron estrenados en dicho año. Adicionalmente se devuelven los tiempos y la memoria empleada. |
| **Implementado (Sí/No)** | Si se implementó, se hizo grupal. |

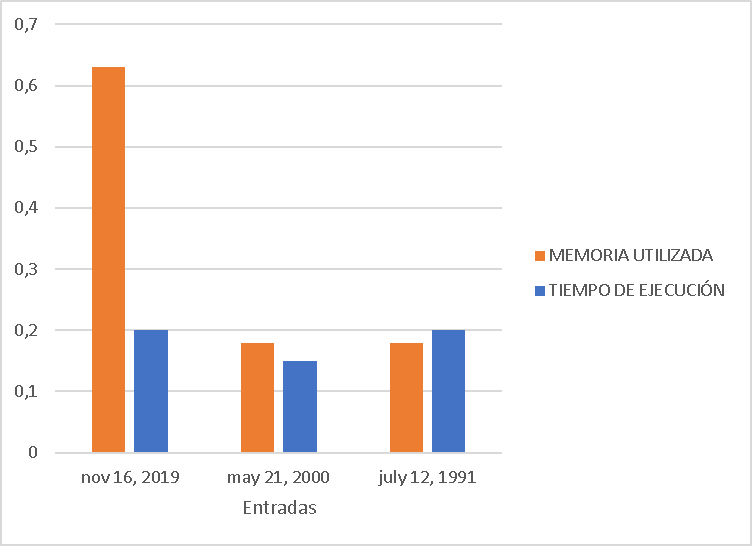
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Con el contains se revisa si existe el año ingresado por parámetro en el catálogo. | O (1) |
| Con el get se guardan en formato de tupla las películas que fueron estrenadas en el año | O (1) |
| For (view) que recorre la lista de la respuesta actualizando los contadores de programas de televisión. | O (N) |
| ***TOTAL*** | ***O (N)*** |

### **Tablas de datos**

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **TIEMPO DE EJECUCIÓN** | **MEMORIA UTILIZADA** |
| november 16, 2019 | 0.20 | 0.62 |
| may 21, 2000 | 0.15 | 0.18 |
| July 12, 1991 | 0.20 | 0.18 |



# **Comparación con Reto 1**

* En el reto 1 no se llegó a implementar la función con large y por lo tanto, no se pueden comparar los tiempos.
* Tanto en el reto 1 como en el reto 2, la complejidad para este requerimiento es de O(N) con la única diferencia de que en el reto 2 se hace uso de menos lineas de código.

# **Requerimiento <3> Filtro por actor**

Plantilla para el documentar y analizar cada uno de los requerimientos.

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Entra el nombre de un actor y el catálogo creado al cargar los datos. |
| **Salidas** | Devuelve la tupla resultante del get donde la llave es el nombre del actor ingresado por parámetro y el valor es la lista de diccionarios que fueron estrenados en dicho año. Adicionalmente se devuelven los tiempos y la memoria empleada. |
| **Implementado (Sí/No)** | Si se implementó y lo hizo Viviana Lara |

