RETO 2: Análisis de Complejidad

Cristian Armando Sánchez Ocampo ca.sanchezo1@uniandes.edu.co 202022112
Luis Felipe Dussán R. lf.dussan@uniandes.edu.co 201912308

Requerimiento 1:

```
def filtrar_PaisCategoria(mapCategory,category,country,size):
    categoria = mp.get(mapCategory,category)
    valueCat = me.getValue(categoria)
    lst_videos = valueCat['videos']
    new_list = lt.newList(datastructure='ARRAY_LIST')
    for video in lt.iterator(lst_videos):
        if video['country']==country:
                alt.addLast(new_list,video)
    sorted_list= sortVideos(new_list,cmpVideosByViews)
    lst_n = lt.subList(sorted_list,1,size)
    return lst_n
```

Complejidad: O(n+nlogn)
Tiempo: 395.70599999999104 ms
Memoria: 19.1640625 KB
Comparación con Reto 1: No hubo una diferencia notable entre ambas implementaciones, tanto en tiempo como en complejidad. En memoria sí pudo notar una mejora, se utilizó menos espacio.

Requerimiento 2: Luis Felipe Dussán Rueda.

```
'Requerimiento #2'

def filtrarPaisTendencia(catalog, country):
    country_s = mp.get(catalog['country'], country)
    if country_s:
        sub_list= me.getValue(country_s)
    sv= sortVideos(sub_list,cmpVideosbyId)
    lst = extraer_ids(sv)
    id,m = id_mas_repetido2(sv)
    return (video_por_id(sv,id,m))
```

Complejidad: O(n+nlogn+n^2)

Tiempo: 766069.5097 ms

Memoria: 36.6953125 KB
Comparación con Reto 1: No hubo una diferencia notable entre ambas implementaciones, tanto en tiempo como en complejidad. En memoria sí pudo notar una mejora, se utilizó menos espacio.

Requerimiento 3: Cristian Armando Sanchéz Ocampo.

```
def video_mas_dias_tendencia(mapCategory,category):
    categoria = mp.get(mapCategory,category)
    valueCat = me.getValue(categoria)
    lst_videos = valueCat['videos']
    new_list = lt.newList(datastructure='ARRAY_LIST')
    for video in lt.iterator(lst_videos):
        if (video['video_id']!= '#NAME?'):
            lt.addLast(new_list,video)
    videosByFreq = extraer_ids(new_list)
    videosSorted = sortVideos(videosByFreq,cmpVideosByFreq)
    return alt.firstElement(videosSorted)
```

Complejidad: O(n+nlogn+n^2)
Tiempo: 298031.9207 ms
Memoria: 35.7734375 KB
Comparación con Reto 1: No hubo una diferencia notable entre ambas implementaciones, tanto en tiempo como en complejidad. En memoria sí pudo notar una mejora, se utilizó menos espacio.

Requerimiento 4:

```
def videos_mas_likes(mapCountry,country,size,tag):
    pais = mp.get(mapCountry,country)
    videos = me.getValue(pais)
    new_list = lt.newList(datastructure='ARRAY_LIST')
    for video in lt.iterator(videos):
        if tag in video['tags']:
            lt.addLast(new_list,video)
    srt_lst = sortVideos(new_list,cmpVideosbyLikes)
    return lt.subList(srt_lst,1,size)
```

Complejidad: O(n+nlogn)
Tiempo: 496.53819999998086 ms
Memoria: 3.3984375 KB
Comparación con Reto 1: No hubo una diferencia notable entre ambas implementaciones, tanto en tiempo como en complejidad. En memoria sí pudo notar una mejora, se utilizó menos espacio.