ANÁLISIS DEL RETO

Daniel Roa Uribe, 202215803, d.roau@uniandes.edu.co

Jose Guevara Pedroza, 202213763, jd.guevarap1@uniandes.edu.co

Jesus Correcha Guarnizo, 202215016, j.correcha@uniandes.edu.co

Requerimiento <<Carga de datos>>

Cargar los datos de los documentos .csv en la estructura de datos esperada.

Descripción

La implementación se basó en el repositorio del laboratorio 4, sin embargo, se hicieron algunos cambios para que se pudiesen implementar los cambios de TAD y algoritmos de Sort. Al final la implementación quedó con las siguientes características:

- Diccionario de configuraciones:
  Para cumplir con los requisitos de cambiar el algoritmo de ordenamiento y el TAD con el que cargarían los datos, se decidió que en la vista hubiese un diccionario de configuraciones. Este diccionario tiene las características que se pasaran al controlador y al modelo al momento de cargar los datos y hacer ordenamientos de los datos dentro del modelo.

```
controller_characteristics={
    #Este diccionario se usa como referenc
    #componen al catalogo
    #tambien algunas otras caracteristicas
    #estas caracteristicas se usan dentro
    "videos":"ARRAY_LIST",
    "stream_services":"ARRAY_LIST",
    "data_size_sufijo":"-small",
    "sort_algoritm":"shell"
}
```

Este contiene el tipo de TAD que se usara en el catálogo de contenidos (en la implementación la lista de contenido se llama “videos”), el sufijo que deberán tener los archivos .csv que se van a cargar y el nombre del algoritmo Sort que se usara para ordenar los datos.

Este diccionario de configuraciones se pasa por parámetro a la mayoría de las funciones que ejecutan algún requerimiento.

- Lectura de archivos:

  Al momento de leer todos los documentos .csv se decidió que todos los archivos se leyeran y que se unieran al final en uno. Como ya se mencionó anteriormente, se usa el diccionario de características para elegir con que sufijo se cargara el archivo. Además, teniendo en cuenta el servicio de streaming asociado con dicho archivo, (Amazon, Disney, Hulu, Netflix) se hace uso de una función aparte que le agrega a cada nuevo registro (antes de agregarlo al catálogo) una nueva variable asociada al streaming service del que proviene.

```
def loadMovies(catalog, characteristics:dict):
    """
    Cargar los datos de las peliculas del archivo csv.
    """
    sufijo= characteristics["data_size_sufijo"]
    #para la prueba se usan con el sufijo -small
    Amazon_data= cf.data_dir + "Streaming/amazon_prime_titles-utf8"+sufijo+".csv"
    Disney_data= cf.data_dir + "Streaming/disney_plus_titles-utf8"+sufijo+".csv"
    Hulu_data= cf.data_dir + "Streaming/hulu_titles-utf8"+sufijo+".csv"
    Netflix_data= cf.data_dir + "Streaming/netflix_titles-utf8"+sufijo+".csv"

    input_file_Amazon= csv.DictReader(open(Amazon_data, encoding= "utf-8"))
    input_file_Disney= csv.DictReader(open(Disney_data, encoding= "utf-8"))
    input_file_Hulu= csv.DictReader(open(Hulu_data, encoding= "utf-8"))
    input_file_Netflix= csv.DictReader(open(Netflix_data, encoding= "utf-8"))

    addMoviefromCSV_Input(catalog, input_file_Amazon, "amazon")
    addMoviefromCSV_Input(catalog, input_file_Disney, "disney")
    addMoviefromCSV_Input(catalog, input_file_Hulu, "hulu")
    addMoviefromCSV_Input(catalog, input_file_Netflix, "netflix")
```

```
def addMoviefromCSV_Input(catalog, input_file, stream_service:str):
    for video in input_file:
        ya_esta= model.already_exist(catalog["videos"], video)
        if ya_esta == True:
            video["show_id"]=video["show_id"]+"-"+stream_service
            video["stream_service"]= stream_service
        else:
            video["stream_service"]= stream_service
        model.addMovie(catalog, video)
```

- Implementación dentro del modelo:
  Como ya se explicó, se uso como base el laboratorio 4 para hacer este requerimiento, por lo que el funcionamiento es muy parecido al implementado en dicho laboratorio.

- Muestra en pantalla:
  Para mostrar los datos en pantalla y cumplir con el requisito de mostrar los 3 primeros y últimos registros se implementó una función que recibe por parámetro un tamaño de muestra X y retorna los X primeros y últimos registros de la lista. Por último, se muestra los datos con la librería tabulate (se recomienda usar con una pantalla relativamente grande, porque dependiendo del tamaño de la pantalla del computador se puede mostrar la tabla de manera incorrecta).

```
def Get_sample_data(catalog, sample_size:int, list_name:str):
    first_samples=[]
    last_samples=[]
    lista=catalog[list_name]
    list_size=Getlistsize(catalog, list_name)
    if list_size>=(sample_size*2):
        for i in range(1, sample_size+1):
            first_samples.append(lt.getElement(lista, i))
        for e in range(list_size, list_size-sample_size, -1):
            last_samples.append(lt.getElement(lista, e))
    else:
        for i in range(1, list_size+1):
            first_samples.append(lt.getElement(lista, i))

    return first_samples+ list(reversed(last_samples))
```

Entrada	Input numérico del usuario en el menú.
Salidas	Cargar datos en memoria. Una tabla con los primeros y últimos 3 registros
Implementado (Sí/No)	Si. José Guevara

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Añadir registro para cada archivo csv. (Add_movie_from_csv)	O(n)
Ordenar los datos dentro del registro	Depende del algoritmo sort que se use.
Obtener los datos de muestra (primeros y últimos 3) (Get_sample_data)	O (1)
TOTAL	***O(complejidad del algoritmo de ordenamiento que se use)***

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Entrada	Tiempo (s)

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Porcentaje de la muestra [pct]	Tamaño de la muestra (ARRAY_LIST)	Insertion Sort [ms]	Selection Sort [ms]	Shell Sort [ms]	Quick Sort [ms]	Merge Sort [ms]
0.50%	228	3266.21	2834.70	3503.96	2660.08	53.34
5.00%	1148	5228.74	3906.65	4530.40	2984.86	3328.00
10.00%	2298	6133.06	8394.08	5764.58	5089.47	7921.23
20.00%	4598	16214.93	23757.23	9652.39	7048.66	7716.13
30.00%	6898	34432.19	42744.79	17974.98	11167.44	16579.66
50.00%	11498	76870.83	50408.55	28594.14	20984.30	27733.80
80.00%	18397	184683.27	117250.82	51636.46	43285.03	41991.76
100.00%	22998	281306.34	285858.65	68687.06	58578.14	68528.28

