Documento de Análisis Reto 1

GRUPO 3:

Requerimiento 3
- Ana Sofía Villa Benavides, 201923361, as.villa@uniandes.edu.co

Requerimiento 4
- Daniela Alejandra Camacho Molano, 202110974, d.camachom@uniandes.edu.co

NA = número artistas
NO= número obras

Análisis de complejidad

Requerimiento 0 (carga de datos)

`initCatalog(estructura)` esta función en view llama a otra del mismo nombre que a su vez llama a la función

- **`newCatalog(estructura):`**
- Esta función es **O (1)**, ya que se genera un catálogo a manera de diccionario con dos llaves, una para artistas y otra para obras. Luego para cada categoría se le crea y asigna una lista vacía con la función de TAD lista newlist(). En este reto decidimos realizar la carga inicial de los datos con ARRAY LIST como estructura ya que en varias ocasiones utilizamos funciones como get.element() sublist() y algoritmos de ordenamiento específicos que tienen una complejidad menor cuando se trata de esta estructura.
-

`loadData(catalog)` esta función en view llama a otra función del mismo nombre en controlador que a su vez llama a dos funciones:

- **`loadArtistas`**Para cada artista en el archivo csv ósea **O(NA)** llama a la función en modelo:
  - **`addArtistb`**Esta función almacena en un diccionario solo la información necesaria de cada artista. A la función solo entra un artista y se accede a su información por llave por lo tanto no hay que recorrer y es **O (1).**
  - Adicionalmente, esta función crea una categoría de obras para cada artista con new.list(ARRAY_LIST) (que es **O (1)**) ya que mantenemos nuestra decisión de utilizar esta estructura, sobre todo porque esta lista de obras para cada artista luego será utilizada con funciones que resultan de menor complejidad con arreglos.
- **`loadObras`** Para cada artista en el archivo csv ósea **O(NO)** llama a la función en modelo:
  - **`def addObra:`** Esta función almacena en un diccionario solo la información necesaria de cada obra. A la función solo entra una obra cuya información se accede por llaves por lo tanto no hay que recorrer y es **O (1).**
  - Adicionalmente, esta función crea una categoría de artistas para cada obra con new.list(ARRAY_LIST) (que es **O (1)**) ya que mantenemos nuestra decisión de utilizar esta estructura, sobre todo porque esta lista de artistas para cada obra luego será utilizada con funciones que resultan de menor complejidad con arreglos.
  - Para realizar la referencia entre artistas y obras en las listas vacías creadas anteriormente, hacemos planteamos una sección de código en la que primero con un for se recorren los artistas de una obra **o(#artitstasenobra)**, asumimos que este número es mucho menor que O(NA) ya que una obra, aunque puede tener varios artistas, según los datos revisados no pasa de 100 artistas por obra. También asumimos que este número entonces es una constante **e** para cada obra. Luego hacemos otro for en el cual se recorren todos los artistas del

catálogo O(NA), para buscar una coincidencia con el código de interés y entonces añadir la obra a la lista de obras del artista y el artista a la lista de artistas de la obra. Esto ultimo se realiza con lt.addlast() que es O(1). La mezcla de estos dos for no alcanza a ser NA^2 ya que **e** es un valor mucho menor, se trataría entonces de una complejidad O(NA*e) ≈ O(NA).

- En total la función **addObras** sería O(NO)*O(NA)= O(NO*NA), sabemos que NO>>NA, entonces O(NO*NA) es mayor que O(NA^2) y menor O(NO^2).

En total, el requisito sería O(NA)+O(NO*NA), como se toma el mayor en la suma sería O(NA*NO)

Requerimiento 1

`sortArtistInDateRange`:

Primero se realizó un Mergesort ya que teóricamente es el algoritmo de ordenamiento con menor complejidad en el peor caso. En este se organiza la lista de artistas por fecha. La complejidad de esto es O (NA Log (NA)). Dentro de esta función se encuentra la función de comparación: "`cmpArtistByDate`", esta tendría complejidad O (1), ya que solo realiza comparaciones mediante las fechas ingresadas.

