Estructuras de Datos y Algoritmos

Jose Luis Tavera - 201821999
Juan Diego Yepes - 202022391

Reto I

1 Primer Requerimiento

El equipo de análisis quiere conocer cuáles son los n videos con más views que son tendencia en un determinado país, dada una categoría específica. Para dar respuesta a este requerimiento el equipo de desarrollo debe recibir como entrada la siguiente información:

- category_name
- country
- Número de videos que quiere listar (n)

Y como respuesta debe presentar en consola la siguiente información:

- trending_date
- title
- channel_title
- publish_time
- views
- likes
- dislikes

Antes de analizar los ordenes de crecimiento del requerimiento, señalaremos las funciones y sus tareas dentro del modelo MVC. Por lo tanto, empezaremos por el view, seguido del controller y finalmente haremos el análisis del orden principalmente en las funciones del model.

Funciones del View

```
elif str(inputs[0]) == "2":
        pais = input("\nIngrese el pais de referencia: ")
        if pais.lower() in lista:
            printCategorias()
            categoria = (input('Ingrese la categoria de referencia: '))
            if categoria in no_categorias:
                n = int(input(
                    "Ingrese el numero de videos que desea imprimir: "))
                print("\nCargando ....")
                resultado = controller.Requerimiento_2(
                    catalog['country'], categoria, pais)
                print(
                    "Para la muestra de",
                    lt.size(catalog['country']),
                    "elementos, el tiempo (mseg) es:",
                    str(resultado[0]))
```

```
                print("\n")
                printResults(resultado[1], int(n))

            else:
                print("\n")
                print("No se encontr  la Categoria")
        else:
            print("\n")
            print("No se encontr  el pais")
```

La función del View del primer requerimiento es invocada por el usuario al ingresar el número "2". Una vez invocada, le pide al usuario tres parámetros: i) el país, ii) la categoría y iii) El tamaño del sample. Estas funcionarán como parámetros de las funciones del controller y el model en las siguientes instancias.

Sin embargo, en el view se encuentra la función encargada de imprimir el número de videos correspondientes, la cual es printResults():

```
def printResults(ord_videos, sample):
    size = lt.size(ord_videos)
    if size > sample:
        print("Los primeros ", sample, " videos ordenados son:")
        i = 1
        while i <= sample:
            video = lt.getElement(ord_videos, i)
            print("\n")
            print(
                'T tulo: ' + str(video.get('title')) + ", " +
                'Nombre del canal: ' + str(video.get('channel_title')) + ", " +
                'Fue tendencia el d a: ' + str(video.get('trending_date'))
                + ", " +
                'Visitas: ' + str(video.get('views')) + ", " +
                'Likes: ' + str(video.get('likes')) + ", " +
                'Dislikes: ' + str(video.get('dislikes')) + ", " +
                'Fecha de publicaci n: ' + str(video.get('dislikes')))
            i += 1
```

Hacemos énfasis en esta función ya que itera la lista sorteada, según el número de videos que se hayan pedido, aunque su orden decrecimiento es pequeño, este es mayor a O(1). No obstante, posteriormente haremos el análisis completo

Funciones del Controller

Análogamente, la función del controller como intermediario entre el view y el model es bastante sencilla al invocar dos funciones:

```
def Requerimiento_1(catalogo, categoria, pais):
    result1 = model.getVideosByCategoryAndCountry(
                catalogo, categoria, pais)
    result = model.sortVideos(
                result1, lt.size(result1), 'ms', 'cmpVideosByViews')
    return result
```

Como es posible apreciar, el controller invoca dos funciones getVideosByCategoryAndCountry() y sortVideos(), esto lo hace para primero filtrar la lista y después sortearla ya que de otro modo el orden de crecimiento sería mayor.

Funciones del Model

En el modelo, esas dos funciones corresponden a:

```
def getVideosByCategoryAndCountry(catalog, category, country):
    sublist = getVideosByCountry(catalog, country)
    sublist2 = getVideosByCategory(sublist, category)
    return sublist2
```

Compuesta subsecuentemente por otras dos funciones que son:

```
def getVideosByCountry(catalog, country):
    operating = True
    i = 1
    while operating and i <= lt.size(catalog):
        pais = lt.getElement(catalog, i)
        nombre_pais = pais.get('country_name')
        if nombre_pais == country:
            operating = False
        else:
            i += 1
    return pais['video']
```

```
def getVideosByCategory(videos, category):
    lista = lt.newList('ARRAY_LIST', cmpVideosByViews)
    i = 1
    while i <= lt.size(videos):
        c_id = int(lt.getElement(videos, i).get('category_id'))
        if category == c_id:
            element = lt.getElement(videos, i)
            lt.addLast(lista, element)
        i += 1

    return lista
```