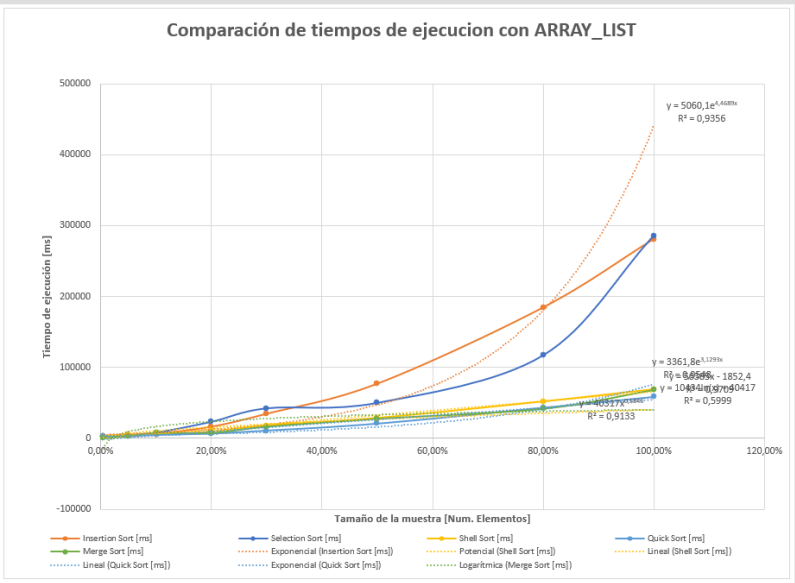
TABLA 1. COMPARACIÓN DE TIEMPOS DE EJECUCIÓN PARA LOS ORDENAMIENTOS ITERATIVOS EN LA REPRESENTACIÓN ARREGLO.

Porcentaje de la muestra [pct]	Tamaño de la muestra (LINKED_LIST)	Insertion Sort [ms]	Selection Sort [ms]	Shell Sort [ms]	Quick Sort [ms]	Merge Sort [ms]
0.50%	228	2991.91	3345.64	6207.82	124.50	80.93
5.00%	1148	53645.94	47554.47	7340.78	1541.20	538.60
10.00%	2298	429885.08	387431.39	17923.16	6582.10	1583.09
20.00%	4598	>10 Min	>10 Min	64325.59	30300.10	4586.40
30.00%	6898	>10 Min	>10 Min	155139.28	74956.90	10376.62
50.00%	11498	>10 Min	>10 Min	412548.86	294385.60	30845.12
80.00%	18397	>10 Min	>10 Min	>10 Min	702395.10	83573.31
100.00%	22998	>10 Min	>10 Min	>10 Min	Más de 10 minutos	123573.23

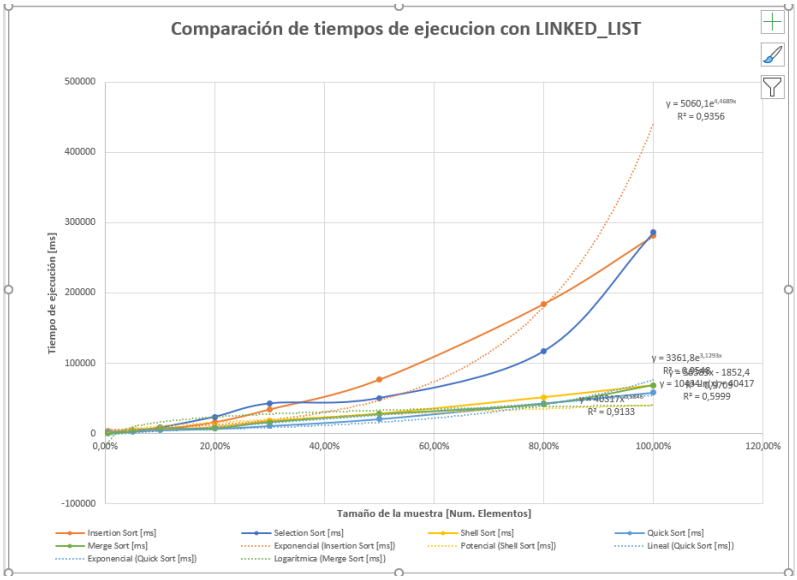
Graficas

Las gráficas con la representación de las pruebas realizadas.

- Comparación de rendimiento ARRAYLIST.



- Comparación de rendimiento LINKED_LIST.



Análisis

Teniendo en cuenta lo visto de los algoritmos de ordenamiento (que en este caso son los que más influyen en el orden de crecimiento), se podría decir que la respuesta está en lo esperado.

Requerimiento <<1>>

Descripción

Se pedía ordenar las películas estrenadas en un periodo de tiempo (entre dos años). Los pasos empleados en este requerimiento son:

- Crear la lista dentro del catálogo:
  En primer lugar, al momento de ejecutar el algoritmo se crea una lista nueva dentro del catálogo la cual se usará para almacenar las películas que cumplan con la condición establecida.

```
def New_list_to_catalog(catalog, List_name:str, TAD:str, cmpfunction):
    "añade o reescribe una nueva lista al catalogo// evita tener que recargar todo el catalogo de peliculas"
    catalog[List_name]=lt.newList(datastructure=TAD, cmpfunction=cmpfunction)
```

  Para esto se implementó una función que crea o sobrescribe una lista dentro del catálogo, esto con el fin de no tener que crear el modelo cada vez que se quiera ejecutar uno de los requerimientos.

- Filtrar registros:
  Se hace un recorrido de todo el catálogo con el objetivo de encontrar cuales registros cumplen con la condición, en este caso, que sean películas y que su año de lanzamiento este entre los dos años pasados por parámetro. Luego, para cada registro valido se crea una copia que solo contendrá los datos que se necesitan mostrar (en este caso: titulo, año de lanzamiento, duración, director, cast y plataforma) esto para no tener que cargar datos innecesarios que solo gastaran más memoria Ram.

```
def Search_movie_by_year(catalog, year1:int, year2:int):
    """
    Se filtran los registros dentro del catalogo y se agregan a una lista los que se encuentren
    entre el rango de años
    """
    New_list_to_catalog(catalog, "movies_by_year", "ARRAY_LIST", compare_videos)
    #recorrer la lista de peliculas
    size_lista_peliculas=lt.size(catalog["videos"])
    for position in range(1,size_lista_peliculas+1):
        video=lt.getElement(catalog["videos"], position)
        #verificar si cumple los requisitos
        if (video["type"]=="Movie")and((int(video["release_year"])>=year1)and(int(video["release_year"])<=year2)):
            #contruir el registro que se va a agregar
            video2={
                "type":video["type"],
                "release_year":video["release_year"],
                "title":video["title"],
                "duration":video["duration"],
                "stream_service": video["stream_service"],
                "director": video["director"],
                "cast":video["cast"]
              }
            lt.addLast(catalog["movies_by_year"], video2)
```

- Mostrar datos en pantalla: se hace una implementación parecida a cuando se cargan los datos para mostrar en pantalla la tabla con los registros y algunas otras variables requeridas.

