Documento de Análisis Reto 2

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
M= tamaño tabla

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 principales, una para artistas y otra para obras. Luego para cada categoría se le crean subdiccionarios que contienen los diferentes mapas vacíos con los cuales organizamos tanto artistas como obras.

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

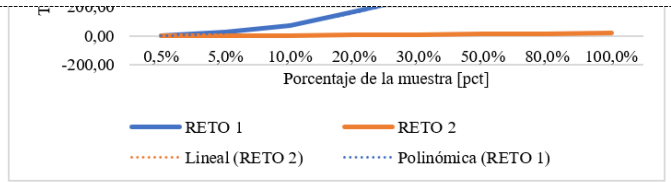
- **`loadArtistas`**Para cada artista en el archivo csv ósea **O(NA)** llama a la función en modelo:
  - **`addArtistb` O (1)** Esta función almacena en un diccionario solo la información necesaria de cada artista. A la función solo entra un artista y se accede a su información por llave por lo tanto no hay que recorrer.
  - Adicionalmente, esta función crea una categoría de obras para cada artista con new.list(ARRAY_LIST) ya que mantenemos nuestra decisión de utilizar esta estructura.
  - Además, por cada artista se crea un mapa vacío que luego va a almacenar sus obras ordenadas por medio como índice.
  - Luego se añade la información del artista a cada mapa planteado O (1)
- **`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, para poder hacer la conexión entre artista y obra se accede a la info de cada artista por el mapa que tiene id como índice lo que resulta O (1)

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

Pruebas de tiempos de ejecución:

En primer lugar, comparamos los mecanismos de linear probing y separate chaining, obtuvimos valores y complejidades muy similares. Sin embargo, durante el laboratorio que realizamos sobre este tema pudimos experimentar mas con estos mecanismos a la hora de aplicar otros requerimientos. A partir de esto y de que la complejidad es un poco menor, escogimos utilizar chaining con un factor de carga =4.

20,0%	163,98438	5,59
30,0%	268,68750	7,67
50,0%		10,97
80,0%		16,26
100,0%		19,62



Requerimiento 1

Análisis de complejidad:

`sortArtistInDateRange`:

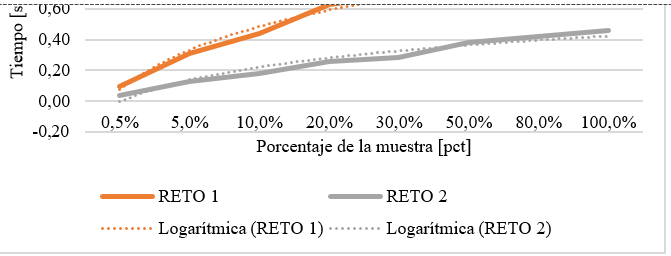
Esta función primero obtiene las llaves del mapa de artistas ordenados por año de nacimiento. Para hacer esto se recorre la tabla O(M), sin embargo, vale la pena ya que hay menos llaves de años que la totalidad de los artistas.

Luego, con un recorrido se van agregando a una lista las obras que se encuentran en las parejas de llave-valor del mapa que se encuentren dentro del rango dado. Esto sería un recorrido de las llaves O(keys). En el caso del archivo large se tiene una variedad de 236 llaves de años. Por lo tanto, en el peor de los casos sería O(CTE) donde 236 es mucho menor que el total de artistas que es alrededor de 15mil.

Para ordenar esta pequeña lista obtenida que corresponde a la información dentro del rango, se utilizó merge sort que sería O(nlogn) donde n es mucho menor a NA ya que es el número de artistas en el rango y no el total.

TOTAL: O(nlogn)

5,0%	0,31250	0,12460
10,0%	0,43750	0,17900
20,0%	0,62500	0,25500
30,0%	0,68760	0,28640
50,0%		0,38375
80,0%		0,42180
100,0%		0,45833



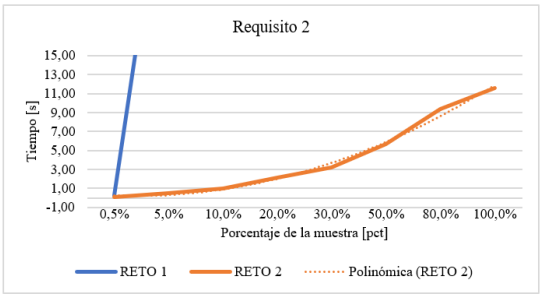
o

Requerimiento 2

Análisis de complejidad:

`sortArtworksandRange(lista,inicial,final):` primero, por medio del mapa de las fechas se sacan todas las fechas por medio de las llaves de esta. A partir de estas se verifica si está en el rango especificado por le usuario y a partir de esto se crea una nueva lista en donde se agrega la información (el valor de la llave según la fecha), luego se usa merge sort para ordenar la lista (esta ya está dentro del rango especificado). Es por esto que su complejidad es: o (nlogn), debido a que la complejidad más alta es esta, ya que, al usar el mapa, solo se debe recorrer las llaves y a partir de estas acceder directamente su información

100,0%		11,61719



El Reto 1 se había planteado una complejidad de O(NO Log(NO)) que en realidad resultó siendo cercana a NO^2 de acuerdo a su comportamiento con la gráfica. En cambio, en este reto se plantea una complejidad de O(nlogn) que es menor. Esto se puede ver en la gráfica ya que la pendiente del RETO 2 es mucho menor..