Finalmente, está la función de sortVideos():

```
def sortVideos(catalog, size, sort_type, cmp):
    sub_list = lt.subList(catalog, 0, size)
    sub_list = sub_list.copy()
    start_time = time.process_time()

    if cmp == 'cmpVideosByViews':
        if sort_type == "ms":
            sorted_list = ms.sort(sub_list, cmpVideosByViews)
        elif sort_type == "qs":
            sorted_list = qs.sort(sub_list, cmpVideosByViews)

    if cmp == 'comparetitles':
        if sort_type == "ms":
            sorted_list = ms.sort(sub_list, comparetitles)
        elif sort_type == "qs":
            sorted_list = qs.sort(sub_list, comparetitles)

    if cmp == 'comparelikes':
        if sort_type == "ms":
            sorted_list = ms.sort(sub_list, comparelikes)
        elif sort_type == "qs":
            sorted_list = qs.sort(sub_list, comparelikes)

    stop_time = time.process_time()
    elapsed_time_mseg = (stop_time - start_time)*1000
    return elapsed_time_mseg, sorted_list
```

2 Segundo Requerimiento (Juan Diego Yepes)

El equipo de análisis quiere conocer cuál es el video que más días ha sido trending para un país específico. Para dar respuesta a este requerimiento el equipo de desarrollo debe recibir como entrada la siguiente información:

- country

Y como respuesta debe presentar en consola la siguiente información:

- title
- channel_title
- country
- número de días

Antes de analizar los ordenes de crecimiento del requerimiento, señalaremos las funciones y sus tareas dentro del modelo MVC. Por lo tanto, empezaremos por el view, seguido del controller y finalmente haremos el análisis del orden principalmente en las funciones del model.

Funciones del View

El segundo requerimiento es invocado en el view cuando el usuario ingresa la opción número "3":

```
elif str(inputs[0]) == "3":
        pais = input("Ingrese el pa s de referencia: ")
        print("\nCargando ....")
        lista = controller.getVideosByCountry(catalog['country'], pais)
        result = controller.sortVideos(
            lista, lt.size(lista), 'ms', 'comparetitles')[1]
        dias_tendencia = controller.getMostTrendingDays(result)
        print("\n")
        printResultsv3(dias_tendencia)
```

En el anterior fragmento de código vemos que la función utiliza otras tres funciones del controller para la filtración y ordenación de los datos.

Funciones del Controller

Asimismo, estas tres funciones llaman a tres funciones en el modelo:

```
def getVideosByCountry(catalog, country):
    return model.getVideosByCountry(catalog, country)
```

```
def sortVideos(catalog, size, sort_type, cmp):
    return model.sortVideos(catalog, size, sort_type, cmp)
```

```
def getMostTrendingDays(catalog):
    return model.getMostTrendingDaysByTitle(catalog)
```

Funciones del Model

Esta función la utilizamos en el primer requerimiento, y la usaremos también para el cuarto:

```
def getVideosByCountry(catalog, country):
    operating = True
    i = 1
    while operating and i <= lt.size(catalog):
        pais = lt.getElement(catalog, i)
        nombre_pais = pais.get('country_name')
        if nombre_pais == country:
            operating = False
        else:
            i += 1
    return pais['video']
```

Esta función es usada en los cuatro requerimientos cada uno con una compare function distinta que no anexamos debido a su simplicidad:

```
def sortVideos(catalog, size, sort_type, cmp):
    sub_list = lt.subList(catalog, 0, size)
    sub_list = sub_list.copy()
    start_time = time.process_time()

    if cmp == 'cmpVideosByViews':
        if sort_type == "ms":
            sorted_list = ms.sort(sub_list, cmpVideosByViews)
        elif sort_type == "qs":
            sorted_list = qs.sort(sub_list, cmpVideosByViews)

    if cmp == 'comparetitles':
        if sort_type == "ms":
            sorted_list = ms.sort(sub_list, comparetitles)
        elif sort_type == "qs":
            sorted_list = qs.sort(sub_list, comparetitles)

    if cmp == 'comparelikes':
        if sort_type == "ms":
            sorted_list = ms.sort(sub_list, comparelikes)
        elif sort_type == "qs":
            sorted_list = qs.sort(sub_list, comparelikes)

    stop_time = time.process_time()
    elapsed_time_mseg = (stop_time - start_time)*1000
    return elapsed_time_mseg, sorted_list
```

