DOCUMENTO ANÁLISIS RETO 2

Requerimiento 2 - Javier Cerino Cod 202020873
Requerimiento 3 - Marco Zuliani Cod 202022412

Requerimiento 1:
Alumnos que trabajaron en este requerimiento: Marco Zuliani (código estudiante: 202022412, Usuario Uniandes: m.zuliani, Nombre de usuario Git: poloiva), Javier Cerino (código estudiante: 202020873, Usuario Uniandes: j.cerino, Nombre de usuario Git: 2jc26)
análisis de complejidad:
O($N^{1.25}$) en el mejor de los casos O(N log N)
Para este requerimiento comenzamos encontrando los videos que se encuentran en una categoría dada con los métodos de los maps. Decidimos realizar este procedimiento tanto para las categorías como para los países. Una vez concluido esto realizamos un Shell sort que cuenta con un orden de complejidad promedio 1.25 N en el mejor de los casos N log N y en el peor de los casos 3/2 N, en la lista de videos de la categoría encontrada por la búsqueda en la tabla de hash. Una vez se ordena por views decidimos comenzar a comparar de manera linear los elementos pertenecientes a esta lista de objetos comparando si se encuentran en la lista del país de interés con la función isPresent en caso de devolver un valor verdadero se agregan a la lista que se devolverá al final.

Requerimiento 2:
Alumnos que trabajaron en este requerimiento: Javier Cerino (código estudiante: 202020873, Usuario Uniandes: j.cerino, Nombre de usuario Git: 2jc26)
análisis de complejidad:
O($N^{1.25}$) en el mejor de los casos O(N log N)
Para este requerimiento se comienza encontrando los videos que se encuentran en un país dado con los métodos de los maps. Una vez concluido esto realizamos un Shell sort en la lista de videos de la categoría encontrada por la búsqueda en la tabla de hash. Una vez se ordena por id se realiza un conteo de orden de complejidad N, donde N es la longitud total de la lista ordenada por id del país de interés conseguido gracias a la tabla de hash, para poder encontrar cual es el video que más veces ha sido tendencia en ese país. Una vez se concluye el conteo y se devuelve cual es video que más ha aparecido en la lista se realiza una búsqueda binaria para poder acceder a todos los datos del video para poder imprimir todos los datos que se necesitan la interfaz.

Requerimiento 3:
Alumnos que trabajaron en este requerimiento: Marco Zuliani (código estudiante: 202022412, Usuario Uniandes: m.zuliani, Nombre de usuario Git: poloiva)
análisis de complejidad:
O($N^{1.25}$) en el mejor de los casos O(N log N)
Para este requerimiento se empieza encontrando y obteniendo los videos que pertenecen a la categoría especificada por el usuario a través del método get de los maps sobre el map de categorías que se encuentra en el catálogo. La lista de videos obtenida anteriormente se ordena con el uso de Shell sort por id o por título, según la opción elegida por el usuario, sobre si se debe comparar por título o por id de los videos. Cuando ya se tiene la lista ordenada por id se realiza un conteo de orden de complejidad N, donde

N es la longitud total de la lista ordenada por id de la categoría de interés, para poder encontrar cual es el video que más veces ha sido tendencia en la categoría especificada. Luego se devuelve cual es el id o título del video que más veces ha aparecido en la lista para realizar una búsqueda binaria que permite acceder a todos los datos del video con el fin de imprimirlos en la interfaz.

Requerimiento 4:
Alumnos que trabajaron en este requerimiento: Javier Cerino (código estudiante: 202020873, Usuario Uniandes: j.cerino, Nombre de usuario Git: 2jc26), Marco Zuliani (código estudiante: 202022412, Usuario Uniandes: m.zuliani, Nombre de usuario Git: poloiva)
análisis de complejidad:
O($N^{1.25}$) en el mejor de los casos O(N log N)
Para este requerimiento comenzamos encontrando los videos que se encuentran en un país dado con los métodos de los maps. Una vez concluido esto realizamos un Shell sort en la lista de videos de la categoría encontrada por la búsqueda en la tabla de hash. Una vez se ordena por likes creamos una nueva lista de tipo arraylist debido a su utilidad para nuestro caso de estudio. En esta nueva lista se rellenará con los videos que cumplan con los tags de interés y además se verifica que los videos ya no existan en la lista para evitar repeticiones. Una vez realizado esto se devuelven los resultados y se imprimen en view.py