Entrada	2 enteros: los años entre los cuales se estrenaron las películas
Salidas	Una tabla con los 3 primeros y últimos registros que se encuentren
Implementado (Sí/No)	Si. José Guevara

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Obtener registros que cumplan con las condiciones	O(n)
Ordenar la lista	Depende del algoritmo de ordenamiento
Obtener primeros y últimos registros para mostrar en pantalla (Get_sample_data)	O(n)
Mostrar en pantalla	O (1)
TOTAL	***O(n)***

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Entrada	**Tiempo (ms)**
1999-2021 0.05%	15.02
1999-2021 5%	17.50
1999-2021 10%	29.03
1999-2021 20%	46.61
1999-2021 30%	67.77
1999-2021 50%	110.24
1999-2021 80%	187.19
1999-2021 100%	234.36

Tablas de datos

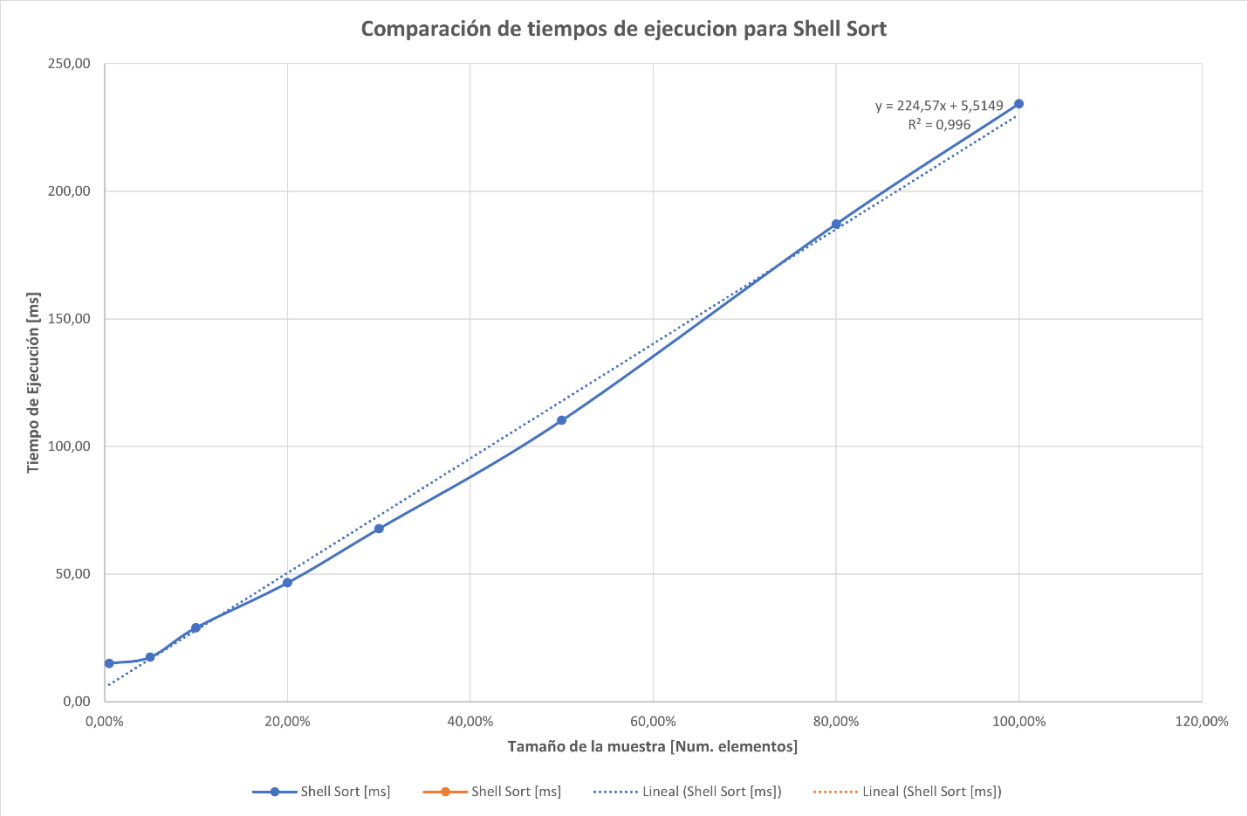
Las tablas con la recopilación de datos de las pruebas.

ARRAY_LIST	**SHELL SORT**	**TIEMPO DE EJECUCIÓN**
0.05%		15.02
5%		17.50
10%		29.03
20%		46.61
30%		67.77
50%		110.24

80%	187.19
100%	234.36

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Como se presencia, la gráfica esta acorde al orden de crecimiento O(n) y aumenta paulatinamente con el tamaño de los datos.

Requerimiento <<2>>

Descripción

Encontrar las series de televisión agregadas en cierto rango de fechas. Los pasos que se siguieron fueron:

- Crear la lista y obtener los registros que cumplan con la condición:
  La implementación de este requerimiento fue muy parecida al anterior, se crea la lista y se recorre todo el catalogo para encontrar los registros que cumplan con la condición. No obstante, en este caso se hace uso de la librería Datetime para transformar los datos que se piden por parámetro y compararlos de manera más fácil.

```
#se crea la lista
New_list_to_catalog(catalog, "tv_shows_by_date", "ARRAY_LIST", compare_videos)
#se transforman los parametros a datetime
initial_date=datetime.strptime(date1, "%B %d, %Y")
final_date=datetime.strptime(date2, "%B %d, %Y")

#se recorre el catalogo de peliculas
size_lista_peliculas=lt.size(catalog["videos"])
for position in range(1,size_lista_peliculas+1):
    tv_show=lt.getElement(catalog["videos"], position)
    if tv_show["date_added"] != "":
        tv_show_date=datetime.strptime(tv_show["date_added"], "%Y-%m-%d")
        #verificar si cumple los requisitos
        if (tv_show["type"]=="TV Show")and(initial_date<=tv_show_date<=final_date):
            show2={
            "type":tv_show["type"],
            "date_added": tv_show["date_added"],
            "title":tv_show["title"],
            "duration":tv_show["duration"],
            "release_year":tv_show["release_year"],
            "stream_service": tv_show["stream_service"],
            "director": tv_show["director"],
            "cast":tv_show["cast"]
          }
            lt.addLast(catalog["tv_shows_by_date"], show2)
```

- Mostrar en pantalla:
  Igual al requerimiento anterior.

