Entrega reto #1

Req. 2 - David Leonardo Almanza Márquez, d.almanza@uniandes.edu.co, 202011293

Req. 3 - Laura Daniela Arias Flórez, l.ariasf@uniandes.edu.co, 202020621

Análisis de complejidad del requerimiento #1

```
def getBestViews(catalog, category_id, country):
    listcopy = catalog["videos"].copy()
    sorted_list = sortVideoCountryCategory(listcopy)
    #uso de merge: nlog(n)
    posvideo = lt.binarySearch(sorted_list, country, "country")
    #Busqueda binaria -> log(n)
    first = False
    while posvideo > 1 and not first:
        if lt.getElement(sorted_list, posvideo-1)['country'] == country:
            posvideo -= 1
        else:
            first = True
    #es para ubicar el primer elemento de la lista de pais en especifico, hay 10 paises -> maximo n/10

    sub_list = lt.newList('ARRAY_LIST')
    while lt.getElement(sorted_list, posvideo)['country'] == country:
        if lt.getElement(sorted_list, posvideo)['category_id'] == category_id:
            lt.addLast(sub_list, lt.getElement(sorted_list, posvideo))
        elif not lt.isEmpty(sub_list):
            return sub_list
        posvideo += 1
    #recorre unicamente mientras no cambiemos de pais -> n/10

    return sub_list
```

Análisis de complejidad del requerimiento #2

```
def getTrendCountry(catalog, country):
    listcopy = catalog["videos"].copy()_
    sorted_list = sortVideoByCountry(listcopy)
    #uso de merge: nlog(n)
    posvideo = lt.binarySearch(sorted_list, country, "country")
    #busqueda binaria: log(n)
    first = False
    while posvideo >= 1 and not first:
        if lt.getElement(sorted_list, posvideo)["country"] == lt.getElement(sorted_list, posvideo-1)["country"]:
            posvideo -=1
        else:
            first = True
    #es para ubicar el primer elemento de la lista de pais en especifico, hay 10 paises -> maximo n/10
    posicion = posvideo
    conteo = 1
    Last = False
    while not Last and posicion < lt.size(sorted_list):
        if lt.getElement(sorted_list, posicion)["country"] == lt.getElement(sorted_list, posicion+1)["country"]:
            conteo += 1
            posicion +=1
        else:
            Last = True
    #cuenta videos de un pais -> n/10
    CountryList = lt.subList(sorted_list, posvideo, conteo)
    #n/10
    SortedCountryList = sortVideoById(CountryList)
    #n/10
    histograma = {}
    i = 0
    while i < lt.size(SortedCountryList):
        Url = lt.getElement(SortedCountryList, i)["video_id"]
        histograma[Url] = histograma.get(Url, 0) +1
        i +=1
    #n/10
    mayor = max(histograma.values())
    for Url in histograma:
        if(histograma[Url] == mayor):
            UrlVideoTrend = Url
            break
    #n/10
```

Análisis de complejidad del requerimiento #3

```
def getTrendCategory(catalog, category_id):
    listcopy = catalog["videos"].copy()
    sorted_list = sortVideoCategoryTitle(listcopy)
    #uso de merge: O(nlog(n)) por merge (3 merge usados)
    posvideo = lt.binarySearch(sorted_list, category_id, "category_id")
    #busqueda binaria: O(log(n))
    first = False
    while posvideo > 1 and not first:
        if lt.getElement(sorted_list, posvideo-1)['category_id'] == category_id:
            posvideo -= 1
        else:
            first = True
    #en el peor de los casos recorre O(m) donde m es igual al número total de videos que hay en esa categoría

    postrend = posvideo
    trendtitle = ""
    trendcount = -1
    count = 0
    while lt.getElement(sorted_list, posvideo)['category_id'] == category_id:
        if posvideo == 1:
            trendtitle = lt.getElement(sorted_list, postrend)['title']
        elif lt.getElement(sorted_list, posvideo)['title'] != lt.getElement(sorted_list, posvideo-1)['title']:
            if count > trendcount:
                trendcount = count
                postrend = posvideo-1
                trendtitle = lt.getElement(sorted_list, postrend)['title']
            count = 1
        elif (lt.getElement(sorted_list, posvideo)['trending_date'] != lt.getElement(sorted_list, posvideo-1)['trending_date']):
            count += 1
        posvideo += 1
    #O(m)
```

```
    video = lt.getElement(sorted_list, postrend)
    sorted_list.clear()
    return video, trendcount
```

Análisis de complejidad del requerimiento #4

```
def getBestTag(catalog, tagname, country):
    listcopy = catalog["videos"].copy()
    sorted_list = sortVideoByCountry(listcopy)
    #uso de merge: O(nlog(n))
    posvideo = lt.binarySearch(sorted_list, country, "country")
    #busqueda binaria: O(log(n))
    first = False
    while posvideo >= 1 and not first:
        if lt.getElement(sorted_list, posvideo)["country"] == lt.getElement(sorted_list, posvideo-1)["country"]:
            posvideo -=1
        else:
            first = True
    #en el peor de los casos recorre O(m) donde m es igual al número total de videos que hay para ese país

    pos = posvideo
    count = 1
    last = False

    while not last and pos < lt.size(sorted_list):
        if lt.getElement(sorted_list, pos)["country"] == lt.getElement(sorted_list, pos+1)["country"]:
            count += 1
            pos +=1
        else:
            last = True
    #O(m)

    CountryList = lt.subList(sorted_list, posvideo, count)

    taglist = lt.newList('ARRAY_LIST', cmpfunction=comparetags)
```

```
    for n in range(1, lt.size(CountryList)+1):
        video = lt.getElement(CountryList, n)
        videotags = video['tags'].split('"|"')
        for videotag in videotags:
            addVideoTags(taglist, videotag.strip('"'), video)
            #O(1) donde l es el numero de tags del video
    #O(m)

    tagpos = lt.isPresent(taglist, tagname)

    if tagpos > 0:
        videos = lt.getElement(taglist, tagpos)['videos']
        return sortVideoByViews(videos)
    else:
        return None


def addVideoTags(taglist, tagname, video):
    postag = lt.isPresent(taglist, tagname)
    if postag > 0:
        tag = lt.getElement(taglist, postag)
    else:
        tag = newTag(tagname)
        lt.addLast(taglist, tag)
    lt.addLast(tag['videos'], video)
```