## **Análisis de complejidad**

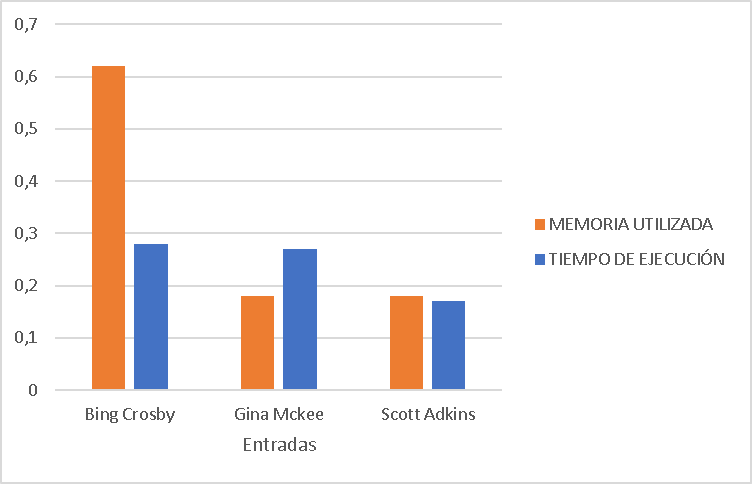
Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Con el contains se revisa si existe el actor ingresado por parámetro en el catálogo. | O (1) |
| Con el get se guardan en formato de tupla las películas que tienen en su cast el actor buscado | O (1) |
| For (view) que recorre la lista de la respuesta actualizando los contadores de programas de televisión y películas donde está el actor buscado. | O (N) |
| ***TOTAL*** | ***O (N)*** |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **TIEMPO DE EJECUCIÓN** | **MEMORIA UTILIZADA** |
| Bing Crosby | 0.28 | 0.62 |
| Gina McKee | 0.27 | 0.18 |
| Scott Adkins | 0.17 | 0.18 |



# **Comparación con Reto 1**

* En el reto 1 no se llegó a implementar la función con large y, por lo tanto, no se pueden comparar los tiempos.
* La complejidad para este requerimiento en el reto 1 fue de O(N^2) pues se hizo uso de dos for en cascada, es decir, anidado uno al otro. Mientras que la complejidad en el reto 2 es de O(N), destacando que se hizo en menos lineas de código.

# **Requerimiento <4> Filtro por género**

Plantilla para el documentar y analizar cada uno de los requerimientos.

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Entra un género y el catálogo creado al cargar los datos. |
| **Salidas** | Devuelve la tupla resultante del get donde la llave es el género ingresado por parámetro y el valor es la lista de diccionarios que fueron estrenados en dicho año. Adicionalmente se devuelven los tiempos y la memoria empleada. |
| **Implementado (Sí/No)** | Si se implementó y lo hizo Santiago Najar |

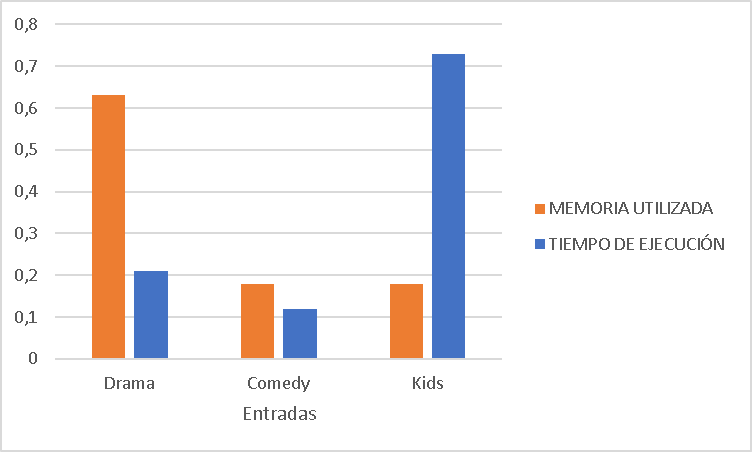
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