Entrada	Dos fechas en formato %B d%, %Y
Salidas	Una tabla con los primeros y últimos 3 registros
Implementado (Sí/No)	Si. Grupo en conjunto

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Recorrer la lista y obtener registros validos	O(n)
Obtener datos de ejemplo que se mostraran en pantalla	O(n)

Mostrar en pantalla	O(1)
TOTAL	***O(n)***

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Entrada	**Tiempo (ms)**
2018 2021 0.05%	138.74
2018-2021 5%	170.23
2018-2021 10%	230.20
2018-2021 20%	270.30
2018-2021 30%	320.40
2018-2021 50%	380.50
2018-2021 80%	440.70
2018-2021 100%	510.10
11th Gen Intel(R)Core (TM) i3-1115G4 @ 3.00GHz 3.00 GHz	
8.0 GB	
Windows 10 Pro - 64 bits	

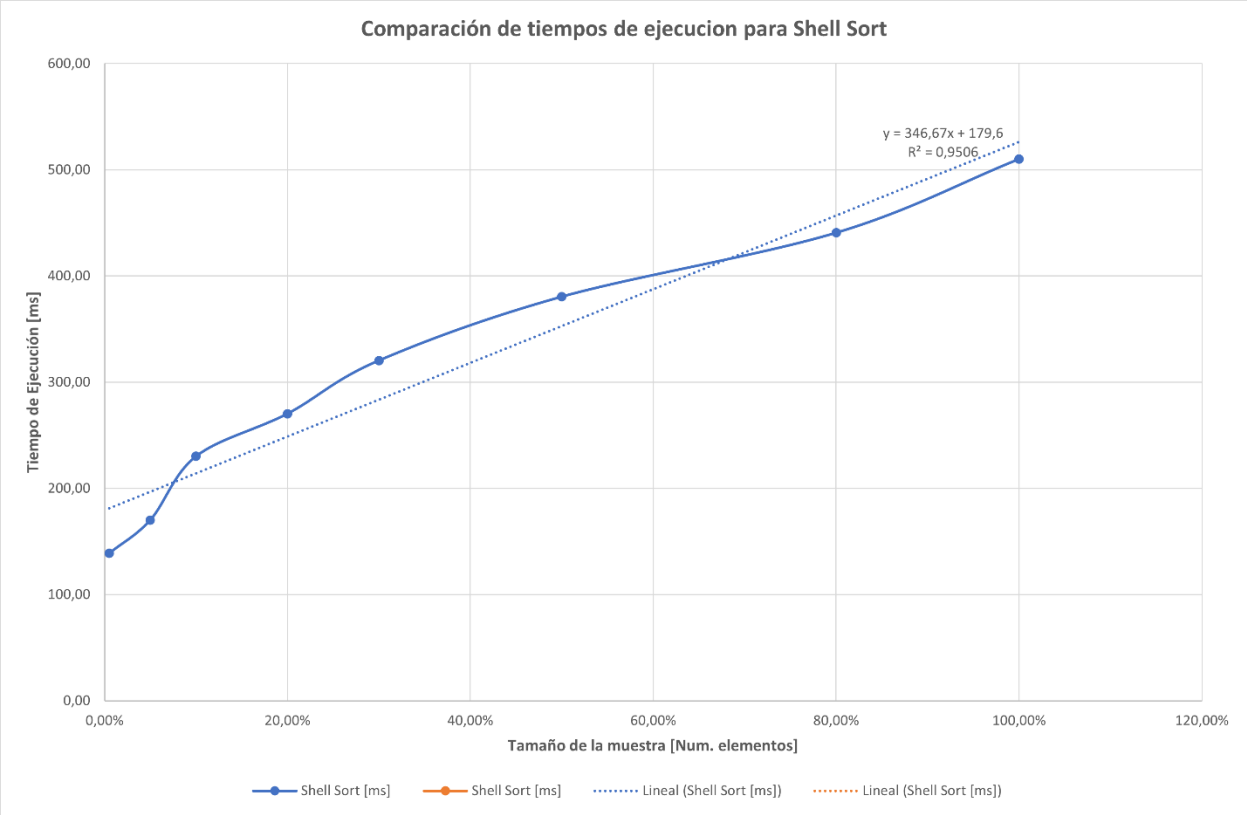
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

ARRAY_LIST	**SHELL SORT**	**TIEMPO DE EJECUCIÓN**
0.05%		138.74
5%		170.23
10%		230.20
20%		270.30
30%		320.40
50%		380.50
80%		440.70
100%		510.10

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

Aunque un poco desviada en algunos casos, se sigue apreciando mantiene un crecimiento casi constante a medida que cambia el volumen de los datos

Requerimiento <<3>>

Descripción

Encontrar contenido en donde participa un actor determinado. Los pasos que se siguieron fueron:

- Recorrer la lista y encontrar los registros que cumplan con los requisitos:

Se hace uso de un bucle for y la unción getItem() para recorrer todo el catálogo de películas. Para cada registro se obtiene la lista de actores y si se encuentra el actor deseado se agrega a la nueva lista.

```
def Search_videos_by_actor(catalog,actor:str):
    New_list_to_catalog(catalog,"videos_by_actor", "ARRAY_LIST", compare_videos)
    #recorrer lista de peliculas
    size_lista_peliculas=lt.size(catalog["videos"])
    videos_by_streaming_service = {}
    for position in range(1,size_lista_peliculas+1):
        video=lt.getElement(catalog["videos"], position)
        #verificar si cumple
        actores=video["cast"]
        separados=actores.split(", ")
        if actor in separados:
            #construir el registro que se va a agregar
            video2={
                "title":video["title"],
                "release_year":video["release_year"],
                "director": video["director"],
                "stream_service": video["stream_service"],
                "duration":video["duration"],
                "cast":video["cast"],
                "country":video["country"],
                "genre":video["listed_in"],
                "description":video["description"]
            }
            #se agrega el video a la lista
            lt.addLast(catalog["videos_by_actor"], video2)
            #agregar al conteo por streaming service
            videos_by_streaming_service[video["type"]] = videos_by_streaming_service.get(video["type"], 0)
            videos_by_streaming_service[video["type"]]+=1
    #se retorna el diccionario con el conteo por genero
    return {"type":videos_by_streaming_service.keys(), "count":videos_by_streaming_service.values()}
```

- Mostrar datos en pantalla:
  Igual que en requerimientos anteriores, se obtienen los primeros y últimos 3 registros ordenados haciendo uso de la función Get_sample_data().