Finalmente, la función de getMostTrendingDaysByTitle() la usamos igual para el requerimiento 2 y 3:

```
def getMostTrendingDaysByTitle(videos):
    elemento = lt.firstElement(videos)
    mayor_titulo = None
    mayor = 0
    i = 0

    for video in lt.iterator(videos):
        if video['video_id'] == elemento['video_id']:
            i += 1
        else:
            if i > mayor:
                mayor_titulo = elemento
                mayor = i
            i = 1
            elemento = video

    if i > mayor:
        mayor_titulo = elemento
        mayor = i
    return (mayor_titulo, mayor)
```

3 Tercer Requerimiento (Jose Luis Tavera Ruiz)

El equipo de análisis quiere conocer cuál es el video que más días ha sido trending para una categoría específica.Para dar respuesta a este requerimiento el equipo de desarrollo debe recibir como entrada la siguiente información:

- country

Y como respuesta debe presentar en consola la siguiente información:

- title
- channel_title
- country
- número de días

Antes de analizar los ordenes de crecimiento del requerimiento, señalaremos las funciones y sus tareas dentro del modelo MVC. Por lo tanto, empezaremos por el view, seguido del controller y finalmente haremos el análisis del orden principalmente en las funciones del model.

Funciones del View

En el momento que el usuario ingresa el número 4, el requerimiento tres se ejecuta en el view invocando otras tres funciones del controller:

```
elif str(inputs[0]) == "4":
        categoria = int(input('Ingrese la categor a de referencia: '))
        print("\nCargando ....")
        result1 = controller.getVideosByCategory(
            catalog['video'], categoria)
        result = controller.sortVideos(
            result1, lt.size(result1), 'ms', 'comparetitles')[1]

        video_tendencia = controller.getMostTrendingDays(result)
        print("\n")
        printResultsv3(video_tendencia)
```

Funciones del Controller

Igualmente, las tres funciones del controler invocan otras tres funciones del model:

```
def getVideosByCategory(catalog, categoria):
    return model.getVideosByCategory(catalog, categoria)
```

```
def sortVideos(catalog, size, sort_type, cmp):
    return model.sortVideos(catalog, size, sort_type, cmp)
```

```
def getMostTrendingDays(catalog):
    return model.getMostTrendingDaysByTitle(catalog)
```

Funciones del Model

Primero hacemos una filtración similar al del requierimiento previo pero hecha por categoría en vez de país:

```
def getVideosByCategory(videos, category):
    lista = lt.newList('ARRAY_LIST', cmpVideosByViews)
    i = 1
    while i <= lt.size(videos):
        c_id = int(lt.getElement(videos, i).get('category_id'))
        if category == c_id:
            element = lt.getElement(videos, i)
            lt.addLast(lista, element)
        i += 1

    return lista
```

La función de es sortVideos usada en los cuatro requerimientos cada uno con una compare function distinta que no anexamos debido a su simplicidad:

```
def sortVideos(catalog, size, sort_type, cmp):
    sub_list = lt.subList(catalog, 0, size)
    sub_list = sub_list.copy()
    start_time = time.process_time()

    if cmp == 'cmpVideosByViews':
        if sort_type == "ms":
            sorted_list = ms.sort(sub_list, cmpVideosByViews)
        elif sort_type == "qs":
            sorted_list = qs.sort(sub_list, cmpVideosByViews)

    if cmp == 'comparetitles':
        if sort_type == "ms":
            sorted_list = ms.sort(sub_list, comparetitles)
        elif sort_type == "qs":
            sorted_list = qs.sort(sub_list, comparetitles)

    if cmp == 'comparelikes':
        if sort_type == "ms":
            sorted_list = ms.sort(sub_list, comparelikes)
        elif sort_type == "qs":
            sorted_list = qs.sort(sub_list, comparelikes)

    stop_time = time.process_time()
    elapsed_time_mseg = (stop_time - start_time)*1000
    return elapsed_time_mseg, sorted_list
```

Finalmente, la función de getMostTrendingDaysByTitle() la usamos igual para el requerimiento 2 y 3:

```
def getMostTrendingDaysByTitle(videos):
    elemento = lt.firstElement(videos)
    mayor_titulo = None
    mayor = 0
    i = 0

    for video in lt.iterator(videos):
        if video['video_id'] == elemento['video_id']:
            i += 1
        else:
            if i > mayor:
                mayor_titulo = elemento
                mayor = i
            i = 1
            elemento = video

    if i > mayor:
        mayor_titulo = elemento
        mayor = i
    return (mayor_titulo, mayor)
```

4 Cuarto Requerimiento

El equipo de análisis quiere conocer cuáles son los n videos diferentes con más likes en un país con un tag específico. Recuerde que el campo tags está compuesto por múltiples palabras o frases entre comillas (””) y separadas por el siguiente caracter —, así que el tag requerido puede ser una sub-cadena en el campo.

