Departamento de Ingeniería de Sistemas y Computación
Estructuras de Datos Y algoritmos
ISIS1225 - 202119

Documento de Análisis

Ana Sofía Castellanos Mosquera
202114167
a.castellanosm@uniandes.edu.co

Martín Santiago Galván Castro
201911013
ms.galvan@uniandes.edu.co

Análisis de complejidades:

Análisis General:

Para la creación de listas en el reto 1, siempre se utilizo como estructura de datos los arreglos. Se tomo esta decisión porque organizar y buscar información en esta estructura de datos tienen complejidades constantes. A diferencia de las listas encadenadas, en las cuales la función de getElement() tienen complejidades lineales.

En la realización de los laboratorios 4 y 5, se demostró que lo anteriormente dicho influye mucho en el rendimiento del código. Más específicamente, en los algoritmos de ordenamiento. En todos los casos, el uso de arreglo demostró ser considerablemente mas eficiente que la lista encadenada.

Requerimiento 1 (Equipo de trabajo):

El código desarrollado para lograr el primer requerimiento del reto se escribió teniendo en cuenta la siguiente estrategia:

1. Revisar en los videos registrados que videos cumplen con los requisitos de búsqueda del primer requerimiento (nombre de la categoría y país)
2. Generar una lista con dichos videos
3. Organizar la lista con algún algoritmo de organización usando como criterio
4. Generar una sublista del tamaño del numero de videos a listar
5. Imprimir la anterior lista.

Para lograr los pasos anteriores, se tuvieron que solucionar los problemas de como acceder a un video a partir de un nombre de categoría, sabiendo que los videos se guardan a con el id de su categoría. Para esto, se realizó una función que accede a la lista de categorías y toma el id del nombre. Con esto, se pasó a realizar el primer paso de la estrategia.

Al realizar el algoritmo que realiza el paso 1 y 2, es decir, que mete en una lista los videos que cumplen con las condiciones, se tuvo que recorrer elemento por elemento de la lista y añadirlo a otra. Esta parte del código tiene una complejidad Big O de O(n). Dado a que recorre toda la lista, pero solo realiza operaciones si el elemento, es decir el video, cumple con los requerimientos. Por lo que, en realidad, no siempre va a realizar n ciclos. Pero lo haría en el peor de los casos.

Luego, dado a que esta lista se tiene que organizar con algún algoritmo, se uso el algoritmo mergesort para realizar dicha tarea. Esto dado a que no hay muchos limites sobre el uso de la memoria en el programa, y es de las menores complejidades para algoritmos de organización. Este algoritmos es de complejidad Big O O(n log(n)).

Dado a que se puede escribir la complejidad del requerimiento 1 como:

$$O(n) + O(n \log(n))$$

Se pasa a describir la complejidad total del algoritmo a ser de:

$$O(n \log(n))$$

Dado a que a función linearitmica tiende a volverse mas grande que la función lineal y se delimita con el peor de los casos.

A pesar de que puede que la complejidad pueda parecer muy alta, el tiempo de ejecución aparenta ser relativamente corto. Esto se debe a que, el algoritmo de los primeros pasos depende de cuantos elementos de la lista cumplan con los criterios. Estos elementos tienden a ser considerablemente mas pequeños que los de la lista completa de videos. Lo que, a su vez, significa que el algoritmo Mergesort tiende a organizar una cantidad mas pequeña de datos que el total de datos ingresados por los archivos de Excel.

Requerimiento 2 (Martín Santiago Galván Castro):

En el caso del requerimiento 2, primero, al igual que en el requerimiento 1, se creo una lista solo con aquellos videos los cuales cumplen con los requisitos. Después de esto, se organiza haciendo uso del algoritmo merge sort. En este caso, se organizan según el elemento “trending_date”, que se refiere a cuando el video apareció en trending. La razón detrás del anterior procedimiento se explicará más adelante.

Después de organizar por la fecha en donde aparecen en trending, se empieza un doble for por cada elemento de la lista. Esto con el propósito de contar el video que más veces aparece en la lista. Para reconocer que un video es el mismo a otro, se compara el id del video. El algoritmo reporta cual es el video que mas veces aparece y cuantas veces aparece en la lista. Adicionalmente, el criterio de comparación para que se supere al máximo en el algoritmo es no incluyente. Lo anterior y la organización por fecha de trending, permite tomar el video en su fecha de trending más reciente.

La complejidad del algoritmo es alta. Se demora alrededor de 10 minutos o menos en la maquina del estudiante encargado del requerimiento 2 (Martín Galván). Dado a esto, se incluyo en el algoritmo un porcentaje de carga para mostrar como va este.