Por otra parte, se implementó un recorrido (for) en el cual se determina que artistas están dentro del rango determinado por el usuario, esto es O(NA) ya que recorre toda la lista inicial y a partir de esta se crea una nueva lista. Dentro de este for se encuentra la función "AddLast" la cual es O (1), debido a que se está implementando un array list.

Complejidad total= O (NA Log (NA)) + O (1) + O(NA)+ O (1)

Complejidad total= O(NA Log (NA))

`printPrimerosyUltimosartistas.` esta función hace dos subListas con el comando lt.sublist() que es O(1) ya que usamos ARRAY LIST, una lista tiene los primeros 3 elementos y la otra los últimos 3. Luego imprime las dos sublistas usando la función:

- **`printartists(lista,True):`** Esta lista recorre la lista dada, y para cada artista de la lista en este caso de tamaño 3 imprime sus valores. Ósea que la complejidad sería 2 *O (3) ya que son dos sublistas.
- En este caso también se pide imprimir obra, por este motivo se recorren las obras de cada artista y se imprime la información de las obras con la función printobras(lista,false).Esto sería O(E), donde E es un número constante para cada obra.

Complejidad total= O(E)+ O (3) + O (3)

Complejidad total= O(E)

En total, el requisito sería O(NA Log(NA))

Requerimiento 2

Análisis de complejidad:

`sortArtworksandRange(lista,inicial,final):` Esta función recibe la lista (ARRAYLIST) de las obras del catálogo. Luego genera una lista vacía con lt.newlist() que es O(1) donde van a ir las obras que se encuentren en el rango. Para, seleccionar las obras en el rango, se recorre con un *for* cada obra con la función *lt.iterator()* este recorrido es O(NO). Si la fecha de la obra esta dentro del rango, entonces se añade a la lista vacía inicial. Adicionalmente, dentro del mismo recorrido se hace el conteo que toma en cuenta si la obra fue comprada o no en una variable que al final retorna la cantidad de las obras en el rango de tiempo que cumplen esta condición. Luego se aplica ins.sort(ListaEnRango), para obtener una lista ordenada de las obras en el rango. Este algoritmo tiene complejidad O (NO Log(NO)).

La complejidad total de esta función sería O (NO Log (NO)+ O (NO) que por jerarquización quedaría O (NO Log (NO)

`printPrimerosyUltimosobras(lista):` esta función hace dos subListas con el comando lt.sublist() que es O(1) ya que usamos ARRAY LIST, una lista tiene los primeros 3 elementos y la otra los últimos 3. Luego imprime las dos sublistas usando la función:

- **`printobras(lista,True):`** Esta lista recorre la lista dada, y para cada obra de la lista en este caso de tamaño 3 imprime sus valores. Ósea que la complejidad sería 2 *O (3) ya que son dos sublistas.
- En este caso también se pide imprimir artistas, por este motivo se recorren los artistas de cada obra y se imprime la información de estos con la función printartistas(lista,false).Esto sería O(E), donde E es un número constante para cada obra.

Complejidad de esta función O(E)+ O (3) + O (3), ósea O(E)

Complejidad de= O (NO Log (NO)

Requerimiento 3 (Ana Sofía Villa Benavides)

`ObrasPorArtistaPorTecnica(catalog,nombre):` esta función primero, en el peor de los casos recorre todos los artistas **NA** hasta encontrar un artista que coincida con el nombre de entrada, Luego, si encuentra el artista, crea un diccionario vacío para las técnicas y hace otro for en el que recorre las obras de artista puntual (lo que sería **no**), si la técnica de esa obra no está guardada se hace un diccionario que contiene su nombre y una lista vacía (ARRAYLIST) creada con lt.newlist(), luego se agrega la información de la obra a la técnica con addlast() que es O(1), si la técnica ya tiene un diccionario, entonces se agrega simplemente la obra con addlast() que es O(1). Todo este procedimiento sería O(NA) *O(no) = O(no*NA)

