OBSERVACIONES DEL RETO 1

Martín Rincón - Cod 201914114 - md.rincon@uniandes.edu.co
Mariana Ruiz - Cod 202011140 – m.ruizg@uniandes.edu.co

REQUERIMIENTO 1 – Trabajo en Equipo

Funciones utilizadas en este requerimiento:

cmpVideosByLikes(video1, video2) -> O(1)

sort_vids_by_likes(Data:list) -> O(nlogn)

filtrar_count_cat(videos, categories, categoria, pais) -> O(1+n+nlogn) = O(nlogn)

En filtrar_count_cat hay un while que se encarga de obtener el id de la categoría dado el nombre de esta; como la lista de categorías se asume constante siempre y tiene tan solo un par de decenas de elementos se asume que este recorrido toma un tiempo constante. Luego de identificar el id de la categoría hay que recorrer todos los videos para pegar en la lista vids_cat solo aquellos videos que cumplan con el requisito de país y categoría; este recorrido tiene complejidad temporal O(n).

Luego de haber filtrado los videos el algoritmo procede a ordenarlos usando el algoritmo Merge Sort, por ser este el más eficiente en tiempo. El algoritmo Merge Sort tiene complejidad temporal O(nlog) así que al aplicar la regla de la suma de la notación Big O tenemos:

filtrar_count_cat(videos, categories, categoria, pais) -> O(1+n+nlogn) = O(nlogn)

En cuanto a espacio, en el caso en que todos los videos cumplan el requerimiento de país y categoría, la nueva lista tendría el mismo tamaño que la original; además Merge Sort tiene complejidad espacial de O(n). La complejidad espacial de este algoritmo seria entonces O(n+n) = O(n).

REQUERIMIENTO 2 – Martín Rincón

Funciones utilizadas en este requerimiento:

max_vids_count(videos: list,pais:str)

Este algoritmo consta de 2 recorridos totales separados, uno se ocupa de filtrar los videos en la lista para obtener un diccionario de con los videos únicos filtrados. Las llaves del diccionario serán los títulos de los videos; su valor será una lista que tendrá los likes y dislikes para la fecha más actual, el número de apariciones del video, y el título del canal que subió el video. Esta etapa solo recorre la lista de videos una vez; el peor caso respecto a espacio se da si todos los videos son diferentes y corresponden al pais solicitado, en ese caso el diccionario tendra tantas parejas de elementos como la lista original. En cuanto a complejidad temporal, se trata de un unico recorrido total, asi que es O(n).

En su segunda etapa, el algoritmo recorre todas las entradas del diccionario y verifica que cumple con el requerimiento likes/dislikes. Si el elemento k tiene más apariciones registradas que el elemento k-p, entonces se registra la llave de ese elemento en la varaible respuesta. Al final del recorrido se retorna el valor asociado a la llave que ocupe el valor de la varaible respuesta. Este recorrido es total, en el peor de los casos se hacen n iteraciones.

Como ambas etapas tienen complejidad n, entonces el algoritmo tiene complejidad temporal O(2n)=O(n).

REQUERIMIENTO 3 – Mariana Ruiz

max_vids_cat(videos:list, categories:list, categoria:str)

Este algoritmo consta de 1 recorrido parcial por la lista de id de categorias y 2 recorridos totales, uno en la lista de todos los videos cargados y otro por el diccionario de videos de la categoría específica. El primer ciclo se encarga de encontrar el id de la categoría que el usuario desea consultar para después poder compararlo con los id de los videos en la lista completa. El pero caso se presenta cuando la categoria que se desea consultar aparece listada de última en la lista de categorías. Como se asume que esta lista tiene un tamaño constante y de pocas cifras, entonces se puede afirmar que la complejidad temporal es constante.

El segundo ciclo filtra los videos por su categoría para obtener un diccionario. Este diccionario tendrá como llaves los títulos de los videos y su valor será una lista que tendrá los likes y dislikes para la fecha más actual, el número de apariciones del video, y el título del canal que subió el video. El peor caso es en el que todos los videos pertenecen a la categoría solicitada, por lo que el diccionario tendrá tantas parejas de elementos como la lista original. En cuanto a complejidad temporal, se trata de un unico recorrido total, asi que es O(n).

En cuanto al tercer recorrido, este recorre todas las entradas del diccionario y verifica que el ratio likes/dislikes sea mayor a 20. Si el elemento k tiene más apariciones registradas que el elemento k-p, entonces se registra la llave de ese elemento en la varaible respuesta. Al final del recorrido se retorna el valor asociado a la llave que ocupe el valor de la varaible respuesta. Este recorrido es total y en el peor de los casos es de complejidad temporal O(n).

Como la primera etapa presenta una complejidad tenporal O(1) y los otros dos ciclos una complejidad temporal O(n), entonces el algoritmo tiene una complejidad temporal O(1+2n) = O(n).

REQUERIMIENTO 4 – Trabajo en Equipo

cmpVideosByComments(video1, video2) -> O(1)

sort_vids_by_comments(Data:list) -> O(nlogn)

filtrar_count_tag(videos, pais, tag) -> O(n+nlogn) = O(nlogn)

En filtrar_count_tag hay un ciclo que recorre la lista de todos los videos cargados y los filtra para encontrar aquellos que cumplen con el requisito de país y tag. Este recorrido tiene una complejidad

temporal O(n). Posteriormente utiliza la función sort_vids_by_comments que ordena la lista generada en el ciclo anterior por medio del algoritmo Merge Sort, el cual tiene un complejidad temporal O(nlogn). Así que al aplicar la regla de la suma de la notación Big O tenemos:

filtrar_count_tag(videos, pais, tag) -> O(n+nlogn) = O(nlogn)

En cuanto a espacio, en el caso en que todos los videos cumplan el requerimiento de país y tag, la nueva lista tendría el mismo tamaño que la original; además Merge Sort tiene complejidad espacial de O(n). La complejidad espacial de este algoritmo en el peor caso sería entonces O(n+n) = O(n).