María Castro 202020850 m.castroi@uniandes.edu.co
Valentina Calderón 202020771 v.calderonm@uniandes.edu.co

Sustentación Reto 1

Función 1/Carga de Datos:

Objetivo: Cargar la información del archivo de videos y categorías. Al final de cada carga reportar el total de registros de videos cargados del archivo, mostrar información del primer video cargado, imprimir la lista de categorías cargadas mostrando su id y nombre.

MVC:

View:

1. Llama a la función initCatalog del controlador
2. Llama a la función loadData del controlador
3. Después de la carga, imprime el primer video de la lista de videos y todos los elementos de la lista de category ids

Controller:

1. Llama a la función initCatalog del modelo
2. loadData() llama a loadVideos() y loadCategoryIds()
    a. loadVideos() recorre cada línea del archivo de videos O(n) y llama al modelo addVideo() para añadir los datos a la lista de 'videos' del catálogo.
    b. loadCategoryIds() recorre cada línea del archivo de categorías O(n) y llama al modelo para que añada cada nueva categoría.

Model:

1. initCatalog() crea la estructura del catálogo de videos.
    a. 'videos' y 'category-id' – listas que debe tener el catálogo
    b. *by_countries' & 'by_categories' – este par de listas las adicionamos para mejorar la eficiencia de los requerimientos. La lista de países tiene almacenado todos los países del archivo con todos los videos que correspondan a ese país. Los mismo aplica para la lista de categorías pero en este caso es con los ids.
        i. **Aunque esto implica más uso de RAM y un poco más de tiempo al cargar los datos, al momento de necesitar solo los videos de un país o una categoría específica, esta lista se podrá obtener con una complejidad de O(n) en el peor caso ya que tendrá que recorrer la lista de países o categorías hasta encontrar la deseada. Añadir estas dos listas permite tener un acceso directo a una filtración por categoría o por país, y no implica tener que hacer un filtro cada vez que se necesiten videos de un país o una categoría específica.**

        ii. **Además tener esta sublistas permite hacer recorridos y ordenamientos más cortos sobre los datos, aunque estos sigan siendo lineales se tiene en cuenta sólo una fracción de los datos y no los 35000+.**
2. addVideos() agrega el video que recibe al final de la lista de 'videos' del catalgo (addLast() - O(1)). Llama a las funciones addVideoCountry() y addVideoCategory()
    a. addVideoCountry() se encarga de adicionar los datos a un país de la lista de paises. (getElement O(1), addLast O(1))
    b. addVideoCategory() se encarga de adicionar los datos a una categoría de la lista de categorías. (getElement O(1), addLast O(1))
    c. Cada lista de país y categoría se modela con un diccionario que tiene 2 llaves, una para identificar el país ('name') o la categoria ('id'), y una para la lista de videos como tal ('videos')
        i. Esto permite poder acceder fácilmente a los videos dado un país o una categoría.
3. addCategory() recibe los datos de una categoría, los convierte en un diccionario con el 'id' y el 'name', y añade esto al final de la lista de category-ids

Complejidad:
O(v) + O(c) → O(n)
- v es la cantidad de videos - lineal
- c es la cantidad de categorías - lineal

Función 2 / Requerimiento 1:

Objetivo: Conocer cuáles son los n videos con más views que son tendencia en un determinado país, dada una categoría específica.

MVC:
View:
1. Se le pregunta al usuario la cantidad de videos, el país, y la categoría.
2. Se llama a una función del controlador que retorna la lista ya filtrada y sorteada con los videos con más views del país y categoría solicitada.
3. Imprime la siguiente información de los videos que se encuentran en el top: trending:date, title, channel_title, publish_time, views, likes, dislikes. (O(n) – imprimir)

Controller:
1. Se llama a una función del modelo getId() la cual le retorna el id asociado con ese nombre de categoría.

2. Se llama a una función del modelo getCategory() la cual retorna la lista de videos asociada con el categoryId obtenido.
   que va añadiendo cada categoría que coincide en una lista de categorías.
3. Se llama a una función del modelo sortViews que devuelve la lista sorteada según la cantidad de views.
4. Se llama a una función del modelo findTopsCountryCategory() que devuelve una lista con los x videos con más views.