- tag

Y como respuesta debe presentar en consola la siguiente información:

- title
- channel_title
- publish_time
- views
- likes
- dislikes
- tags

Antes de analizar los ordenes de crecimiento del requerimiento, señalaremos las funciones y sus tareas dentro del modelo MVC. Por lo tanto, empezaremos por el view, seguido del controller y finalmente haremos el análisis del orden principalmente en las funciones del model.

Funciones del View

En este caso solo usamos una función que invoca al controller además de las funciones accesorias que se encargan de imprimir los resultados:

```
elif str(inputs[0]) == "5":
        pais = input("Ingrese el pa s de referencia: ")
        tag = input('Ingrese el tag de referencia: ')
        n = int(input("Ingrese el n mero de videos que desea imprimir: "))
        print("\nCargando ....")

        result = controller.getVideosByCountryAndTag(
             catalog['country'], tag, pais)

        print(
            "Para la muestra de",
            lt.size(catalog['country']),
            "elementos, el tiempo (mseg) es:",
            str(result[0]))

        printResultsv2(result[1], n)
```

Además, para asegurarnos que los videos impresos sean distintos agregamos una función en el view de

```
def printResultsv2(ord_videos, sample):
    printlist = []
    i = 1
    while len(printlist) <= (sample - 1):
        element = lt.getElement(ord_videos, i)
        title = str(element.get('title'))
        if title not in printlist:
            printlist.append(title)
            print("\n")
            print(
                'T tulo: ' + str(element.get('title')) + ", " +
                'Nombre del canal: ' + str(element.get('channel_title'))
```

```
                + ", " + 'Visitas: ' + str(element.get('views')) + ", " +
                'Likes: ' + str(element.get('likes')) + ", " +
                'Dislikes: ' + str(element.get('dislikes')) + ", " +
                'Tags: ' + str(element.get('tags')))
        i += 1
```

Funciones del Controller

Esta función del controller invoca su función homónima del modelo:

```
def getVideosByCountryAndTag(catalog, tag, country):
    return model.getVideosByCountryAndTag(catalog, tag, country)
```

Funciones del Model

Como se puede apreciar en la siguiente función, se hacen dos filtraciones, primero por país y después por tags, y posteriormente, se ordena la lista.

```
def getVideosByCountryAndTag(catalog, tag, country):
    sublist = getVideosByCountry(catalog, country)
    sublist2 = getVideosByTag(sublist, tag)
    sorted_list = sortVideos(sublist2, int(lt.size(sublist2)), 'ms', 'comparelikes')
    return sorted_list
```

```
def getVideosByCountry(catalog, country):
    operating = True
    i = 1
    while operating and i <= lt.size(catalog):
        pais = lt.getElement(catalog, i)
        nombre_pais = pais.get('country_name')
        if nombre_pais == country:
            operating = False
        else:
            i += 1
    return pais['video']
```

```

def getVideosByTag(videos, tag):
    lista = lt.newList('ARRAY_LIST')
    i = 1

    while i <= lt.size(videos):
        c_tags = lt.getElement(videos, i).get('tags')
        tagpresence = tag in c_tags

        if tagpresence:
            element = lt.getElement(videos, i)
            lt.addLast(lista, element)

        i += 1

    return lista
```

```
def sortVideos(catalog, size, sort_type, cmp):
    sub_list = lt.subList(catalog, 0, size)
    sub_list = sub_list.copy()
    start_time = time.process_time()

    if cmp == 'cmpVideosByViews':
        if sort_type == "ms":
            sorted_list = ms.sort(sub_list, cmpVideosByViews)
        elif sort_type == "qs":
            sorted_list = qs.sort(sub_list, cmpVideosByViews)

    if cmp == 'comparetitles':
        if sort_type == "ms":
            sorted_list = ms.sort(sub_list, comparetitles)
```

```
        elif sort_type == "qs":
            sorted_list = qs.sort(sub_list, comparetitles)

    if cmp == 'comparelikes':
        if sort_type == "ms":
            sorted_list = ms.sort(sub_list, comparelikes)
        elif sort_type == "qs":
            sorted_list = qs.sort(sub_list, comparelikes)

    stop_time = time.process_time()
    elapsed_time_mseg = (stop_time - start_time)*1000
    return elapsed_time_mseg, sorted_list
```