`buscarTecnicaMasRep(Tecnicas):` Basado en el diccionario que devuelve la función anterior que tiene las técnicas de una artista, y las obras por técnicas, se hace un recorrido de estas O(NT) un for para hallar la técnica con mas obras. Se usa la función lt.size() que es O(1)

`printobras((Tecnicas[Tecnica]["obras"]),False):` Esta lista recorre la lista de obras dada que corresponde a las obras de cada técnica más usada del artista en cuestión, para cada obra de la lista en este caso de tamaño no imprime sus valores. Ósea que la complejidad sería **O (no)**. Se plantea que no>NA>NO ya que solo corresponde al numero de obras de una técnica en especifico de un artista en específico.

El requerimiento en total sería O (no) + O (no*NA) ≈ O (no*NA) ≈ O (NA)

Requerimiento 4 (Daniela Alejandra Camacho)

Análisis de complejidad:

Requerimiento 5

`OrdenarDepartamentoAsignarPrecioyPeso(catalog,departamento):` Recorre todas las obras ósea que es O(NO), con el fin de comprobar si el departamento coincide con el dado e ir almacenando en una lista, también en el mismo recorrido hace la asignación de precio según otra función y va haciendo la sumatoria de los precios.

- **`def AsignarPrecio(object):`** asigna precio la obra que entra la función de acuerdo a los parámetros dados. Es O(1)

`print5obrasMasCaras(listaporprecio)` **y** **`print5obrasMasAntiguas(listaporfecha),`** aunque estas funciones hacen doble recorrido con for loop, en ninguna de las dos se maneja NO ni Na, en cambio se manejan sublistas pequeña de 5 obras o varios artistas cada obra. No alcanzaría a considerarse O(NO) u O(NA), en ningún caso.

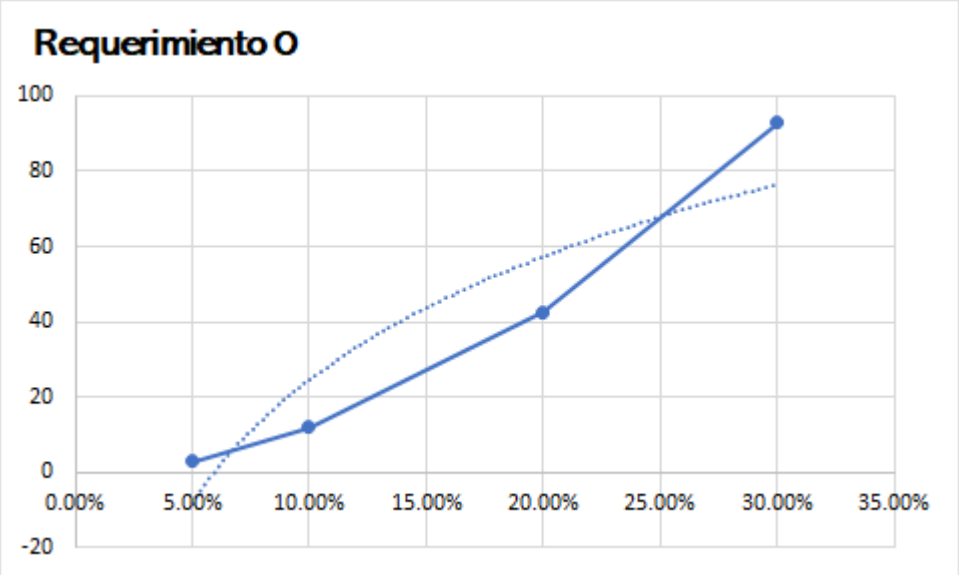
Este requerimiento tiene complejidad O(NO)