Requerimiento 3 (Ana Sofía Villa Benavides)

Análisis de complejidad:

`ObrasPorArtistaPorTecnica(catalog,nombre):`

En esta función únicamente se accede a valores de mapas, por lo tanto resulta ser O(1). Primero se accede a la información del artista usando su nombre como llave, luego dentro de la información del artista se accede a la información de las técnicas ya que desde la carga los datos de ordenaron así. Además para poder tener el número de obras total sin recorrer el mapa de técnicas, simplemente se saca el size del array lista con todas las obras de ese artista.

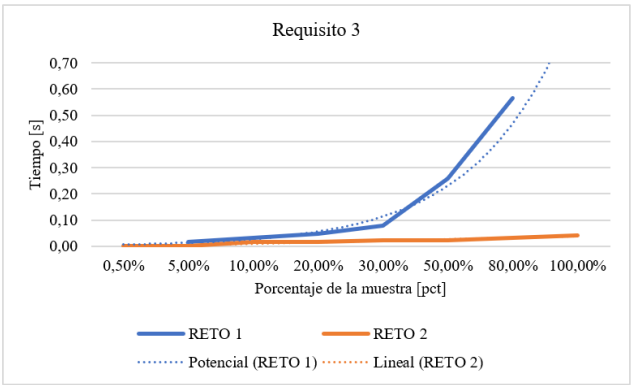
`buscarTecnicaMasRep(Tecnicas):`

Esta función recorre el mapa de las técnicas del artista puntual encontrado para encontrar cual es la técnica culla lista de obras es mas grande. Este proceso sería **O(m)** donde m es el tamaño de este mapa relativamente pequeña que solo contiene las obras de este artista.

Total = **O(m)**

Pruebas de tiempos de ejecución:

Req 3		
Porcentaje de la muestra [pct]	**Tiempo [s]**	
	RETO 1	**RETO 2**
0,50%		0,00000
5,00%	0,01563	0,00001
10,00%	0,03125	0,01563
20,00%	0,04688	0,01563
30,00%	0,07813	0,02340
50,00%	0,25765	0,02343
80,00%	0,56543	0,03125
100,00%		0,04164



El Reto 1 se había planteado una complejidad de ≈ O (no*NA) que se asemeja a una complejidad potencial, en cambio en este reto se plantea una complejidad de O(m). La complejidad teórica del reto 2 debe ser mucho menor ya que m se trata de las técnicas de un solo artista mientras que no*NA es aún mayor que el total de artistas. Esto coincide con lo visto en la gráfica ya que las pendientes son significativamente diferentes.

Requerimiento 4 (Daniela Alejandra Camacho)

Análisis de complejidad:

`RankingCountriesByArtworks`

Pruebas de tiempos de ejecución:

El Reto 1 se había planteado una complejidad de *****, en cambio en este reto se plantea una complejidad de O(*). Esto se puede ver en la gráfica.....

Requerimiento 5

Análisis de complejidad:

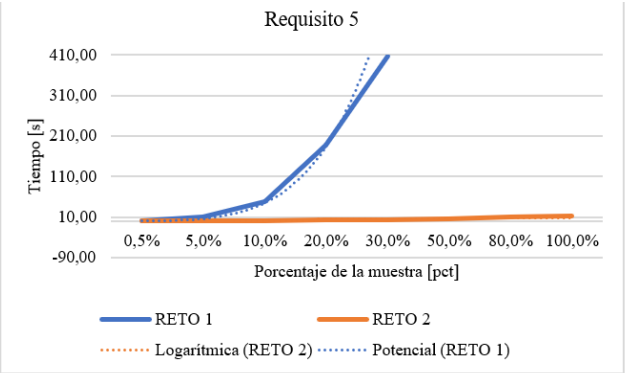
`OrdenarDepartamentoAsignarPrecioyPeso(catalog,departamento):` accede al valor de la pareja llave valor donde la llave es el departamento. Esto se logra ya que en la carga de datos se planteó un mapa con este índice. Acceder a este valor es O(1)

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

`SortArtworksByPrice y sortArtworksByDate:` Ordena las obras del departamento dado (que fueron seleccionadas con la función anterior) ambas usan el algoritmo de merge sort, por lo tanto tienen una complejidad de O(n log n) donde n corresponde al número de obras de un solo departamento.

Pruebas de tiempos de ejecución:

Req 5		
Porcentaje de la muestra [pct]	Tiempo [s]	
	RETO 1	RETO 2
0,5%	0,14063	0,04688
5,0%	10,75000	0,48374
10,0%	47,96875	0,94680
20,0%	187,71875	2,05300
30,0%	407,09375	3,15580
50,0%		5,34375
80,0%		8,94000
100,0%		11,50000



El Reto 1 se había planteado una complejidad de O(NO log NO), que en la gráfica se aproxima más a un n^2, en cambio en este reto se plantea una complejidad de O(n log n) donde n corresponde al número de obras de un solo departamento. En este reto obtenemos un procedimiento con complejidad temporal mucho menor que el anterior ya que no hay que recorrer todas las obras para seleccionar las del departamento dado, esto se puede ver claramente en la tabla y gráfica.