5 Cálculos de Ordenes de Crecimiento

En este apartado haremos los cálculos de los ordenes de crecimiento de cada una de las funciones. Ya que sería negligente analizar las funciones en términos de "N" (siento este el número total de datos) definiremos las siguientes variables y haremos una aproximación en términos de N para su mejor y peor caso:

- P**:** Es el número total de países cargados por el catálogo que corresponde a 10 países.
- C**:** Es el número total de categorías cargadas por el catálogo que corresponde a 32 categorías.
- NP: Es el tamaño promedio de datos filtrados por país. Su mejor caso es igual a: NP_B y su peor caso es: NP_W
- NC: Es el tamaño promedio de datos filtrados por categoría. Su mejor caso es igual a: NC_B y su peor caso es: NC_W
- NPC : Es el tamaño promedio de una filtración por país y categoría. Su mejor caso es igual a: NPC_B y su peor caso es: NPC_W
- NPT : Es el tamaño promedio de una filtración por país y tags. Su mejor caso es igual a: NPT_B y su peor caso es: NPT_W

Ahora bien, analizaremos paso a paso las funciones que de mayor orden de crecimiento por requerimiento

- **Requerimiento 1:**

  getVideosByCountry: Se demora (P), ya que revisa las llaves del catálogo en el apartado de catalog["countries"].

  getVideosByCategory: Se demora (NP), ya que filtra la lista anterior por categoría para lo cual necesita iterarla.

  sortVideos: Al usar merge sort se demora ($NPC \cdot log(NPC)$ en su mejor, peor y caso promedio.

  printResultsv1: Se demora en congruencia con el parámetro "n", es decir (n) porque esas son las veces que itera la lista.

- **Requerimiento 2:**

  getVideosByCountry: Se demora (P), ya que revisa las llaves del catálogo en el apartado de catalog["countries"].

  sortVideos: Al usar merge sort se demora ($NP \cdot log(NP)$ en su mejor, peor y caso promedio.

  getMostTrendingDaysByTitle: Se demora (NP) ya que debe iterar toda la lista sorteada para buscar el mayor.

- **Requerimiento 3:**

  getVideosByCategory: Se demora (C), ya que revisa las llaves del catálogo en el apartado de catalog["category_id"].

  sortVideos: Al usar merge sort se demora ($NC \cdot log(NC)$ en su mejor, peor y caso promedio.

  getMostTrendingDaysByTitle: Se demora (NC) ya que debe iterar toda la lista sorteada.

- **Requerimiento 4:**

  getVideosByCountry: Se demora (P), ya que revisa las llaves del catálogo en el apartado de catalog["countries"]

  getVideosByTag: Se demora (NPT),ya que filtra la lista anterior por tags para lo cual necesita iterarla.

sortVideos: Al usar merge sort se demora $(NPT \cdot log(NPT)$ en su mejor, peor y caso promedio.

printResultsv2: Se demora (NPT) en el peor caso ya que debe escoger los mejores videos y asegurarse de no repetirlos iterando la lista de tamaño (NPT).

6 Resultados

En congruencia con lo anterior, haciendo los cálculos pertinentes para cada una de las notaciones en relación a las variables de tamaño definidas anteriormente, obtuvimos los siguientes ordenes de crecimiento para cada uno de los requerimientos:

	Requerimiento 1	**Requerimiento 2**
Notacion Tilda	$\sim N(15 + P + NP + (NPC \cdot log(NPC) + n)$	$\sim N(10 + P + NP + (NP \cdot log(NP))$
Big Theta	$\Theta(NP)$	$\Theta(NP)$
Big O	$O(NP_W)$	$O(NP_W)$
Big Omega	$\Omega(NP_B))$	$\Omega(NP_B))$

	Requerimiento 3	**Requerimiento 4**
Notacion Tilda	$\sim N(10 + NC + (NC \cdot log(NC))$	$\sim N(20 + P + 2NPT + (NPT \cdot log(NPT))$
Big Theta	$\Theta(NC)$	$\Theta(NPT \cdot log(NPT))$
Big O	$O(NC_W)$	$O(NPT_W \cdot log(NPT_W))$
Big Omega	$\Omega(NC_B)$	$\Omega(NPT_B \cdot log(NPT_B))$