DOCUMENTO DE ANALISIS

Estudiante A: Santiago Ballén Cod 202023544

Estudiante B: Paola Campiño Cod 202020785

Este documento pretende analizar la complejidad de cada uno de los requerimientos propuestos para el reto 1

Requerimiento 1: Consultar n número de videos más vistos por país y en una categoría especifica

La primera función que se usó para hacer posible la busque de videos fue la función `getvideobycountry(catalog, pais)` la cual fue creada para filtrar todos los videos por el país suministrado por usuario.

El orden de complejidad de esta parte del código es de O(N) ya se está haciendo un recorrido por todos los datos que le hemos suministrado al sistema. Hasta este momento el código lleva un orden de complejidad de O(N). A continuación, se puede ver el codigo para función getVideobycountry(catalog, pais):

```
def getvideobycountry(catalog, pais):
    Country_list = lt.newList()
    Country_list["key"]="ARRAY_LIST"
    n=0
    while n < lt.size(catalog):
          video=lt.getElement(catalog,n)
          if video["country"]==pais:
              lt.addLast(Country_list, video)
          n+=1
    return Country_list
```

Más adelante se hace uso de la función GetVideosbycategoria (catalog, categoria) para ordenar los videos por categorías como se muestra a continuación:

```
def GetVideosbycategoria(catalog, categoria):
    vid_categ = lt.newList()
    n=0
```

```
    while n < lt.size(catalog):
          video=lt.getElement(catalog,n)
          if video["category_id"]==categoria:
              lt.addLast(vid_categ, video)
          n+=1
    return vid_categ
```

Esta función tiene un orden de complejidad de O(N), ya que recorre cada uno de los datos. Hasta este punto el orden de complejidad total es de O(2N)

Después de filtrar todos los videos por categoría, se ordenó todos los datos en base a las vistas con la función: `sortVideos_byViews(catalog, size)` que se puede visualizar a continuación:

```
def sortVideos_byViews(catalog, size):
    sub_list = lt.subList(catalog,1, size)
    sub_list = sub_list.copy()
    sorted_list=merg.sort(sub_list, cmpVideosByViews)
    return  sorted_list
```

Esta función tiene un orden complejidad de O (N log N), dando así un orden de complejidad total de O (2N+ N(logN)) hasta este punto.

Después de tener las funciones listas para imprimir se va a hacer uno de la función imprime_toda_lista_econtrada_req1(catalog) que va a mirar cada uno de los componentes de la última lista resultado de la función sortVideos_byViews y los va a imprimir de manera organizada en la consola como se puede ver a continuación:

```
def imprime_toda_lista_econtrada_req1(catalog):
     n=0
     while n<lt.size(video_ordenados_por_vistas):
           video_ordenado=lt.getElement(catalog,n)
           print("Posición: "+str(1+n))
           print("-"+"Trending_date: "+video_ordenado["trending_date"])
           print("-"+"Title: "+video_ordenado["title"])
           print("-"+"Chanel_title: "+video_ordenado["channel_title"])
           print("-"+"Publish_time: "+video_ordenado["publish_time"])
           print("-"+"Views: "+video_ordenado["views"])
           print("-"+"Likes: "+video_ordenado["likes"])
```

```
            print("-"+"Dislikes: "+video_ordenado["dislikes"])
            print(separador ())
            n+=1
```

La complejidad de esta función es de O(N) ya que se está recorriendo todos los elementos de la lista.

Entonces ya finalizadas cada una de las funciones en la complejidad total es de:

O(3N + N(logN))

Requerimiento 2: Encontrar video tendencia por país (Estudiante_A)

A la primera función que llama este requerimiento es `GetVideosbyCountry(catalog, pais)`; esta función se encarga de filtrar todos los videos del país es escoja en usuario en una nueva lista. La complejidad de este algoritmo es de O(N) ya que solo se realiza una iteración a lo largo de todo el catálogo.

```
def GetVideosbyCountry(catalog, pais):
    vid_country = lt.newList()
    n=0
    while n < lt.size(catalog):
          video=lt.getElement(catalog,n)
          if video["country"]==pais:
              lt.addLast(vid_categ, video)
          n+=1
    return vid_country
```

