Integrantes:


```
Esteban Leiva - 202021368 - e.leivam@uniandes.edu.co
Michelle Vargas - 201914771 - bm.vargas@uniandes.edu.co
```


Análisis de complejidad:

REQ 1:

```
def R1(categoria,pais,num,catalog):
    ID=model.categoriaporID(categoria,catalog) # 1
    if ID==None:
        return 'Categoría no válida'
    else:
        l=model.lporcyp(ID,pais,catalog['videos']) # 2
        if l==None:
            return 'País no válido.'
        else:
            l2=model.sortVideos(l,lt.size(l),model.cmpVideosbyViews)[1] # 3
            if num>lt.size(l2):
                return 'El número ingresado excede la cantidad de videos que cump
len con los requisitos. Intente con un número igual o menor a '+str(lt.size(l))
            else:
                n=0
                c=''
                final=lt.subList(l2,1,num)
                i=it.newIterator(final)
                while it.hasNext(i): # O(num)
                    n+=1
                    vid=it.next(i)
                    c=c+'\nPuesto '+str(n)+'\ntrending_date: '+str(vid['trending_
date'])+'; title: '+vid['title']+'; channel_title: '+vid['channel_title']+'; publ
ish_time: '+vid['publish_time']+'; views: '+vid['views']+'; likes: '+vid['likes']
+ '; dislikes: '+vid['dislikes']+'\n'
                return c
```

1:

```
def categoriaporID(name,catalog):
    categorias=catalog['categories']
    i=1
    while i<=lt.size(categorias):
        c=lt.getElement(categorias,i)
        if name.lower() in (c['name']).lower():
            return c['id']
        i+=1
```

O(N) donde N es el número de elementos de catalog[“categories”]

2:

```
def lporcategoria(ID,lista):
    final=lt.newList(datastructure='ARRAY_LIST')
    i=it.newIterator(lista)
    while it.hasNext(i):
        v=it.next(i)
        if v['category_id']==ID:
            lt.addFirst(final,v)
    if lt.isEmpty(final)==True:
        return None
    else:
        return final
```

O(N) donde N es el número de elementos de la lista dada

3:

```
def sortVideos(lista,size,cmpfunction):
    if size <= lt.size(lista):
        sub_list = lt.subList(lista, 1, size)
        sub_list = sub_list.copy()
        start_time=time.process_time()
        mrge.sort(sub_list, cmpfunction)
        stop_time=time.process_time()
        elapsed_time_mseg = round((stop_time - start_time)*1000,2)
        return elapsed_time_mseg, sub_list
    else:
        return None
```

O(N*log(N)) donde N es el tamaño a “sortear” de la lista dada.

Conclusión:

Entonces, es claro que la complejidad del Requerimiento 1 es O(N*log(N))

REQ 2:

```
def R2(pais,catalog):
    l1 = model.lporpais(pais,catalog['videos']) # 1
    if l1==None:
        return 'No hay información para este país.'
    else:
        l2 = model.sortVideos(l1,lt.size(l1),model.cmpVideosbyTitleandDate)[1] #2
        tupla=model.maxrep('title',l2) # 3
        return 'title: '+tupla[0]+'; channel_title: '+tupla[1]+'; country: '+tupl
a[3]+'; días: '+str(tupla[4])
```

1:

```
def categoriaporID(name,catalog):
    categorias=catalog['categories']
    i=1
    while i<=lt.size(categorias):
        c=lt.getElement(categorias,i)
        if name.lower() in (c['name']).lower():
            return c['id']
        i+=1
```

O(N) donde n es el tamaño de la lista catalog[“categories”]

2:

```
def sortVideos(lista,size,cmpfunction):
    if size <= lt.size(lista):
        sub_list = lt.subList(lista, 1, size)
        sub_list = sub_list.copy()
        start_time=time.process_time()
        mrge.sort(sub_list, cmpfunction)
        stop_time=time.process_time()
        elapsed_time_mseg = round((stop_time - start_time)*1000,2)
        return elapsed_time_mseg, sub_list
    else:
        return None
```

O(N*log(N)) donde N es el tamaño a “sortear” de la lista dada.

