Análisis de Complejidad Reto-1

Integrantes:

- E1: Daniela Alvarez Rodriguez-202020209-d.alvarezr@uniandes.edu.co
- E2: Andrés Mendoza Silva-202012676-af.mendoza@uniandes.edu.co

Requerimiento 1: Video Tendencia por categoría y país

```
def videos_categoria_pais(catalog, categoria, pais, numero):
(NlogN)videos = sortVideos(catalog["videos"], "None", 4, cmpviews)
(M)    categoria = categorias(catalog,categoria)
(1)    lista_videos = lt.newList(datastructure='ARRAY_LIST')

(N)    for i in range(1, lt.size(videos)):
(1)        video = lt.getElement(videos, i)
(1)        if video["category_id"] == categoria:
(1)            if video["country"].lower() == pais.lower():
(1)                if numero > 0:
(7)                    vid_t = {"Nombre del video": video["title"], "Trending date": video["trending_date"],
                        "Nombre del canal": video["channel_title"], "Fecha Publicación": video["publish_time"],
                        "Reproducciones": video["views"], "Likes": video["likes"], "Dislikes": video["dislikes"]}
(1)                    lt.addLast(lista_videos, vid_t)
(1)                    numero-=1
(1)                elif numero == 0:
(1)                    break

    return lista_videos
```

Cálculo de complejidad:

N= lt.size(catalog[“videos”])

M = lt.size(catalog[“categorias”])

M<N

$$O(n) = NlogN + \mathrm{M} + 1 + 13N = 13N = N$$

Discusión

La complejidad de este algoritmo utilizando la notación BIG O es O(n) esto quiere decir que su orden de crecimiento es lineal por lo que el tiempo de procesamiento será proporcional a la cantidad de datos

Requerimiento 2: Video tendencia por país (Daniela Alvarez Rodriguez)

```
def video_tendencia_pais(catalog, pais):
 (1) videos_pais = {}
 (1) tendencia_videos = {}
 (N)  for i in range(1, lt.size(catalog["videos"])):
 (1)    video = lt.getElement(catalog["videos"], i)
 (1)    if video["country"].lower() == pais.lower():
 (1)           if video["video_id"] in tendencia_videos:
 (1)               tendencia_videos[video["video_id"]] = tendencia_videos[video["video_id"]] + 1
 (1)           else:
 (1)               tendencia_videos[video["video_id"]] = 1
 (1)               videos_pais[video["video_id"]]= video

 (1) mas_dias = 0
 (1) video = {}
 (N)   for i in tendencia_videos:
 (1)      if tendencia_videos[i] > mas_dias:
 (1)           mas_dias = tendencia_videos[i]
 (1)           video = videos_pais[i]

 (1) video["Dias Tendencia"] = mas_dias
    return video
```

Cálculo de complejidad:

N = lt.size(catalog[“videos”])

$$O(n) = 2 + 5N + 2 + 3N = 8N + 4 = 8N = N$$

Discusión:

La complejidad de este algoritmo, utilizando la notación Big O es O(n). Esto quiere decir que su orden de crecimiento es lineal y el tiempo de procesamiento será proporcional al numero de datos procesados.

Función Consulta Categorías (Se utilizará más adelante)

```
def categorias(catalog, categoria):
(1)    categorias = catalog["categorias"]
(N)    for i in range(lt.size(categorias)):
(1)        a = lt.getElement(categorias,i)
(1)        if categoria.lower() in a["name"].lower():
(1)            return a["id"]
```

Cálculo de complejidad:

N = lt.size(catalog[“categorias”])

$$O(n) = 1 + 3N = 3N = N$$

Requerimiento 3: Video tendencia por categoría(Andrés Mendoza Silva)

```
def video_tendencia_categoria(catalog, categoria):
(M)    idcategoria = categorias(catalog, categoria)
(1)    categoria_veces={}
(1)    categoria_p1={}
(1)    mayor=0
(1)    respuesta =""
(1)    entregar = ""

(N)    for i in range(lt.size(catalog["videos"])):
(1)        objeto=lt.getElement(catalog["videos"],i)
(1)        if objeto["category_id"] == idcategoria:
(1)            if objeto["video_id"] in categoria_veces:
(1)                categoria_veces[objeto["video_id"]]=categoria_veces[objeto["video_id"]]+1
(1)            else:
(1)                categoria_veces[objeto["video_id"]]=1
(1)                categoria_p1[objeto["video_id"]]=objeto

(P)    for o in categoria_veces:
(1)        if categoria_veces[o] > mayor:
(1)            mayor=categoria_veces[o]
(1)            respuesta=(o,categoria_veces[o])
(Q)    for h in categoria_p1:
(1)        if respuesta[0] == h:
(1)            entregar=categoria_p1[h]
(1)            break
    return  (entregar["title"], entregar["channel_title"],entregar["category_id"], mayor)
```

Cálculo de complejidad:

N= lt.size(catalog[“videos”])

M = lt.size(catalog[“categorias”])

P = len(categoría_veces)

Q = len(categoría_p1)

$$M < Q, P < N$$

$$O(N) = M + 5 + 5N + 3P + 3Q = 5N = N$$

La complejidad de este algoritmo utilizando la notación BIG O es O(n) esto quiere decir que su orden de crecimiento es lineal por lo que el tiempo de procesamiento será proporcional a la cantidad de datos

Requerimiento 4: Video con más likes

```
def videos_likes(catalog, pais, tag, numero):
(1)    videos_pais = lt.newList(datastructure="ARRAY_LIST")

(N)    for i in range(1, lt.size(catalog["videos"])):
(1)        video = lt.getElement(catalog["videos"], i)
(1)        if video["country"].lower() == pais.lower():
(1)            lista_tag = video["tags"]
(M)            for e in range(len(lista_tag)):
(1)                if tag in lista_tag[e]:
(1)                    lt.addLast(videos_pais,video)

(QlogQ) videos = sortVideos(videos_pais, "None", 4, comparelikes)
(1)     vids = lt.subList(videos, 1, numero)
(1)     respuesta = lt.newList()

(P)    for i in range(1, lt.size(vids)):
(1)        video = lt.getElement(vids, i)
(7)        vid_t = {"Nombre del video": video["title"], "Nombre del canal": video["channel_title"],
                "Fecha Publicación": video["publish_time"],"Reproducciones": video["views"],
                    "Likes": video["likes"], "Dislikes": video["dislikes"], "Tags": video["tags"]}
(1)        lt.addLast(respuesta, vid_t)

    return respuesta
```

Cálculo de complejidad:

N = lt.size(catalog[“videos”])

M = lt.size(lista_tags) “*Lista de tags de cada video”*

Q = lt.size(videos_pais)

P = lt.size(vids) = lt.size(lt.subList(videos_pais)

$$P < Q < N$$

$$O(n) = 1 + (2N)(2M) + QlogQ + 2 + 9P =$$

$$4NM + QlogQ + 9P + 3 = 4NM = NM$$

Discusión:

La complejidad de este algoritmo, utilizando la notación Big O es O(NM). Esto quiere decir que su orden de crecimiento es cuadrático y el tiempo de procesamiento aumenta considerablemente a medida que se procesan mas datos.