Model:

1. getId() - Compara la lista de categorías posibles y busca si la ingresada por parámetro está, al encontrarla retorna el category_id o None si no la encontró. (O(n))
2. getCategory() - Busca la lista correspondiente al ID ingresado por parámetro esta presente en la lista de 'by_categories'. Devuelve la lista o None si no la encuentra O(1)
3. sortViews() - Se hace un mergesort para organizar la lista comparando los views. O(nlogn)
4. Se crea una nueva lista vacía. Recorre los elementos de la lista anteriormente ordenada hasta que ésta ya tenga los top x videos o se haya llegado al fin de la lista ordenada. Un video se añade a la nueva lista si su país es el mismo que el que desea el usuario. PeorCaso → O(n)

Complejidad:

O(n) + O(1) + O(nlogn) + O(n) + O(n)

O(nlogn) → Complejidad más grande

Función 3 / Requerimiento 2 (Valentina):

Objetivo: Conocer cuál es el video que más días ha sido trending para un país específico.

MVC:

View:

1. Se le pregunta al usuario el país a consultar el video trending por más días.
2. Se llama a la función del controlador getCountry() que retorna el país de la categoría ingresada por el usuario.
3. Si existe el país se llama a la función del controlador topVidByCountry() que retorna el video que más días ha sido tendencia del país ingresado por parámetro.
4. Imprime la siguiente información del vídeo que ha sido trending por más días: title, channel_title, coutry, número de días como tendencia.

Controller:

1. getCountry() llama a la función del modelo getCountry() que retorna los datos de los videos del país solicitado por parámetro.
2. topVidByCountry() se encarga de:

a. Llama a la función del modelo sortVideoId() que devuelve una lista ordenada por los video_id .
b. Llama a la función del modelo findTopCountry() a cual retorna la información del video que más días ha sido tendencia y la cantidad de días tendencia del país ingresado por parámetro

Model:

1. getCountry(): compara la lista de países posibles y busca si la ingresada por parámetro está, al encontrarla retorna los datos del país o None si no la encontró. (O(n))
2. sortVideoId: Se hace un mergesort para organizar la lista comparando los video_ids. O(nlogn)
3. findTopVideoCountries():
   - Crea una lista en la que se guardará cada video y las veces que este aparece dentro de la lista ordenada. Recorreré la lista ordenada y para cada video revisa si el siguiente es el mismo (tienen el mismo país), si lo son le suma uno a la cantidad de repeticiones. Si es un video diferente, añade a la lista el video con sus datos, y la cantidad de veces que este apareció. (O(n)). Al final este recorrido, se recorre la lista que se creó con cada video y sus repeticiones para determinar cual tiene la mayor cantidad de repeticiones (O(n)).
   - Se retorna los datos del video con más elemento y su cantidad de repeticiones.

Complejidad:
O(n) + O(nlogn) + O(n) + O(n)
O(nlogn) → Complejidad más grande

Función 4 / Requerimiento 3 (María):

Objetivo: Conocer cuál es el video que más días ha sido trending para una categoría específica.

MVC:
View:

1. Se le solicita al usuario ingresar la categoría a consultar el video trending por x días.
2. Se llama a la función del controlador getId() que retorna el id de la categoría ingresada por el usuario.
3. Si existe la categoría se llama a la función del controlador topVidByCategory() que retorna el video que más días ha sido trending.
4. Imprime la siguiente información del vídeo que ha sido trending por más días: title, channel_title, category_id, número de días como tendencia

Controller:

1. getId: llama a la función del modelo getId()
2. topVidByCategory()
    a. llama la función del modelo getCategory() la cual retorna los datos ded los videos de la categoria deseada
    b. Llama a la función del modelo sortVideoId() la devuelve una lista ordenada por los video_id
    c. Llama la función del modelo findTopVideo() la cual retorna la información del video que más días ha sido tendencia y la cantidad de días tendencia.