Entrada	Un string que representa el nombre del actor
Salidas	Una tabla en donde se muestren las participaciones del actor buscado.
Implementado (Sí/No)	Si. Daniel Roa Uribe

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Recorrer la lista y obtener registros validos	O(n)
Obtener datos de ejemplo que se mostraran en pantalla	O(n)
Mostrar en pantalla	O(1)

TOTAL	***O(n)***

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Entrada	**Tiempo (s)**
Sissy Spacek	4.97
Se hizo la prueba con 0.05% del tamaño total.	
Computador de Daniel Roa	
Sissy Spacek	
Se hizo la prueba con 100% del tamaño total	46.01
Computador de Daniel Roa	

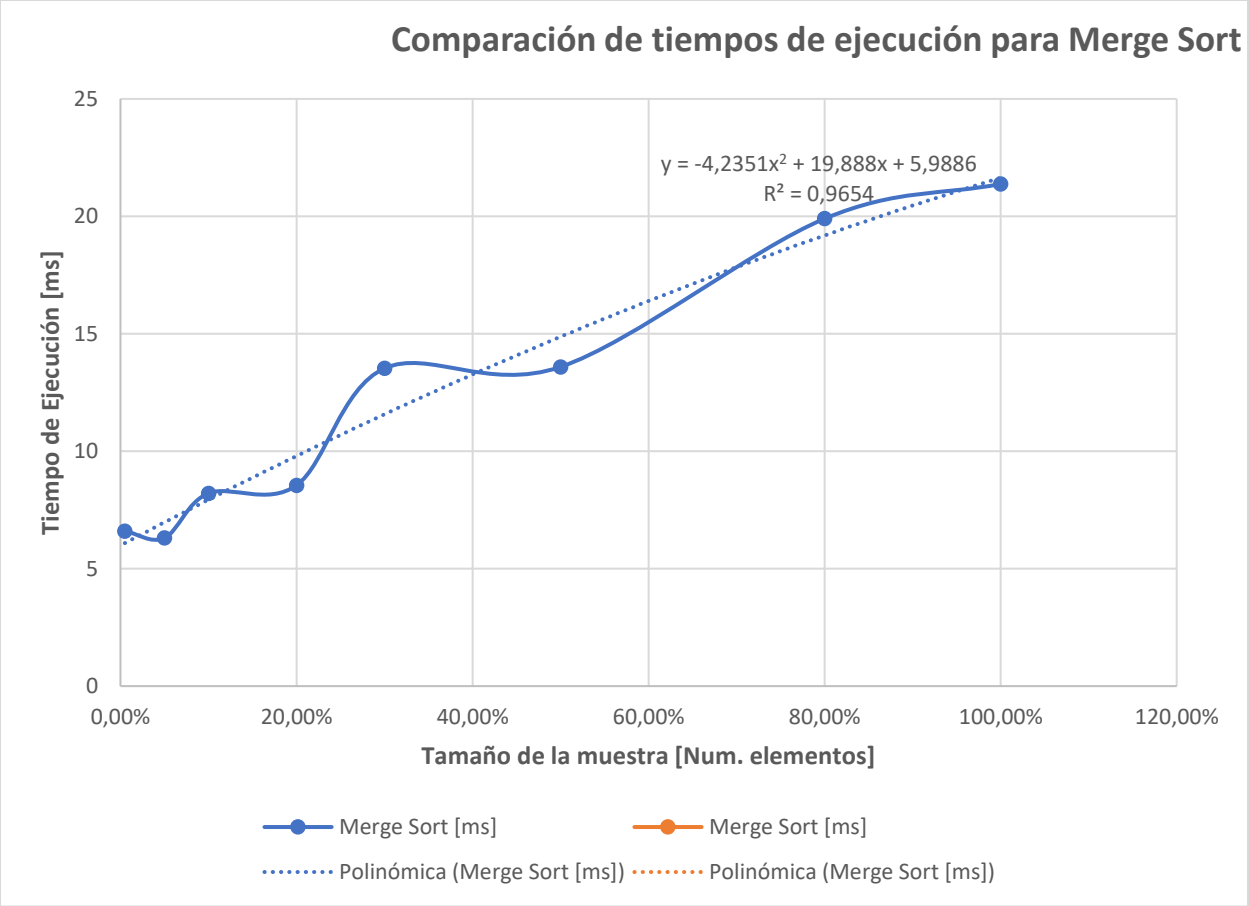
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Entrada	**Merge Sort [ms]**
“Sissy Spacek” 0.05%	4.96
“Sissy Spacek” 5%	5.89
“Sissy Spacek” 10%	7.74
“Sissy Spacek” 20%	9.29
“Sissy Spacek” 30%	22.87
“Sissy Spacek” 50%	31.19
“Sissy Spacek” 80%	37.98
“Sissy Spacek” 100%	46.01

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

Según las pruebas realizada la complejidad temporal de este requerimiento es ~N porque crece linealmente dependiendo del tamaño de la muestra

Requerimiento <<4>>

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Este requerimiento pedía encontrar los registros que estuvieran listados por un género especifico. Los pasos que se siguieron en la implementación fueron:

- Obtener registros que cumplan con las condiciones:
  Al igual que en requerimientos anteriores se recorre la lista y se escogen los registros que en la lista de géneros tengan presente aquel pasado por parámetro.

```
def Contentgender(catalog,gender:str):

    New_list_to_catalog(catalog, "generos", "SINGLE_LINKED", compare_videos)
    size_lista_peliculas=lt.size(catalog["videos"])
    for position in range(1,size_lista_peliculas+1):
        video=lt.getElement(catalog["videos"], position)
        if gender.lower() in video["listed_in"].lower():
            video2={

                "title":video["title"],
                "release_year":video["release_year"],
                "director": video["director"],
                "stream_service": video["stream_service"],
                "duration":video["duration"],
                "cast":video["cast"],
                "country":video["country"],
                "rating":video["rating"],
                "listed_in":video["listed_in"],
                "description":video["description"]
              }
            lt.addLast(catalog["generos"],video2)
```

- Obtener datos de ejemplo y mostrar en pantalla:
  Se aplica la misma lógica usada en los demás requisitos

Entrada	El nombre del género que se va a buscar
Salidas	Una tabla con los datos de los registros encontrados
Implementado (Sí/No)	Si. Jesús Correcha

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Recorrer la lista y obtener registros validos	O(n)

Obtener datos de ejemplo que se mostraran en pantalla	O(n)
Mostrar en pantalla	O(1)
TOTAL	***O(n)***

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Entrada	**Tiempo (s)**
0.50%	0.70
5.00%	1.70
10.00%	3.4
20.00%	6.80
30.00%	8.30
50.00%	27.50
80.00%	63.60
100.00%	52.20

