**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, nombrecategoria, pais, numero):

(1)    nombrecategoria = nombrecategoria.replace(" ", "").lower()

(1)    categoria = mp.get(catalog['videos\_por\_categoria'], nombrecategoria)

(1)    videos = me.getValue(categoria)

(NlogN)videos\_ordenados = sortVideos(videos, compareviews)

(1)    lista\_videos = []

(N)    for i in range(1, lt.size(videos\_ordenados)):

(1)        video = lt.getElement(videos\_ordenados, i)

(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)                lista\_videos.append(vid\_t)

(1)                numero-=1

(1)            elif numero == 0:

(1)                break

    return lista\_videos

Cálculo de complejidad:

N= Tamaño lista videos para la categoria

Discusión

La complejidad de este algoritmo utilizando la notación BIG O es O(NlogN) esto quiere decir que su orden de crecimiento es linearítmico por lo que el tiempo de procesamiento va a aumentar en menor proporción a medida que aumenta la cantidad de datos.

***Función videos tendencia:*** (Se utilizará más adelante)

def video\_tendencia(videos):

(1)    videos\_por\_id = mp.newMap(lt.size(videos),

                             maptype='CHAINING',

                             loadfactor=2.0,

                             comparefunction=comparecategories)

(1)    tendencia\_videos = mp.newMap(lt.size(videos),

                             maptype='CHAINING',

                             loadfactor=2.0,

                             comparefunction=comparecategories)

(N)    for i in range(1, lt.size(videos)):

(1)        video = lt.getElement(videos, i)

(M)        if mp.contains(tendencia\_videos, video["video\_id"]):

(1)            vid = mp.get(tendencia\_videos, video["video\_id"])

(1)            dias\_tendencia = me.getValue(vid) + 1

(1)        else:

(1)            dias\_tendencia = 1

(1)            mp.put(videos\_por\_id, video["video\_id"], video)

(1)        mp.put(tendencia\_videos, video["video\_id"], dias\_tendencia)

(1)    mas\_dias = 0

(1)    video = {}

(M)    keys = mp.keySet(tendencia\_videos)

(M)    for i in lt.iterator(keys):

(1)        pareja\_dias = mp.get(tendencia\_videos, i)

(1)        num\_dias = me.getValue(pareja\_dias)

(1)        if num\_dias > mas\_dias:

(1)            mas\_dias = num\_dias

(1)            vid = mp.get(videos\_por\_id, i)

(1)            video = me.getValue(vid)

(1)    video["Dias Tendencia"] = mas\_dias

    return video

Cálculo de complejidad:

N= Tamaño lista videos entrados por parámetro

M = Tamaño Map donde aparece solo una llave por id de video

***Requerimiento 2:*** Video tendencia por país

def video\_tendencia\_pais(catalog, pais):

(1)     pais = pais.replace(" ", "").lower()

(1)     videos\_pais = mp.get(catalog['videos\_por\_pais'], pais)

(1)     videos = me.getValue(videos\_pais)

(NlogN)   sortVideos(videos, comparelikes)

(NM)     return video\_tendencia(videos)

Cálculo de complejidad:

N = Tamaño lista de videos para el país

M = Tamaño Map donde aparece solo una llave por id de video

O(n) = 3 + NlogN + NM = NM

Discusión:

La complejidad de este algoritmo, utilizando la notación Big O es O(nm). Esto quiere decir que su orden de crecimiento es cuadrática y el tiempo de procesamiento incrementará considerablemente entra más datos procesados

***Requerimiento 3:*** Video tendencia por categoría

def video\_tendencia\_categoria(catalog, categoria):

(1)     categoria = categoria.replace(" ", "").lower()

(1)    videos\_categoria = mp.get(catalog['videos\_por\_categoria'], categoria)

(1)    videos = me.getValue(videos\_categoria)

(NlogN) SortVideos(videos, comparelikes)

(NM)    return video\_tendencia(videos)

Cálculo de complejidad:

N = Tamaño lista de videos para el país

M = Tamaño Map donde aparece solo una llave por id de video

O(n) = 3 + NlogN + NM = NM

Discusión:

La complejidad de este algoritmo, utilizando la notación Big O es O(nm). Esto quiere decir que su orden de crecimiento es cuadrática y el tiempo de procesamiento incrementará considerablemente entra más datos procesados

***Requerimiento 4:*** Video con más likes

def videos\_likes(catalog, pais, tag, numero):

(1)    pais = pais.replace(" ", "").lower()

(1)    videos\_pais = mp.get(catalog['videos\_por\_pais'], pais)

(1)    videos = me.getValue(videos\_pais)

(NlogN)videos\_ordenados = sortVideos(videos, comparelikes)

(1)    lista\_videos = []

(N)    for i in range(1, lt.size(videos)):

(1)        video = lt.getElement(videos\_ordenados, i)

(1)        lista\_tag = video["tags"]

(T)        for e in range(len(lista\_tag)):

(1)            if tag in lista\_tag[e]:

(1)                if numero > 0:

(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)                    lista\_videos.append(vid\_t)

(1)                    numero-= 1

(1)                elif numero == 0:

(1)                    break

    return lista\_videos

Cálculo de complejidad:

N = Tamaño lista videos para el país

P = Tamaño lista de tags para cada video

P < N

O(n) = 3 + NlogN + 1 + 22NT = NlogN + 22NT + 4 = NlogN

Discusión:

La complejidad de este algoritmo utilizando la notación BIG O es O(NlogN), esto quiere decir que su orden de crecimiento es linearítmico por lo que el tiempo de procesamiento va a aumentar en menor proporción a medida que aumenta la cantidad de datos.