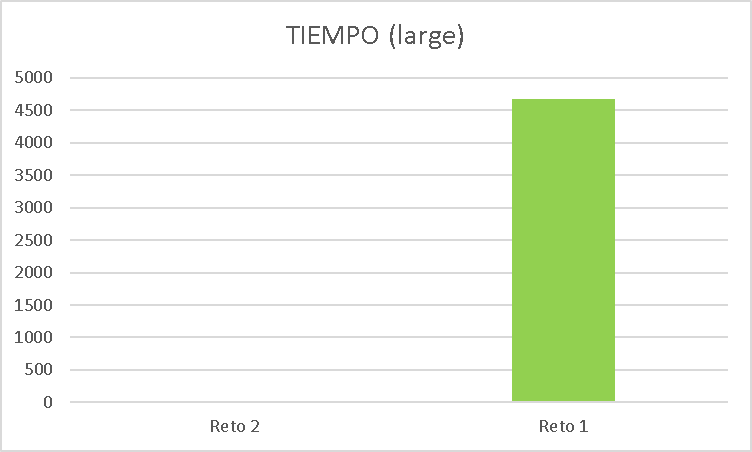
|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Con el contains se revisa si existe el género ingresado por parámetro en el catálogo. | O (1) |
| Con el get se guardan en formato de tupla las películas que hacen parte del género buscado. | O (1) |
| For (view) que recorre la lista de la respuesta actualizando los contadores de programas de televisión y de películas que tienen el género deseado. | O (N) |
| ***TOTAL*** | ***O(N)*** |

### **Tablas de datos**

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **TIEMPO DE EJECUCIÓN** | **MEMORIA UTILIZADA** |
| Drama | 0.21 | 0.63 |
| Comedy | 0.12 | 0.18 |
| Kids | 0.73 | 0.18 |



# **Comparación con Reto 1**



* En la gráfica se pueden comparar los tiempos de ejecución de las pruebas realizadas tanto para el reto 1 como para el reto 2 en este requerimiento, concluyendo que la diferencia entre las dos es bastante amplia y el reto 2 es mucho más eficiente que no se alcanza a ver que tiempo usó.
* La complejidad para este requerimiento en el reto 1 fue de O(N^2) pues se hizo uso de dos for en cascada, es decir, anidado uno al otro. Mientras que la complejidad en el reto 2 es de O(N), destacando que se hizo en menos lineas de código.

# **Requerimiento <5> Filtro por país**

Plantilla para el documentar y analizar cada uno de los requerimientos.

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Entra un país y el catálogo creado al cargar los datos. |
| **Salidas** | Devuelve la tupla resultante del get donde la llave es país ingresado por parámetro y el valor es la lista de diccionarios que fueron estrenados en dicho año. Adicionalmente se devuelven los tiempos y la memoria empleada. |
| **Implementado (Sí/No)** | Si se implementó y lo hizo Giuliana Volpi |

## **Análisis de complejidad**

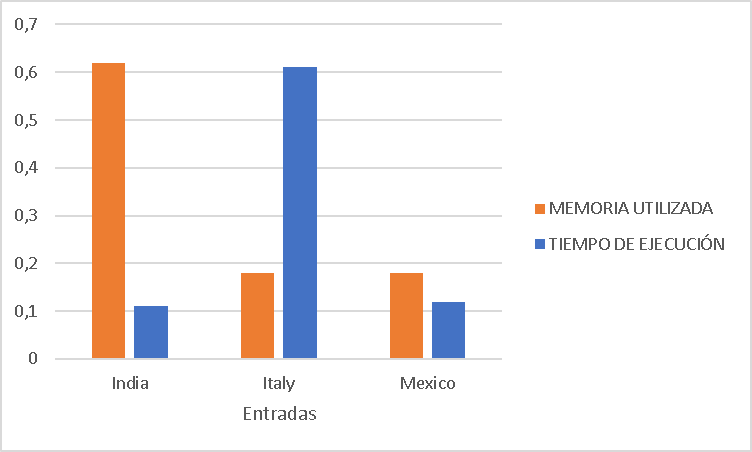
Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Con el contains se revisa si existe el país ingresado por parámetro en el catálogo. | O (1) |
| Con el get se guardan en formato de tupla las películas que fueron producidas en el país | O (1) |
| For (view) que recorre la lista de la respuesta actualizando los contadores de programas de televisión y películas que fueron producidas en el país que se está filtrando. | O (N) |
| ***TOTAL*** | ***O (N)*** |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **TIEMPO DE EJECUCIÓN** | **MEMORIA UTILIZADA** |
| India | 0.11 | 0.62 |
| Italy | 0.61 | 0.18 |
| Mexico | 0.12 | 0.18 |



# **Comparación con Reto 1**

* En el reto 1 no se llegó a implementar la función con large y, por lo tanto, no se pueden comparar los tiempos.
* Tanto en el reto 1 como en el reto 2, la complejidad para este requerimiento es de O(N) con la única diferencia de que en el reto 2 se hace uso de menos lineas de código.