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

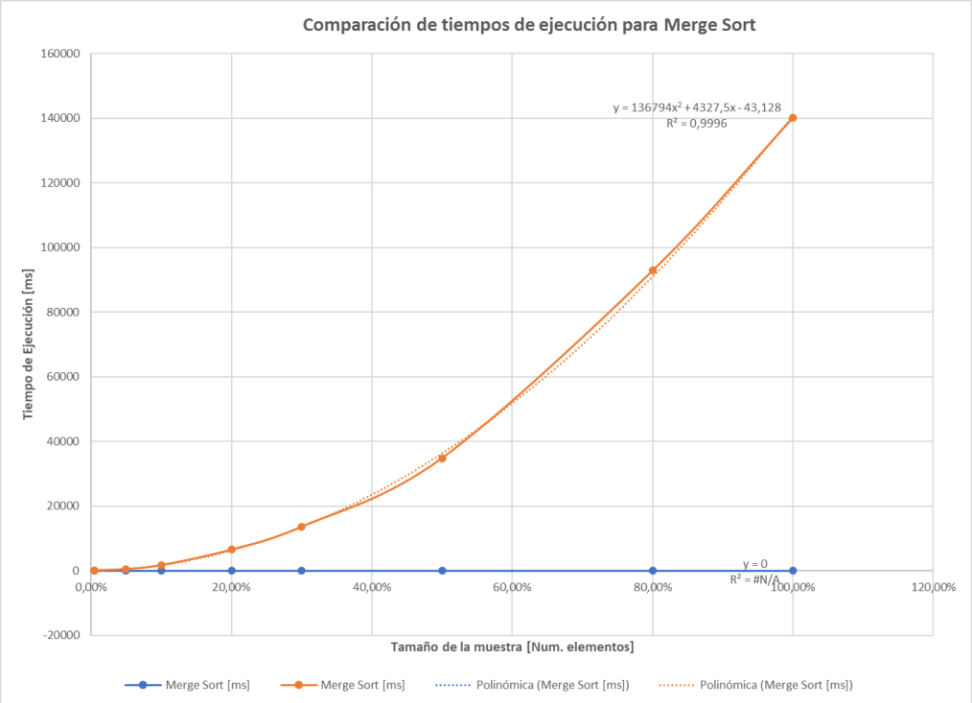
Jesus Correcha

Genero: Fantasy

Porcentaje de la muestra [pct]	**Tamaño de la muestra (ARRAY_LIST)**	**Merge Sort [ms]**
0.50%	228	0.70
5.00%	1148	1.70
10.00%	2298	3.4
20.00%	4598	6.80
30.00%	6898	8.30
50.00%	11498	27.50
80.00%	18397	63.60
100.00%	22998	52.20

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

En este caso la gráfica se desvía un poco del orden de crecimiento esperado, no obstante está cerca del comportamiento ideal.

Requerimiento <<5>>

(Requerimiento individual) Se pedía encontrar contenido producido en un país. Los pasos seguidos para la implementar este requerimiento fueron:

- Obtener los registros que cumplan con los requisitos:
  Al igual que en requerimientos anteriores se recorre toda la lista y se encuentran los registros que cumplan con las condiciones, en este caso que su país de producción sea el pasado por parámetro. Además, para cada registro valido se obtiene el tipo de programa (película o tv_show) y el servicio de streaming. Esto se guarda en un diccionario que se usa en la Vista para mostrar una tabla con el conteo para cada servicio de streaming.

```python
def Search_videos_by_Country(catalog, country:str):
    New_list_to_catalog(catalog, "videos_by_country", "ARRAY_LIST", compare_videos)
    #recorrer la lista de peliculas
    size_lista_peliculas=lt.size(catalog["videos"])
    videos_by_streaming_service={}
    for position in range(1,size_lista_peliculas+1):
        video=lt.getElement(catalog["videos"], position)
        #verificar si cumple los requisitos
        if (video["country"]==country):
            #contruir el registro que se va a agregar
            video2={
                "type":video["type"],
                "release_year":video["release_year"],
                "title":video["title"],
                "duration":video["duration"],
                "stream_service": video["stream_service"],
                "director": video["director"],
                "cast":video["cast"]
              }
            #se agrega el video a la lista
            lt.addLast(catalog["videos_by_country"], video2)
            #agregar al conteo por streaming service
            videos_by_streaming_service[video["type"]]= videos_by_streaming_service.get(video["type"], 0)
            videos_by_streaming_service[video["type"]]+=1
    #se retorna el diccionario con el conteo por streaming service
    return {"type":videos_by_streaming_service.keys(), "count":videos_by_streaming_service.values()}
```

- Obtener datos de ejemplo:
  Se aplica el mismo procedimiento que con requerimientos anteriores.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	El nombre del país del que se quiere buscar contenido (en inglés)
Salidas	Una tabla con las primeras y últimas ocurrencias encontradas.
Implementado (Sí/No)	Si. José Guevara

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Filtrar el catálogo para obtener los registros deseados	O(n)
Obtener datos de ejemplo que se mostraran en pantalla	O(n)
Mostrar en pantalla	O(1)
TOTAL	***O(n)***

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Entrada	Tiempo (s)
0.50%	16.00
5.00%	17.30
10.00%	18.60
20.00%	33.90
30.00%	35.20
50.00%	86.90
80.00%	110.07
100.00%	150.16

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

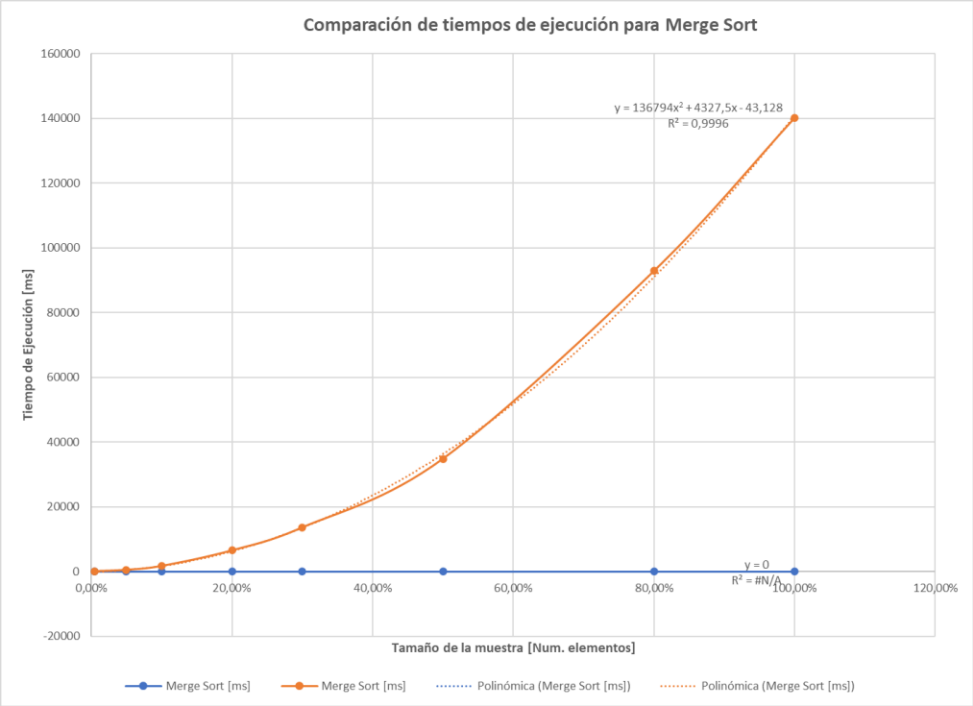
Jesus Correcha

País: United States

Porcentaje de la muestra [pct]	Tamaño de la muestra (ARRAY_LIST)	Merge Sort [ms]
0.50%	228	16.00
5.00%	1148	17.30
10.00%	2298	18.60
20.00%	4598	33.90
30.00%	6898	35.20
50.00%	11498	86.90
80.00%	18397	110.07
100.00%	22998	150.16

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Al igual que en el caso anterior, el comportamiento difiere un poco del ideal, pero sigue teniendo un tiempo de ejecución dentro de lo esperado. El orden de crecimiento ideal es de O(n).

