Documento análisis reto 1

Grupo 2 sección 4 ISIS1225.

Gregorio Salazar 202022085.

Valentina Uribe 201817485.

Requerimiento 2 por Valentina Uribe y requerimiento 3 por Gregorio Salazar.

Requerimiento 1.

```
def get_most_view_videos(catalog,country_name,category_name):
    category_id=get_category_id(catalog,category_name)
    videos_country_category=lt.newList('ARRAY_LIST')
    countries=catalog['countries']
    poscountry = lt.isPresent(countries,country_name)
    if category_id==0 or poscountry==0:
        return None
    else:
        country = lt.getElement(countries, poscountry)
        for j in range(1,lt.size(country['videos'])+1):
            video=lt.getElement(country['videos'],j)
            if video['category_id']==category_id:
                 lt.addLast(videos_country_category,video)
        return merge.sort(videos_country_category,cmp_videos_by_views)
```

La función get_category_id recorre la lista del archivo de categorías y encuentra el id correspondiente al nombre gasta un tiempo fijo por lo que diremos que es O(1). Luego se construye una nueva lista videos_country_category que contiene los videos que son del país correspondiente y también de la categoría especificada. Para esto se recorre la lista de videos del país y se agregan los videos de la categoría, esto representa solo un recorrido por lo que la complejidad es O(n). Finalmente se hace un ordenamiento merge sobre la lista nueva, por lo que en el peor de los casos este ordenamiento tiene una complejidad de O(nlogn). Por lo que finalmente la complejidad de este requerimiento es en el peor de los casos O(nlogn).

Requerimiento 2.

```
def get_most_time_trending_country(catalog,country_name):
    countries=catalog['countries']
    poscountry = lt.isPresent(countries,country_name)
    country = lt.getElement(countries, poscountry)
    country_videos=country['videos']
    trending_counter=lt.newList('ARRAY_LIST', cmpfunction=compare_videos_by_id)
    size=lt.size(country_videos)
    for i in range(1,size+1):
        video=lt.getElement(country_videos,i)
        posvideo=lt.isPresent(trending_counter,video['video_id'])
        if posvideo==0:
            video_trending={'video_id':video['video_id'],'title':video['title'],'
counter':1,'channel_title':video['channel_title'],'country':country_name}
            lt.addLast(trending_counter,video_trending)
        else:
            video_trending=lt.getElement(trending_counter,posvideo)
            video_trending['counter']+=1
    x=0
    more_trending=None
    for j in range(1,lt.size(trending_counter)+1):
        video=lt.getElement(trending_counter,j)
        if video['counter']>x:
            x=video['counter']
            more_trending=video
    return more_trending
```

Para contar la cantidad de días que un video fue trending en un país, se crea una lista nueva en donde se agregan los videos con un nuevo parámetro llamado “counter” que cuenta la cantidad de veces que aparece el video en el país. Para esto se recorren los videos del país y se crea una nueva lista llamada trending_counter que es la descrita anteriormente. Durante el ciclose aplica la función isPresent para revisar si el video ya está en trending_counter y como esta función recorre la lista, se hace un doble recorrido, por lo que esta parte del código tiene complejidad O(n^2). Luego solo se busca el video con mayor “counter”. Esto tarda O(n), por lo que finalmente el requerimiento tiene complejidad O(n^2).

Requerimiento 3.

```
def get_most_time_trending_category(catalog,category_name):
```

```
    category_id=get_category_id(catalog,category_name)
    categories=catalog['categories']
    poscategory = lt.isPresent(categories,category_id)
    category = lt.getElement(categories, poscategory)
    category_videos=category['videos']
    trending_counter=lt.newList('ARRAY_LIST', cmpfunction=compare_videos_by_title
)
    size=lt.size(category_videos)
    for i in range(1,size+1):
        video=lt.getElement(category_videos,i)
        posvideo=lt.isPresent(trending_counter,video['title'])
        if posvideo==0:
            video_trending={'video_id':video['video_id'],'title':video['title'],'
counter':1,'channel_title':video['channel_title'],'category_id':category_id}
            lt.addLast(trending_counter,video_trending)
        else:
            video_trending=lt.getElement(trending_counter,posvideo)
            video_trending['counter']+=1
    x=0
    more_trending=None
    for j in range(1,lt.size(trending_counter)+1):
        video=lt.getElement(trending_counter,j)
        if video['counter']>x:
            x=video['counter']
            more_trending=video
    return more_trending
```

Este requerimiento es exactamente igual al 2, solo que en vez de recorrer el país se recorre la categoría. Por lo mismo dicho anteriormente el requerimiento tiene complejidad O(n^2) en el peor caso.

Requerimiento 4.

```
def get_most_likes_tag(catalog,tag,country_name):
    videos=catalog['videos']
    size=lt.size(videos)
    tag_country_videos=lt.newList('ARRAY_LIST')
    for i in range(1,size+1):
        video=lt.getElement(videos,i)
        video_tags=video['tags']
        if tag in video_tags and video['country']==country_name:
            lt.addLast(tag_country_videos,video)
    return merge.sort(tag_country_videos,cmp_videos_by_likes)
```

Para este requerimiento se recorren todos los videos y se escogen los que contengan el tag especifiado como subcadena de sus tags y que sean del país específicado. Esto solo tiene complejidad O(n) ya que se

hace un recorrido simple. Finalmente se hace un ordenamiento merge que tiene complejidad O(nlogn). El requerimiento en el peor de los casos tendrá entonces complejidad O(nlogn).