# **Requerimiento <6> Filtro por director**

Plantilla para el documentar y analizar cada uno de los requerimientos.

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Entra el nombre de un director y el catálogo creado al cargar los datos. |
| **Salidas** | Devuelve la tupla resultante del get donde la llave es el nombre del director ingresado por parámetro y el valor es la lista de diccionarios que fueron estrenados en dicho año. Adicionalmente se devuelven los tiempos y la memoria empleada. |
| **Implementado (Sí/No)** | Si se implementó y fue grupal. |

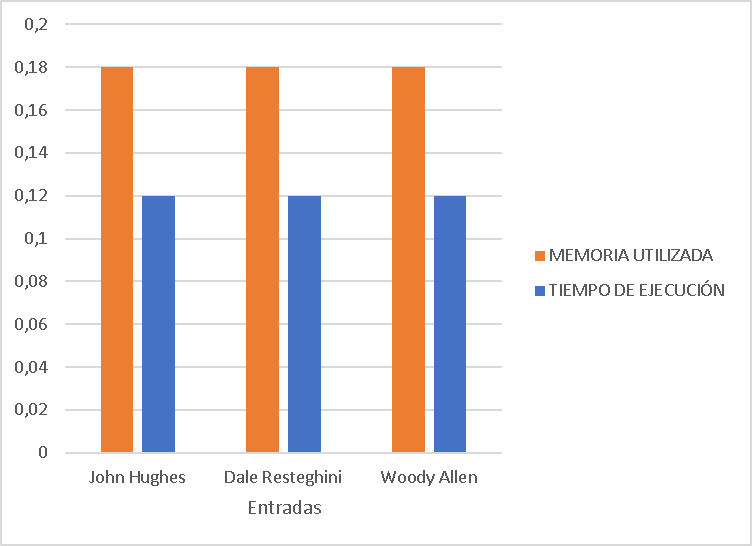
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| **For que recorre cada género en la lista de géneros** | **O(N)** |
| Con el contains se revisa si existe el director ingresado por parámetro en el catálogo. | O (1) |
| Con el get se guardan en formato de tupla las películas cuyo director es el deseado. | O (1) |
| For anidado que recorre cada película en la lista resultante de películas que tienen el director deseado | O(N) |
| For anidado que recorre la lista de géneros dentro de cada una de las películas de la lista resultante descrita en el paso anterior. | O(N) |
| For que recorre los géneros que se han ido agregando a un diccionario vacío que devuelve los géneros que cumplen con los dos filtros realizados anteriormente, si el género no está en la lista que se devolverá al usuario, se agrega. | O(N) |
| For (view) que recorre la lista de la respuesta actualizando los contadores por cada tipo de plataforma. | O (N) |
| ***TOTAL*** | ***O(N^2)*** |

### **Tablas de datos**

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **TIEMPO DE EJECUCIÓN** | **MEMORIA UTILIZADA** |
| John Hughes | 0.12 | 0.18 |
| Dale Resteghini | 0.12 | 0.18 |
| Woody Allen | 0.12 | 0.18 |



# **Comparación con Reto 1**

* En el reto 1 no se llegó a implementar la función con large y, por lo tanto, no se pueden comparar los tiempos.
* Tanto en el reto 1 como en el reto 2, la complejidad para este requerimiento es de O(N^2) con la única diferencia de que en el reto 2 se hace uso de menos lineas de código.

# **Requerimiento <7> Top N géneros**

Plantilla para el documentar y analizar cada uno de los requerimientos.