Requerimiento <<6>>

Se requería encontrar las películas y programas de tv teniendo en cuenta al director. Los pasos que se siguieron al implementar este requerimiento fueron:

- Filtrado de datos: se siguió el mismo procedimiento usado en requerimientos anteriores para obtener los registros cuyo director fuera igual al pasado por parámetro. Además, se retornan dos diccionarios que contienen el conteo de ocurrencias tanto por servicio de streaming como para tipo de producción.

```python
for position in range(1,size_lista_peliculas+1):
    video=lt.getElement(catalog["videos"], position)
    #verificar si cumple los requisitos
    if (video["director"].strip()==director):
        #contruir el registro que se va a agregar
        video2={

            "title":video["title"],
            "release_year":video["release_year"],
            "director": video["director"],
            "stream_service": video["stream_service"],
            "type":video["type"],
            "duration":video["duration"],
            "cast":video["cast"],
            "country":video["country"],
            "rating":video["rating"],
            "listed_in":video["listed_in"],
            "description":video["description"]
          }
        #se agrega el video a la lista
        lt.addLast(catalog["director"], video2)
        #agregar al conteo por streaming service
        videos_by_streaming_service[video["stream_service"]]= videos_by_streaming_service.get(video["stream_ser
        videos_by_streaming_service[video["stream_service"]]+=1
        #agregar conteo por tipo
        videos_by_type[video["type"]]= videos_by_type.get(video["type"], 0)
        videos_by_type[video["type"]]+=1
#se retorna el diccionario con el conteo por streaming service
```

- Mostrar en pantalla: sigue los mismos pasos que en requerimientos anteriores.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	El nombre del director
Salidas	Varias tablas con los conteos para los registros asociados al director y los 3 primeros y últimos registros de ejemplo
Implementado (Sí/No)	Si. El grupo en general.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Filtrar el catálogo para obtener los registros deseados	O(n)
Obtener datos de ejemplo que se mostraran en pantalla	O(n)
Mostrar en pantalla	O(1)
TOTAL	***O(n)***

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Entrada	Tiempo (s)
“John Hughes”	6.60
Prueba con 0.05% de los datos	
“John Hughes” Prueba con 100% de los datos	21.38

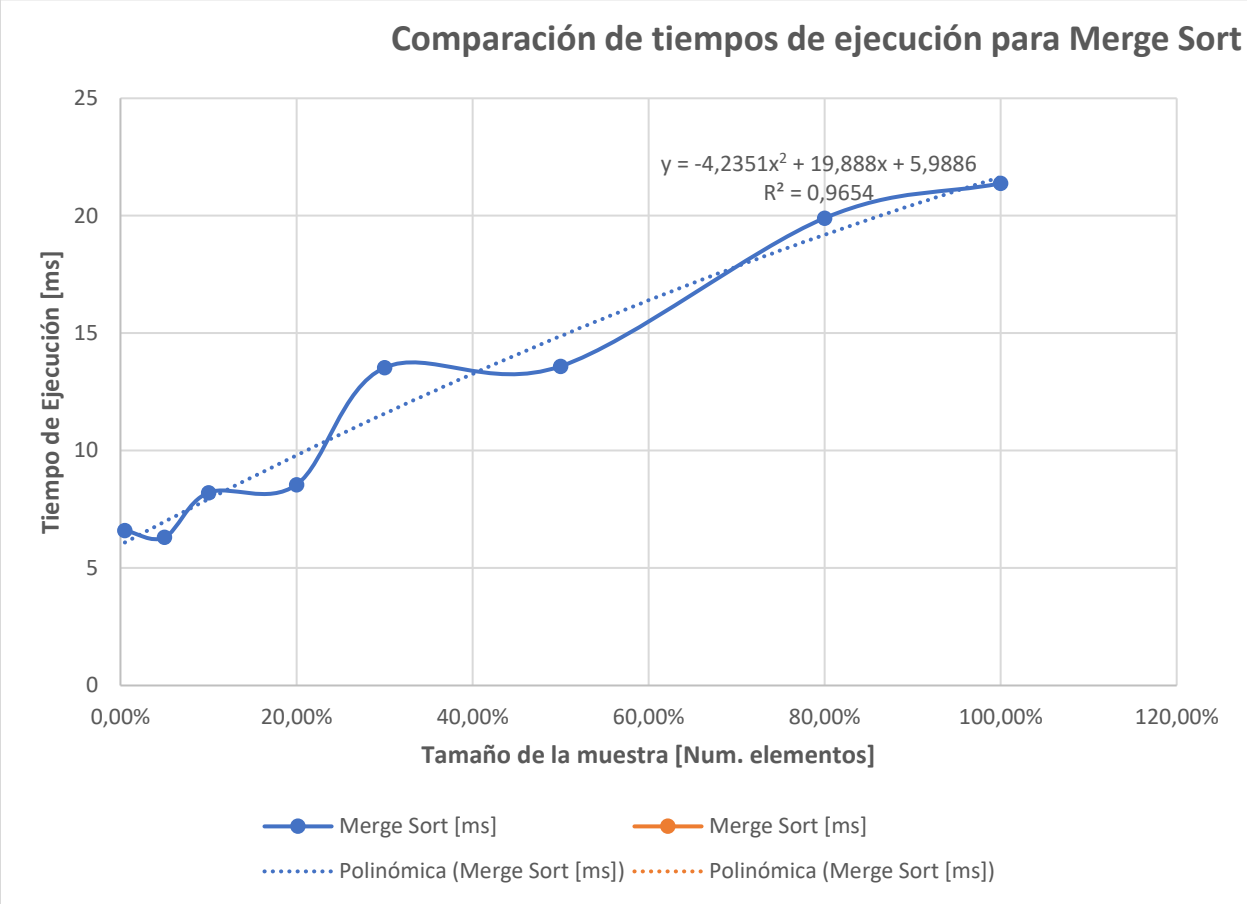
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Entrada	Tiempo (ms)
“John Hughes” 0.05%	6.60
“John Hughes” 5%	6.31
“John Hughes” 10%	8.21
“John Hughes” 20%	8.55
“John Hughes” 30%	13.53
“John Hughes” 50%	13.59
“John Hughes” 80%	19.90
“John Hughes” 100%	21.38

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Dentro de lo esperado.