Se puede representar la complejidad del algoritmo comprendiendo la complejidad de cada parte del algoritmo. Primero, la creación de la lista con los videos que cumplen el requerimiento tiene una complejidad de O(n). Luego, el algoritmo de merge sort tiene complejidad de O(n log(n)) y el algoritmo que cuenta la moda tiene complejidad de O(n^2). Se puede entender como:

$$O(n) + O(n \log(n)) + O(n^2)$$

Por lo anterior, dado a que la máxima complejidad es la de O(n^2), la complejidad del algoritmo para resolver el requerimiento 2 es de:

$$O(n^2)$$

Requerimiento 3 (Ana Sofia Castellanos Mosquera):

El código desarrollado para lograr el tercer requerimiento del reto se escribió teniendo en cuenta la siguiente estrategia:

1. Se creo una lista que contuviera solo aquellos videos que cumplieran con la categoría dada por parámetro y que se percibieran de forma positiva, es decir con un ratio likes / dislikes mayor a 20.
2. Se creo una lista nueva para ir almacenado cada video (de forma única) con la cantidad de días que fue tendencia
3. Se realizó un ciclo doble para revisar cada video y crear un contador para almacenar la cantidad de días que fue tendencia. Para revisar estos días y asegurarse de que no se repitieran para cada nombre de video se creó una lista que almacenaba las fechas y se suma al contador si por un lado el nombre coincide con el nombre del video que se está almacenando y si por otro lado la fecha de dicho video no se encuentra ya dentro de la lista de fechas.
4. Se devuelve la lista que cuenta con los videos y los días que fue tendencia.

Este algoritmo tiene un orden de crecimiento temporal del tipo O(N^2) que resulta de analizar el algoritmo. En primer lugar, el algoritmo realiza un ciclo con todos los videos del catálogo por lo cual cuenta con una ecuación Big O del tipo O(N) puesto que debe ingresar la cantidad de datos al ciclo.

Tras ello se realiza un ciclo doble para revisar los videos, crear el contador de días y añadir cada video único en la lista resultado, lo cual cuenta con un Big O de O(N^2) ya que ingresa al algoritmo en un primer ciclo para revisar el primer elemento, luego el segundo etc hasta n veces y luego a partir de la posición en la que se encuentra entra a otro ciclo n-1, luego n-2 etc hasta llegar a recorrer solo 1 elemento de lo cual queda que recorre N * N, es decir tiene una complejidad de N^2.

Luego ordena por días con Merge Sort el cual tiene un Big O de O(N log N) para poner los que tienen más días en la parte superior.

A partir de allí queda una ecuación del tipo:

$$\text{Complejidad} = O(N) + O(N^2) + O(N \log N)$$

Debido a que se considera el peor caso y la máxima complejidad, el orden de crecimiento temporal es de:

$$O(N^2)$$

Requerimiento 4 (Equipo de trabajo):

El código desarrollado para lograr el cuarto requerimiento fue muy similar al requerimiento 1. Esto se debe a que los dos requerimientos tienen mucho en común. Las diferencias es que en el requerimiento 4 se buscan videos por una etiqueta en vez que por categoría y por el número de comentarios en vez del número de likes. Gracias a esto, no fue necesario la generación de una función que devuelve el id de una categoría a partir del nombre.

La estrategia llevada a cabo para el requerimiento fue la siguiente:

1. Revisar en los videos registrados que videos cumplen con los requisitos de búsqueda del primer requerimiento (Etiqueta del video y país)
2. Generar una lista con dichos videos
3. Organizar la lista con algún algoritmo de organización usando como criterio el número de comentarios
4. Generar una sublista del tamaño del número de videos a listar
5. Imprimir la anterior lista.

De la misma manera que en el requerimiento 1, revisar la lista de videos por aquellos videos que cumplan los criterios tiene una complejidad de O(n). Posteriormente, ordenar la lista de elementos que cumplen los requisitos tiene una complejidad de O(n log(n)). Sin embargo, a diferencia del requerimiento 1, puede que se tengan que organizar muchos mas elementos en la nueva lista. Esto se debe a que los videos pueden tener mas de un tag, causando la posibilidad de que muchos videos compartan el mismo tag.

Para reportar la complejidad del algoritmo realizado para el requerimiento 4, se tienen compejidades de O(n) y de O(n log(n)):

$$O(n) + O(n \log(n))$$

Por lo que la complejidad del algoritmo es de:

$$O(n \log(n))$$