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Entra el N, es decir, el número de géneros a comparar y el catálogo creado al cargar los datos. |
| **Salidas** | Devuelve una lista con las N producciones que el usuario deseaba conocer, junto con los contadores clasificados por género y las medidas de tiempo y memoria utilizada. |
| **Implementado (Sí/No)** | Si se implementó y fue grupal. |

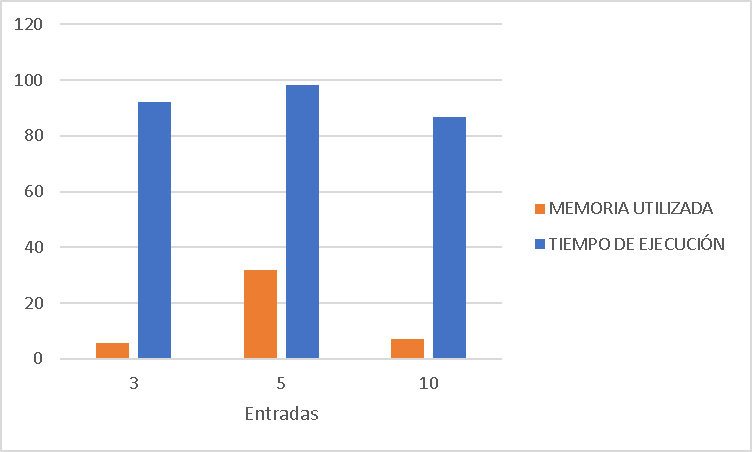
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

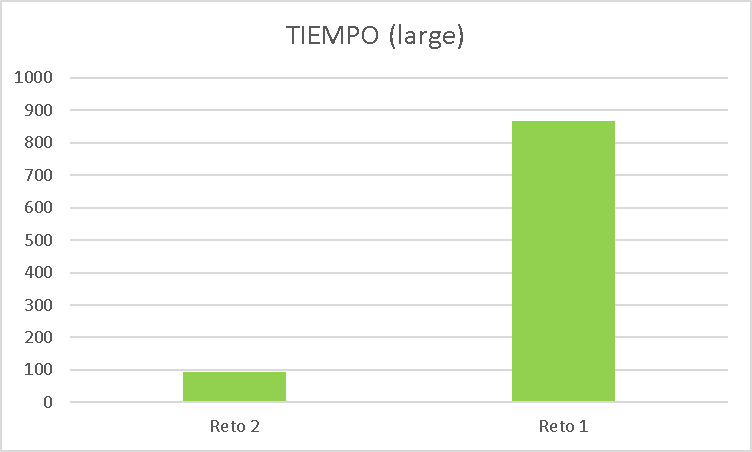
|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Con el kyeset se obtiene una lista con todas las llaves que son los géneros existentes. | O (1) |
| For que recorre la lista de géneros | O (N) |
| Con el get se guardan en formato de tupla las películas cuyo género es el que el for anterior está recorriendo. | O (1) |
| For anidado que recorre cada película en la lista resultante de películas por género donde se actualizan los contadores por tipo de producción y de plataforma. | O(N) |
| ***TOTAL*** | ***O(N^2)*** |

### **Tablas de datos**

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **TIEMPO DE EJECUCIÓN** | **MEMORIA UTILIZADA** |
| 10 | 86.79 | 7.13 |
| 5 | 98.20 | 31.78 |
| 3 | 92.17 | 5.89 |



# **Comparación con Reto 1**



* En la gráfica se pueden comparar los tiempos de ejecución de las pruebas realizadas tanto para el reto 1 como para el reto 2 en este requerimiento, concluyendo que la diferencia entre las dos es amplia y el reto 2 es mucho más eficiente.
* Tanto en el reto 1 como en el reto 2, la complejidad para este requerimiento es de O(N^2) con la única diferencia de que en el reto 2 se hace uso de menos lineas de código.