Requerimiento <<7>>

Descripción

Se pedía obtener el top N de los géneros listados en películas. Los pasos que se siguieron fueron:

- Recorrer todo el catálogo y obtener el conteo de apariciones para cada género:
  Al igual que en los requerimientos anteriores, se hace uso de un bucle y la función getItem() para recorrer cada registro dentro del catálogo. Luego, para cada video se obtienen los géneros en los que este listado y se hace un conteo de estos. Eso se agrega después a un diccionario que lleva control de cuantas veces aparece cada género.

```
#{listed_in:str, count:int, type:str(dict{movie:int, tv_show:int}), stream_service:str(dict{stream:count...})}
registros_list=[]
#diccionario de claves valores, se usa para acceder y modificar algun registro en la lista de registros
#sin tener que recorrerla toda cada vez
claves_valores={}
#recorrer el catalogo de peliculas
size_lista_peliculas=lt.size(catalog["videos"])
for position in range(1,size_lista_peliculas+1):
    video=lt.getElement(catalog["videos"], position)

    #obtener generos listados
    genres=video["listed_in"].split(",")
    n_genders=len(genres)
    for i in range(0, n_genders):
        genres[i] = genres[i].strip()
```

```
#añadir generos al la lista de registros
for gender in genres:
    #se verifica que este genero se encuentre ya en la lista
    #para ello se busca en el diccionario de claves_valores
    indice=claves_valores.get(gender, -1)
    if indice == -1:
        #si no se encuentra
        type_dict={}
        type_dict[video["type"]]=1
        stream_dict={}
        stream_dict[video["stream_service"]]=1
        new_gender={"listed_in":gender, "count":1, "type":type_dict, "stream_service":stream_dict}

        #añdir clave_valor para este genero en la lista
        claves_valores[gender]=len(registros_list)
        #añadir genero a la lista
        registros_list.append(new_gender)
    else:
        #si se encuentra
        # cambiar conteo
        registros_list[indice]["count"]+=1
        #cambiar count de type
        type_count=registros_list[indice]["type"].get(video["type"], 0)
        registros_list[indice]["type"][video["type"]]=type_count+1
        #cambiar count de stream service
        stream_count=registros_list[indice]["stream_service"].get(video["stream_service"], 0)
        registros_list[indice]["stream_service"][video["stream_service"]]=stream_count+1
```

- Obtener datos de ejemplo y mostrar en pantalla:
  Siguen la lógica aplicada a los requerimientos anteriores.

Entrada	Un entero N que va a determinar de cuánto va a ser el top
Salidas	Una tabla ordenada por el ranking para cada genero encontrado y el conteo por tipo y servicio de streaming para dicho genero.
Implementado (Sí/No)	Si. El grupo en general

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Filtrar el catálogo para obtener los registros deseados	O(n*Y), donde Y es el promedio de cuantos géneros son listados en un registro.

Obtener datos de ejemplo que se mostraran en pantalla	O(n)
Mostrar en pantalla	O(1)
TOTAL	***O(n*Y)***

Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Entrada	**Tiempo (ms)**
Top5 0.05%	7,24
Top 5 5%	15,14
Top 5 10%	25,02
Top 5 20%	40
Top 5 30%	45,2
Top 5 50%	54,07
Top 5 80%	63,56
Top 5 100%	85,4
11th Gen Intel(R)Core (TM) i3-1115G4 @ 3.00GHz 3.00 GHz	Especificaciones de la máquina
8.0 GB	
Windows 10 Pro - 64 bits	

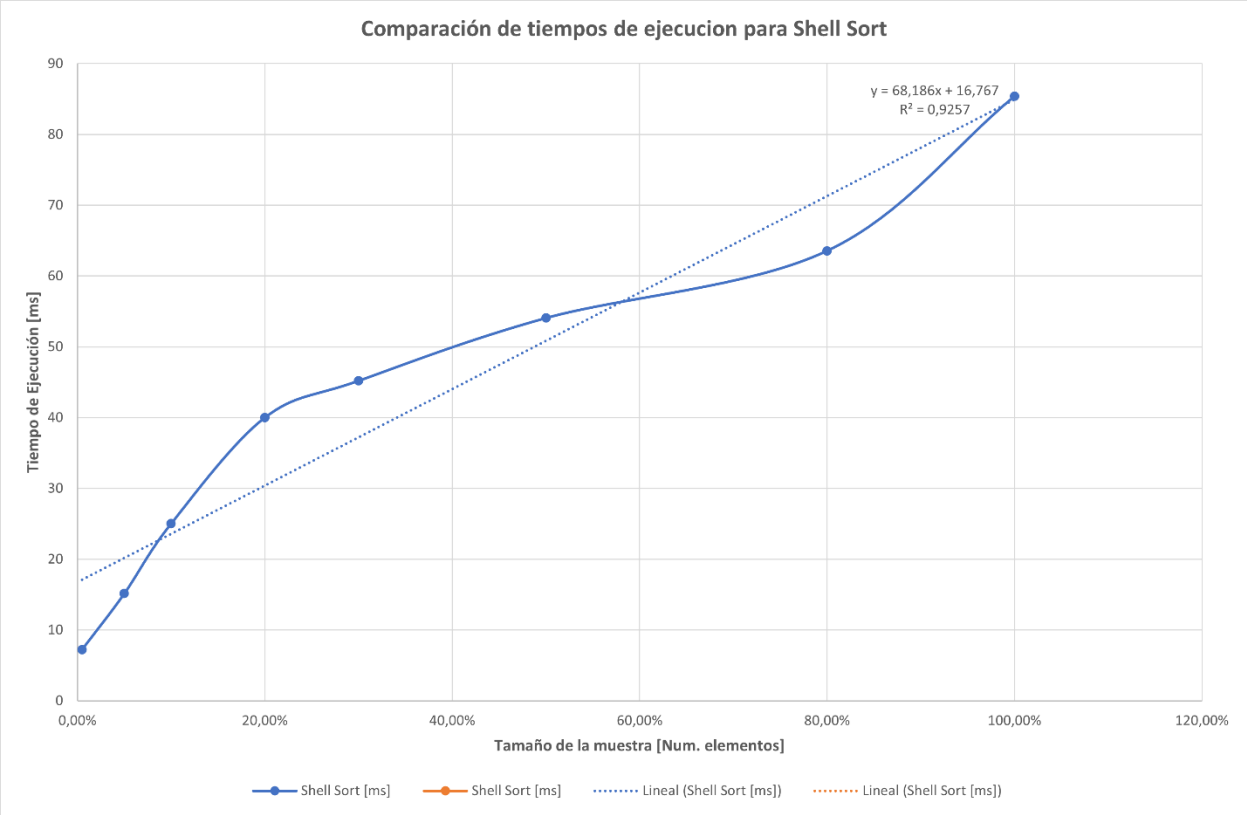
Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

ARRAY_LIST	**SHELL SORT**	**TIEMPO DE EJECUCIÓN**
0.05%		7,24
5%		15,14
10%		25,02
20%		40
30%		45,2
50%		54,07
80%		63,56
100%		85,4

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

Aunque afectado por el rendimiento del computador, se aprecia que la línea de tendencia va acorde con el orden de crecimiento esperado.