Pruebas de tiempos de ejecución
Requerimiento 0

Tabla:

Req 0		
Porcentaje de la muestra [pct]	**Tamaño de la muestra (ARRAYLIST)**	**Tiempo [ms]**
5.00%	6970	3.046875
10.00%	13418	11.859375
20.00%	29227	42.671875
30.00%	44402	92.875

Grafica:



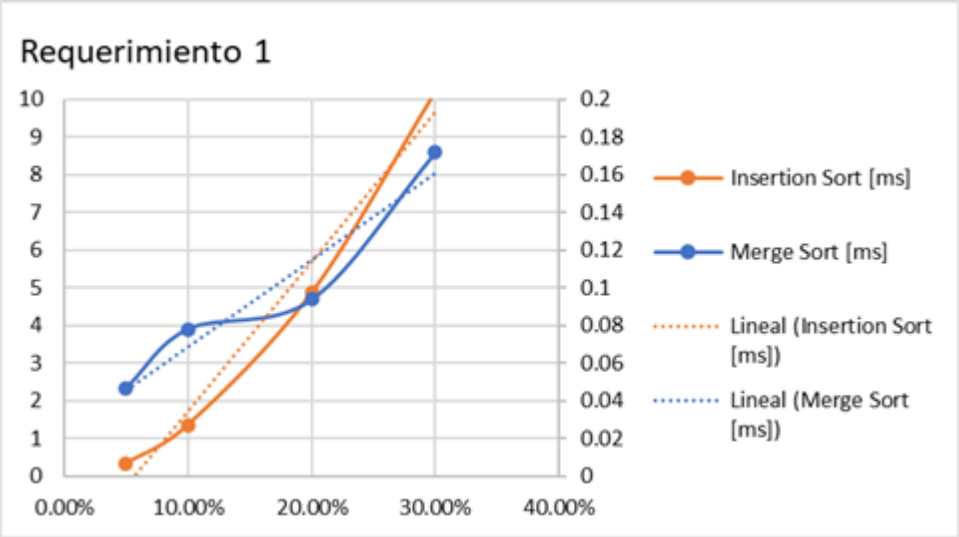
Complejidad según análisis: O(NA*NO) , esto coincide con loq ue se ve en la gráfica.

Requerimiento 1

Tabla:

Req 1			
Porcentaje de la muestra [pct]	**Tamaño de la muestra (ARRAYLIST)**	**Insertion Sort [ms]**	**Merge Sort [ms]**
5.00%	6970	0.328125	0.046875
10.00%	13418	1.359375	0.078125
20.00%	29227	4.875	0.09375
30.00%	44402	10.21875	0.171875

Grafica:



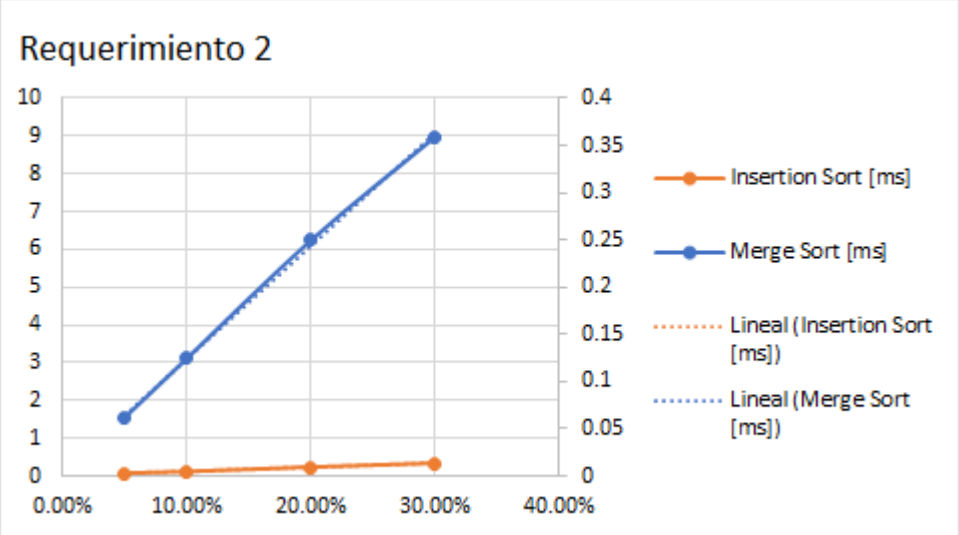
Complejidad según análisis: O(NA Log(NA)) DECIR SI TIENE SENTIDO SEGÚN LA GRÁFICA