3:

```
def maxnorep(parametro,lista):
    title=''
    mayortotal=0
    mayorparcial=1
    n=2
    while n<=lt.size(lista):
        vid=lt.getElement(lista,n)
        ant=lt.getElement(lista,n-1)
        l=lt.newList()
        lt.addLast(l,ant['trending_date'])
        if vid[parametro]==ant[parametro]:
            if lt.isPresent(l,vid['trending_date'])==0:
                lt.addLast(l,vid['trending_date'])
                mayorparcial+=1
        else:
            l=lt.newList()
            lt.addLast(l,vid['trending_date'])
            if mayorparcial>mayortotal:
                mayortotal=mayorparcial
                title=ant['title']
                channel_title=ant['channel_title']
                category_id=ant['category_id']
                country=ant['country']

            mayorparcial=1
        n+=1

    if mayorparcial>mayortotal:
        mayortotal=mayorparcial
        title=ant['title']
        channel_title=ant['channel_title']
        category_id=ant['category_id']
        country=ant['country']

    return title,channel_title,category_id,country,mayortotal
```

Peor caso (todos los videos son iguales): O(N^2)

Conclusión:

Entonces, la complejidad del Requerimiento 2 es O(N^2)). Sin embargo, los datos no son este peor caso, lo cual indica que se encuentra entre O(Nlog(N)) y O(N^2)

REQ 3:

```
def R3(categoria,rep,catalog):
    ID=model.categoriaporID(categoria,catalog) # 1
    if ID==None:
        return 'Categoría no válida'
    else:
        l1=model.lporcategoria(ID,catalog['videos']) # 2
        l2=model.sortVideos(l1,lt.size(l1),model.cmpVideosbyTitleandDate)[1] #3
        if rep==1:
            tupla=model.maxnorep('title',l2) # 4
        elif rep==0:
            tupla=model.maxrep('title',l2) # 5
        else:
            return 'La opción ingresada (diferente a 0 y 1) no es válida'

        return 'title: '+tupla[0]+'; channel_title: '+tupla[1]+'; category_id: '+
tupla[2]+'; días: '+str(tupla[4])
```

1:

```
def categoriaporID(name,catalog):
    categorias=catalog['categories']
    i=1
    while i<=lt.size(categorias):
        c=lt.getElement(categorias,i)
        if name.lower() in (c['name']).lower():
            return c['id']
        i+=1
```

O(N) donde n es el tamaño de la lista catalog[“categories”]

2:

```
def lporcategoria(ID,lista):
    final=lt.newList(datastructure='ARRAY_LIST')
    i=it.newIterator(lista)
    while it.hasNext(i):
        v=it.next(i)
        if v['category_id']==ID:
            lt.addFirst(final,v)
    if lt.isEmpty(final)==True:
        return None
    else:
        return final
```

O(N) donde N es el número de elementos de la lista dada

3:

```
def sortVideos(lista,size,cmpfunction):
    if size <= lt.size(lista):
        sub_list = lt.subList(lista, 1, size)
        sub_list = sub_list.copy()
        start_time=time.process_time()
        mrge.sort(sub_list, cmpfunction)
        stop_time=time.process_time()
        elapsed_time_mseg = round((stop_time - start_time)*1000,2)
        return elapsed_time_mseg, sub_list
    else:
        return None
```

O(N*log(N)) donde N es el tamaño a “sortear” de la lista dada.

4 y 5:

```
def maxnorep(parametro,lista):
    title=''
    mayortotal=0
    mayorparcial=1
    n=2
    while n<=lt.size(lista):
        vid=lt.getElement(lista,n)
        ant=lt.getElement(lista,n-1)
        l=lt.newList()
        lt.addLast(l,ant['trending_date'])
        if vid[parametro]==ant[parametro]:
            if lt.isPresent(l,vid['trending_date'])==0:
                lt.addLast(l,vid['trending_date'])
                mayorparcial+=1
        else:
            l=lt.newList()
            lt.addLast(l,vid['trending_date'])
            if mayorparcial>mayortotal:
                mayortotal=mayorparcial
                title=ant['title']
                channel_title=ant['channel_title']
                category_id=ant['category_id']
                country=ant['country']

            mayorparcial=1
```

```
        n+=1

    if mayorparcial>mayortotal:
        mayortotal=mayorparcial
        title=ant['title']
        channel_title=ant['channel_title']
        category_id=ant['category_id']
        country=ant['country']

    return title,channel_title,category_id,country,mayortotal
```

```
def maxnorep(parametro,lista):
    title=''
    mayortotal=0
    mayorparcial=1
    n=2
    while n<=lt.size(lista):
        vid=lt.getElement(lista,n)
        ant=lt.getElement(lista,n-1)
        l=lt.newList()
        lt.addLast(l,ant['trending_date'])
        if vid[parametro]==ant[parametro]:
            if lt.isPresent(l,vid['trending_date'])==0:
                lt.addLast(l,vid['trending_date'])
                mayorparcial+=1
        else:
            l=lt.newList()
            lt.addLast(l,vid['trending_date'])
            if mayorparcial>mayortotal:
                mayortotal=mayorparcial
                title=ant['title']
                channel_title=ant['channel_title']
                category_id=ant['category_id']
                country=ant['country']

            mayorparcial=1
        n+=1

    if mayorparcial>mayortotal:
        mayortotal=mayorparcial
        title=ant['title']
        channel_title=ant['channel_title']
        category_id=ant['category_id']
        country=ant['country']
```

```
    return title,channel_title,category_id,country,mayortotal
```

Peor caso (todos los videos son iguales): O(N^2)

Conclusión:

Entonces, la complejidad del Requerimiento 3 es O(N^2)). Sin embargo, los datos no son este peor caso, lo cual indica que se encuentra entre O(Nlog(N)) y O(N^2)

REQ 4:

```
def R4(tag,pais,num,catalog):
    l1=model.lportyp(tag,pais,catalog['videos']) # 1
    if l1==None:
        return 'No hay información para el país y/o tag ingresados.'
    else:
        orde=model.sortVideos(l1,lt.size(l1),model.cmpVideosbyLikes)[1] # 2
        """
        print("orde")
        print(lt.subList(orde,1,10))
        print("")
        """

        final=model.sacar(num,orde) # 3
        if final==None:
            return 'El número ingresado excede la cantidad de videos que cumplen
con los requisitos.'
        else:
            c=''
            i=it.newIterator(final)
            n=0
            while it.hasNext(i): # 4
                n+=1
                v=it.next(i)
                c=c+'\nPuesto '+str(n)+'\ntitle: '+v['title']+'; channel_title: '
+v['channel_title']+'; publish_time: '+str(v['publish_time'])+'; views: '+str(v['
views'])+'; likes: '+str(v['likes'])+'; dislikes: '+str(v['dislikes'])+'; tags: '
+v['tags']+'\n'
            return c
```

1:

```
def lportyp(tag,pais,lista):
    i=it.newIterator(lista)
    final=lt.newList(datastructure='ARRAY_LIST')
    while it.hasNext(i):
        x=it.next(i)
        if tag in x['tags'] and pais.lower()==x['country']:
            lt.addLast(final,x)
    if lt.isEmpty(final)==True:
        return None
    else:
        return final
```

O(N)

2:

```
def sortVideos(lista,size,cmpfunction):
    if size <= lt.size(lista):
        sub_list = lt.subList(lista, 1, size)
        sub_list = sub_list.copy()
        start_time=time.process_time()
        mrge.sort(sub_list, cmpfunction)
        stop_time=time.process_time()
        elapsed_time_mseg = round((stop_time - start_time)*1000,2)
        return elapsed_time_mseg, sub_list
    else:
        return None
```

O(N*log(N)) donde N es el tamaño a “sortear” de la lista dada.

3:

```
def sacar(num,lista):
    if num<=lt.size(lista):
        titulos=lt.newList(datastructure="ARRAY_LIST")
        print(titulos)
        final=lt.newList(datastructure="ARRAY_LIST")
        print(final)
        primero=lt.firstElement(lista)
        print(primero)
        lt.addLast(titulos,primero['title'])
```

```
        lt.addLast(final,primero)
        i=it.newIterator(lista)
        while it.hasNext(i) and lt.size(final)<num:
            v=it.next(i)
            tit = v["title"]
            x = lt.isPresent(titulos,tit)
            if x == 0:
                lt.addLast(titulos,v['title'])
                lt.addLast(final,v)
        return final
    else:
        return None
```

O(N^2) debido al while y al lt.isPresent donde N = num

4: O(N) donde N = num

Conclusión:

Entonces, la complejidad del Requerimiento 4 es O(N^2)). Sin embargo, es eficiente porque los valores de N no son considerablemente grandes en los requerimientos pedidos.