La segunda función a la que llama el requerimiento es: `Get_trending_pais(catalog)`; al igual que la función anterior, tiene una complejidad O(N) ya que solo realiza una iteración.

```
def Get_trending_pais(catalog):
    trending_dates= lt.newList()
    lista_prohibido=[]
    n=0
    while n < lt.size(catalog):
          video=lt.getElement(catalog,n)
          ID=video["video_id"]
```

```
            if ID not in lista_prohibido:
                dato={"ID": ID,"title":video["title"], "Channel title":
video["channel_title"], "country":video["country"],"dias":1}
                lt.addLast(trending_dates,dato)
                lista_prohibido.append(ID)
            else:
                 posicion=lista_prohibido.index(ID)
                 elemento_cambiar=lt.getElement(trending_dates,posicion)
                 elemento_cambiar["dias"]+=1
            n+=1


    return trending_dates
```

Después de lo anterior, se usó de la función sortVideos_byDias(catalog). Esta función nos permitió crear una lista nueva en la que todos los componentes estarían ordenados de mayor a menor. La función tiene una complejidad de O(Nlog(N)) ya que usamos un Merge Sort para hacer que la organización fuera más rápida. A continuación, se puede ver la función `sortVideos_byDias(catalog):`

```
def sortVideos_byDias(catalog):
    sub_list = lt.subList(catalog,1, 1)
    sub_list = sub_list.copy()
    sorted_list=merg.sort(sub_list, cmpVideosBydias)
    return  sorted_list
```

Hasta este punto el requerimiento tiene una complejidad de O(2N + NlogN).

Al momento de presentar la información se hiso uso de la función print_req3(catalog)que tiene una complejidad de O(K).

```
def print_req2(catalog):
    vid=lt.getElement(catalog,1)
    print("-"+"Title: "+vid["title"])
    print("-"+"Channel title: "+vid["Channel title"])
    print("-"+"Country: "+vid["country"])
    print("-"+"Días: "+str(vid["dias"]))
    print(separador())
```

Finalmente, la complejidad total del requerimiento es de:

O(2N +N log N)

Requerimiento 3: Encontrar video tendencia por categoría (Estudiante_B)

La primera parte del requerimiento 3 implica hacer uso de la función GetVideosbycategoria

(categoria, category_name), esta función filtra el catálogo de estrada de manera que al final se obtenga una nueva lista con los videos de una categoría especifica. La función tiene una complejidad de O(N) como se muestra a continuación:

```
def GetVideosbycategoria(catalog, categoria):
    vid_categ = lt.newList()
    n=0
    while n < lt.size(catalog):
          video=lt.getElement(catalog,n)
          if video["category_id"]==categoria:
              lt.addLast(vid_categ, video)
          n+=1
    return vid_categ
```

Esta función tiene la complejidad mencionada ya que se está haciendo un recorrido a cada uno de los elementos en la lista. Hasta este punto el requerimiento tiene una complejidad de O(N).

Después de haber filtrado los datos, se hace uso de la función Get_rending_categoria(catalog), la cual permite sumar la cantidad de veces que el video aparece en la lista a analizar. La función tiene una complejidad de O(N) ya que se está haciendo un recorrido por toda la lista. Hasta este punto la complejidad de todo el requerimiento es de O(2N)

```
def Get_rending_categoria(catalog):
    trending_dates= lt.newList()
    lista_prohibido=[]
    n=0
    while n < lt.size(catalog):
          video=lt.getElement(catalog,n)
          ID=video["video_id"]
```

```
            if ID not in lista_prohibido:
                dato={"ID": ID,"title":video["title"], "Channel title":
video["channel_title"], "country":video["country"],"dias":1}
                lt.addLast(trending_dates,dato)
                lista_prohibido.append(ID)
            else:
                 posicion=lista_prohibido.index(ID)
                 elemento_cambiar=lt.getElement(trending_dates,posicion)
                 elemento_cambiar["dias"]+=1


            n+=1
```

Después de hacer el conteo de las veces que aparece un video en la lista, fue fundamental hacer uso de la función sortVideos_byDias(catalog). Esta función nos permitió crear una lista nueva en la que todos los componentes estarían ordenados de mayor a menor. La función tiene una complejidad de O(Nlg N) ya que usamos un merge sort para hacer que la organización fuera más rapida. A continuación, se puede ver la función sortVideos_byDias(catalog):

```
def sortVideos_byDias(catalog):
    sub_list = lt.subList(catalog,1, 1)
    sub_list = sub_list.copy()
    sorted_list=merg.sort(sub_list, cmpVideosBydias)
    return  sorted_list
```

Hasta este punto el requerimiento tiene una complejidad de O(2N + NlogN).

Al momento de presentar la información se hiso uso de la función print_req3(catalog)que tiene una complejidad de O(K).

```
def print_req3(catalog):
    vid=lt.getElement(catalog,1)
    print("-"+"Title: "+vid["title"])
    print("-"+"Channel title: "+vid["Channel title"])
    print("-"+"Category ID "+vid["ID"])
    print("-"+"Días: "+str(vid["dias"]))
    print(separador ())
```

La complejidad total del requerimiento es de:

O(2N +N log N)

Requerimiento 4: Consultar los n videos con más likes por un tag de acuerdo a un país especifico

La primera función que se usó para hacer posible la busque de videos fue la función `getvideobycountry(catalog, pais)` la cual fue creada para filtrar todos los videos por el país suministrado por usuario en código la función se visualiza de la siguiente forma:

```
def getvideobycountry(catalog, pais):
    Country_list = lt.newList()
    Country_list["key"]="ARRAY_LIST"
    n=0
    while n < lt.size(catalog):
          video=lt.getElement(catalog,n)
          if video["country"]==pais:
              lt.addLast(Country_list, video)
          n+=1
    return Country_list
```

El orden de complejidad de esta parte del código es de O(N) ya se está haciendo un recorrido por todos los datos que le hemos suministrado al sistema. El orden total hasta este punto es de O(N)

Más adelante se hace uso de la función `video_por_etiqueta(catalog, label)` para ordenar por tag como se muestra a continuación:

```
def video_por_etiqueta(catalog, label):
    vid_label = lt.newList()
    n=0
    while n < lt.size(catalog):
          video=lt.getElement(catalog,n)
          if label in video["tags"]:
              lt.addLast(vid_label, video)
          n+=1
    return vid_label
```

Esta función también tiene un orden de complejidad de O(N), ya que recorre cada uno de los datos.

Hasta este punto el orden de complejidad total es de O(2N)

se ordenó todos los datos en base a los likes con la función: videos_por_likes(catalog,size) la cual tiene un orden de complejidad de O (N log N), como se muestra a continuación:

```
def videos_por_likes(catalog,size):
    sub_list = lt.subList(catalog,1, size)
    sub_list = sub_list.copy()
    sorted_list=merg.sort(sub_list, cmpVideosByLikes)
    return  sorted_lis
```

Hasta este punto la complejidad total es de O (2N+ N(logN)).

Y para finalizar hay que mencionar la función `print_req4(catalog)` la cual imprime todos los elementos encontrados la función es la siguiente:

```
def print_req4(catalog):
    n=0
    while lt.size(video_por_likes)>n:
            vid=lt.getElement(video_por_likes,n)
            print("Video "+str(n+1))
            print("-"+"Title: "+vid["title"])
            print("-"+"Chanel_title: "+vid["channel_title"])
            print("-"+"Publish_time: "+vid["publish_time"])
            print("-"+"Likes: "+vid["likes"])
            print("-"+"Dislikes: "+vid["dislikes"])
            print("-"+"Tags: "+vid["tags"])
            print(separador ())
            n+=1
```

Esta función tiene una complejidad de O(N) haciendo que la complejidad total de este requerimiento sea de:

O (3N + NlogN).