DOCUMENTO ANÁLISIS RETO 1

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(NlogN)

Encontramos la complejidad de N log N en el algoritmo de ordenamiento de tipo Merge que se utiliza al inicio del código para ordenar por países. Este ordenamiento está realizado con el objetivo de realizar una futura búsqueda binaria (complejidad de log(N)) para encontrar el país que ingresa el usuario por parámetro. Luego de la búsqueda binaria se realiza una sublista que contiene solamente los videos pertenecientes al país de interés. Una vez creada esta sublista se procede a ordenar esta nueva lista por orden de categorías. Este ordenamiento también tiene un orden de complejidad (n log n) no obstante, toca considerar que este n es menor al N inicial ya que estamos trabajando con una sublista de la inicial. Una vez ordenada esta última lista de países se procede a realizar nuevamente una búsqueda binaria para encontrar la categoría que se ingresó por parámetro para así poder tener acceso nuevamente a los videos que pertenecen a la categoría de interés para una vez más realizar un nuevo ordenamiento por vistas para dar el resultado del primer requerimiento.

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(NlogN)

Encontramos la complejidad de N log N en el algoritmo de ordenamiento de tipo Merge que se utiliza al inicio del código para ordenar por países. Este ordenamiento está realizado con el objetivo de realizar una futura búsqueda binaria (complejidad de log(N)) para encontrar el país que ingresa el usuario por parámetro. Luego de la búsqueda binaria se realiza una sublista que contiene solamente los videos pertenecientes al país de interés. Una vez creada esta sublista se procede a ordenar esta nueva lista por id. Este ordenamiento también tiene un orden de complejidad (n log n) no obstante, toca considerar que este n es menor al N inicial ya que estamos trabajando con una sublista de la inicial. Una vez ordenada esta última lista de países se procede a realizar un contento de orden de complejidad n donde n es la longitud total de la lista ordenada por id del país de interés 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(NlogN)

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(NlogN)

Encontramos la complejidad de N log N en el algoritmo de ordenamiento de tipo Merge que se utiliza al inicio del código para ordenar por países. Este ordenamiento está realizado con el objetivo de realizar una futura búsqueda binaria (complejidad de log(N)) para encontrar el país que ingresa el usuario por parámetro. Luego de la búsqueda binaria se realiza una sublista que contiene solamente los videos pertenecientes al país de interés. Una vez creada esta sublista se procede a ordenar esta nueva lista por likes. Este ordenamiento también tiene un orden de complejidad (n log n) no obstante, toca considerar que este n es menor al N inicial ya que estamos trabajando con una sublista de la inicial. Una vez terminado este ordenamiento 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