Análisis de Complejidad Temporal
Reto 1

Juan Camilo Colmenares Ortiz - 202011866 - j.colmenares@uniandes.edu.co
Juan Andrés Ospina Sabogal - 202021727 - ja.ospinas1@uniandes.edu.co

Requerimiento 1

Para este requerimiento se organizó la información en **catalog[‘country_videos’]**. Este catálogo se construye a partir de un Array List y tiene la información organizada primero por views, después por categorías y, por último, por países. La organización se da por el algoritmo de ordenamiento Merge Sort.

El requerimiento utiliza la función **find_position_category()**, cuya complejidad temporal es **O(n),** ya que recorre toda la lista de categorías del archivo category-id.csv. Esta función recibe como parámetro la categoría que quiere el usuario y retorna el ID de esa categoría, de este modo podemos identificar la categoría en la lista de videos.

Adicional a esto, se creó el arreglo **catalog[‘country_map’]**. Este arreglo en **catalog** surge al recorrer **catalog[‘country_videos’]** por lo que la complejidad temporal de su creación es **O(n)**. Este mantiene la posición en donde se encuentran los diferentes países. Por ejemplo, si los elementos del país Canadá están desde la posición 200 hasta la 300, el catálogo nos dará esa información. Así podremos llegar a la posición necesaria y, una vez allí, se itera óolo el fragmento de la lista necesario para encontrar la información que nos pide el usuario.

Requerimiento 2 (Juan Camilo Colmenares Ortiz - 202011866)

Para este requerimiento se organizó la información en el **catalog[‘videos’]**, este catálogo se construye a partir de un Array List y tiene la información primero organizada por la id de los videos y después por países. La organización se da por el algoritmo de ordenamiento Merge Sort.

En este requerimiento se usa un catálogo descrito en el requerimiento 1 llamado **catalog[‘country_map’]**, con ayuda de este solo recorremos la fracción del país que nos pide el usuario una vez y de este modo hallamos el video con más días de trending. Es por esto que este requerimiento tiene un comportamiento **O(n)**.

Requerimiento 3 (Juan Andrés Ospina Sabogal - 202021727)

Para este requerimiento se organizó la información con un TAD Lista de tipo arreglo (Array List). Este arreglo se encuentra con la llave **'category_videos'** en **catalog** (**catalog['category_videos']**). Para hacer más eficiente la implementación del requerimiento, se ordenó este arreglo con el ordenamiento recursivo Merge Sort, tomándose como referencia para el orden el ID de la categoría de cada video.

El tener **catalog['category_videos']** ordenado por ID de categoría permite determinar el número de veces que un video de una categoría específica ha estado en tendencia de forma eficiente. El algoritmo desarrollado inicia por comenzar a recorrer **catalog['category_videos']**, hasta que se topa con un elemento (video) de la categoría deseada. Una vez allí, realiza un conteo de las apariciones de cada video y archiva dicha información en un diccionario auxiliar. Nótese que, al estar organizado **catalog['category_videos]'** por categorías, los videos de cada categoría están uno después del otro. Esto significa que, una vez el siguiente elemento sea de una categoría diferente, se habrán recorrido y sumado todos los videos de la categoría en cuestión. Así, una vez esto ocurre, el algoritmo recibe la instrucción **break** y deja de recorrer **catalog['category_videos']** aún si no ha recorrido todo el arreglo, pues ya recorrió el fragmento con los videos de la categoría de interés. Para finalizar, el algoritmo encuentra el video con el mayor número de apariciones de los pertenecientes a la categoría y lo devuelve, junto a su número de apariciones (días en tendencia) y otro diccionario auxiliar con la información de los canales de los videos de la categoría analizada.

El peor caso de este algoritmo es **O(n)** o, en notación tilda, **N**. Esto solo ocurre si la categoría de interés del usuario es la última de la lista (es decir, su ID es el menor, o mayor, dependiendo de la función de comparación empleada en Merge Sort). Nótese que el algoritmo nunca recorrerá el arreglo más de una vez y solo en un caso específico tendrá que recorrerlo todo. Es por esto que, en su peor caso, su complejidad temporal es **O(n)**.

Requerimiento 4

Para este requerimiento se organizó la información en el **catalog[‘like_videos’]**. Este catálogo se construye a partir de un Array List y tiene la información organizada primero por likes y después por países. La organización se da con el algoritmo de ordenamiento Merge Sort.

En este requerimiento se usa un catálogo descrito en el requerimiento 1 llamado **catalog[‘country_map’]**. Con la ayuda de este solo recorremos la fracción del país que nos pide el usuario una vez y de este modo en base a la cantidad de videos que quiere el usuario con el tag que él nos da, se recorre la lista y se le retorna la respuesta. Es por esto que este requerimiento tiene complejidad temporal **O(n)**.