Requerimiento 2

Tabla:

Req 2			
Porcentaje de la muestra [pct]	**Tamaño de la muestra (ARRAYLIST)**	**Insertion Sort [ms]**	**Merge Sort [ms]**
5.00%	6970	0.0625	0.0625
10.00%	13418	0.109375	0.125
20.00%	29227	0.234375	0.25
30.00%	44402	0.34375	0.359375

Grafica:



Complejidad según análisis: O(NO Log(NO)) DECIR SI TIENE SENTIDO SEGÚN LA GRÁFICA

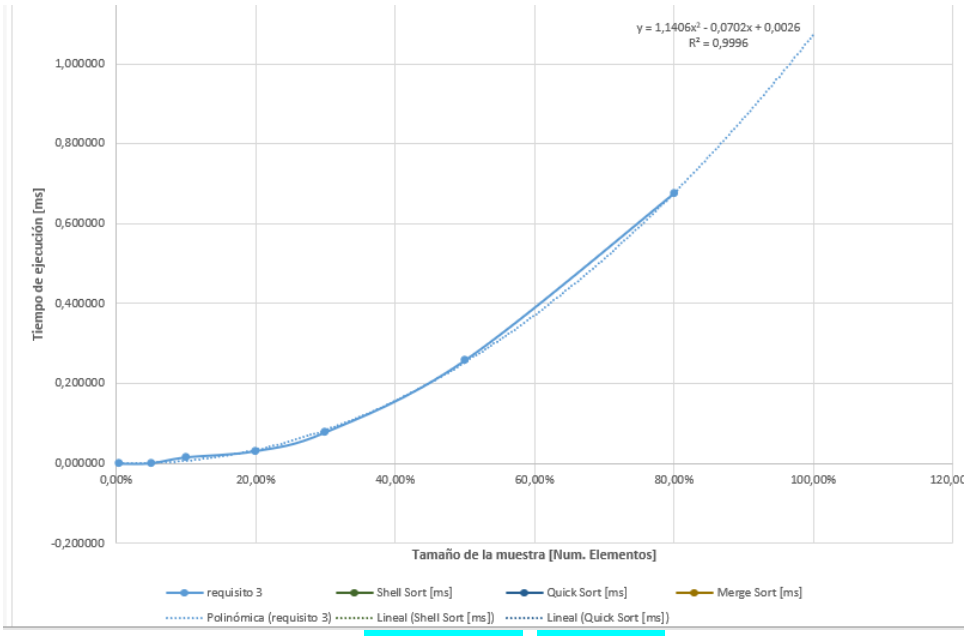
Requerimiento 3 (Ana Sofía Villa Benavides)

Tabla:

Porcentaje de la muestra [pct]	requisito 3
0,50%	0,000000
5,00%	0,000900
10,00%	0,015625

20,00%	0,031250
30,00%	0,077850
50,00%	0,257650
80,00%	0,675430
100,00%	

Grafica:



Complejidad según análisis: ≈ O (no*NA)> O (NA^2) tiene sentido ya que es una polinómica parecida a cómo sería n^2 pero con una pendiente menor.

Requerimiento 4 (Daniela Alejandra Camacho)

Tabla:

Grafica:

Complejidad según análisis:

Requerimiento 5

Tabla:

Grafica:

Complejidad según análisis: