**Análisis de Complejidad Reto-2**

Integrantes:

* E1: Daniela [Alvarez Rodriguez-202020209-d.alvarezr@uniandes.edu.co](mailto:Alvarez%20Rodriguez-202020209-d.alvarezr@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

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

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.