**Aanalísis complejidad temporal reto 2**

Grupo 2 sección 4 ISIS1225.

Requerimiento 2: Gregorio Salazar 202022085

Requerimiento 3: Valentina Uribe 201817485

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']

    exist\_country=mp.contains(countries,country\_name)

    if exist\_country:

        entry=mp.get(countries,country\_name)

        country=me.getValue(entry)

    videos=lt.iterator(country['videos'])

    for video in videos:

          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 solo recorre la categorías y dado el nombre retorna el id, tiene una complejidad O(1) ya que es constante y corta. Luego se extraen los videos del país usando maps, que tiene una complejidad de O(1) gracias a la eficiencia de los mapas, y se recorre está lista de videos, que tiene complejidad O(n). Luego se ordena la lista usando merge. Finalmente, esta función tiene una complejidad de O(nlogn) por el merge.

Requerimiento 2:

def get\_most\_time\_trending\_country(catalog,country\_name):

    countries=catalog['countries']

    exist\_country=mp.contains(countries,country\_name)

    if exist\_country:

        entry=mp.get(countries,country\_name)

        country=me.getValue(entry)

    country\_videos=country['videos']

    trending\_counter=mp.newMap(maptype='Probing', loadfactor=0.5)

    size=lt.size(country\_videos)

    for i in range(1,size+1):

        video=lt.getElement(country\_videos,i)

        exist\_video=mp.contains(trending\_counter,video['video\_id'])

        if not exist\_video:

            video\_trending={'video\_id':video['video\_id'],'title':video['title'],'counter':1,'channel\_title':video['channel\_title'],'country':country\_name}

            mp.put(trending\_counter,video['video\_id'],video\_trending)

        else:

            video\_trending=me.getValue(mp.get(trending\_counter,video['video\_id']))

            video\_trending['counter']+=1

    videos=mp.valueSet(trending\_counter)

    videos=lt.iterator(videos)

    x=0

    more\_trending=None

    for video in videos:

        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 un map nuevo 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 un nuevo map llamado trending\_counter que es el descrito anteriormente. Por las operaciones del map este ciclo es un ciclo simple y la complejidad hasta ahora es tan solo O(n). Luego solo se busca el video con mayor “counter” recorriendo la lista de valores del map trending\_counter. Esto también tiene complejidad O(n), por lo que toda la función tiene una complejidad de tan solo O(n).

Requerimiento 3.

def get\_most\_time\_trending\_category(catalog,category\_name):

    category\_id=get\_category\_id(catalog,category\_name)

    categories=catalog['categories']

    exist\_category=mp.contains(categories,category\_id)

    if exist\_category:

        entry=mp.get(categories,category\_id)

        category=me.getValue(entry)

    category\_videos=category['videos']

    trending\_counter=mp.newMap(maptype='Probing', loadfactor=0.5)

    size=lt.size(category\_videos)

    for i in range(1,size+1):

        video=lt.getElement(category\_videos,i)

        exist\_video=mp.contains(trending\_counter,video['video\_id'])

        if not exist\_video:

            video\_trending={'video\_id':video['video\_id'],'title':video['title'],'counter':1,'channel\_title':video['channel\_title'],'category\_id':category\_id}

            mp.put(trending\_counter,video['video\_id'],video\_trending)

        else:

            video\_trending=me.getValue(mp.get(trending\_counter,video['video\_id']))

            video\_trending['counter']+=1

    videos=mp.valueSet(trending\_counter)

    videos=lt.iterator(videos)

    x=0

    more\_trending=None

    for video in videos:

        if video['counter']>x:

            x=video['counter']

            more\_trending=video

    return more\_trending

Estes requerimiento es exactamente igual al requerimiento 2 pero se recorren las categorías en vez de los países. Su complejidad es O(n).

Requerimiento 4.

def get\_most\_likes\_tag(catalog,tag,country\_name):

    countries=catalog['countries']

    entry=mp.get(countries,country\_name)

    country=me.getValue(entry)

    country\_videos=country['videos']

    videos=lt.iterator(country\_videos)

    tag\_country\_videos=lt.newList('ARRAY\_LIST')

    for video in videos:

        video\_tags=video['tags']

        if tag in video\_tags:

            lt.addLast(tag\_country\_videos,video)

    return merge.sort(tag\_country\_videos,cmp\_videos\_by\_likes)

Para este requerimiento se recorren todos los videos de un país dado y se escogen los que contengan el tag especifiado como subcadena de sus tags. Esto solo tiene complejidad O(n) ya que es un recorrido simple, y extraer el país con maps es O(1). Luego se ordenan los videos con merge. La complejidad de la función es finalmente O(nlogn).