Model:

1. getId(): compara la lista de categorías posibles y busca si la ingresada por parámetro está, al encontrarla retorna el category_id o None si no la encontró. (O(n))
2. getCategory(): busca la lista correspondiente al ID ingresado por parámetro esta presente en la lista de 'by_categories'. Devuelve la lista o None si no la encuentra O(1)
3. sortVideoId(): Se hace un mergesort para organizar la lista comparando los video_ids. O(nlogn)
1. findTopVideo(): crea una lista en la que se guardará cada video y las veces que este aparece dentro de la lista ordenada. Recorreré la lista ordenada y para cada video revisa si el siguiente es el mismo (tienen el mismo ID), si lo son le suma uno a la cantidad de repeticiones. Si es un video diferente, añade a la lista el video con sus datos, y la cantidad de veces que este apareció. (O(n)). Al final este recorrido, se recorre la lista que se creo con cada video y sus repeticiones para determinar cual tiene la mayor cantidad de repeticiones (O(n)). Se retorna los datos del video con más elemento y so cantidad de repeticiones.
    a. *No se tiene en cuenta los elementos que tengan como video_id '#NAME?". Aunque estos videos se podrían tener en cuenta si también se utilizará como criterio el title del vídeo, hacer esta doble verificación implicaría volver a recorrer toda la lista y comprar el elemento actual con el título de todos los videos. (Como la lista está organizada por video_id, nada garantiza que antes del elemento actual no haya habido videos con el mismo título). Este doble recorrido doble de la lista ordenada, O(n^2), afecta significativamente la eficiencia.*

Complejidad:

O(n) + O(1) + O(nlogn) + O(n) + O(n)

O(nlogn) → Complejidad más grande

Función 5 / Requerimiento 4:

Objetivo: Conocer cuáles son los n videos diferentes con más likes en un país con un tag específico.

MVC:

View:

1. Se le solicita al usuario que ingrese la cantidad, el país y el tag a consultar de videos con más likes.
2. Se llama a una función del controller getCountry() que retorna una lista de videos pertenecientes al país solicitado.
3. Si se encuentra el paísL, se llama a una función del controller que retorna una lista con los videos con más likes dados un pais y un tag. listVidTag()
4. Imprime la siguiente información de los videos que se encuentran en el top de likes: title, channel_title, publish_time, views, likes, dislikes, tags. (O(n) – imprimir)

Controller:

1. getCountry() - Se llama a una función del modelo que al encontrar que el país ingresado por parámetro coincide con uno de la lista inicial de países, devuelve el país.
2. listVidTag()
    a. llama a la función del model findWithTags() que retorna una lista filtrada con los videos que tenga dentro de sus tags el tag deseado por el usuario.
    b. Llama a la función del modelo sortLikes() la cual retorna una lista ordenada por likes
    c. Llama a la función del modelo findMostLikes() la cual retorna la lista con los x videos con más likes.

Model:

1. findWithTags() - Crea una lista vacía. Recorre todos los videos de la lista que recibe (que ya está filtrada por paises) y si estos tiene el tag deseado los agrega a la lista nueva. Devuelve la lista filtrada por tags. O(n)
2. sortLikes() - Se hace un mergesort para organizar la lista comparando los likes. O(nlogn)
3. findMostLikes() - se crea una lista vacía que tendrá los x top videos, que comprara por video id, a esta se le agrega el último elemento de la lista ordenada por likes. Se recorre la lista de todos los elementos organizados por likes hasta que ya se tengan los top x elementos o se llegue al principio de la lista (se recorre al revés ya que los views se ordena de menor a mayor) O(n). Se añade una video a lista de top si no existe un video con el mismo video_id - (isPresent (O(n), n máx será 1 menos que la cantidad de videos que solicite el usuario).

Complejidad:

$O(n_1) + O(nlogn) + O(n_2)$

O(nlogn) → Mayor complejidad

n_1 número de elementos que tiene el tag

n_2 número de elementos que ta hay en la lista de los top (depende de cuantos elementos haya pedido el usuario y cuantos ya se han agregado a esta lista)