Reto 2: Documento de análisis

Link: https://github.com/EDA2021-2-SEC02-G11/Reto2-G11

Integrantes del grupo

Grupo 11: **Federico Melo Barrero.**
202021525.
f.melo@uniandes.edu.co
Implementa el requerimiento individual 3.

Juan Camilo Prieto Avella.
201814815.
jc.prietoa@uniandes.edu.co
Implementa el requerimiento individual 4.

Análisis de complejidad de los requerimientos

Requerimiento 1

Para el requerimiento uno, se usa una tabla de hash en la cual cada llave es un año y el valor que le corresponde es una lista (concretamente un arreglo, ARRAYLIST) de los artistas cuya fecha de nacimiento fue en aquel año. Cada uno de los artistas es a su vez un diccionario.
Cuando se ejecuta el requerimiento, se pide al usuario un año inicial **initial_year** y un año final **final_year** en el archivo **view.py**. En el archivo **model.py** se realiza un **get** para cada año del rango en la tabla de hash, lo cual tiene una complejidad de O(1) por año, es decir O(n) en el peor de los casos.

- Se obtiene el número total de artistas en el rango sumando los tamaños de cada arreglo de artistas, que se obtienen en O(1) con la la operación **size** del API del TAD lista.
- Se obtienen los primeros tres artistas y los últimos tres artistas con la función **requirement1**, que está construida para iniciar desde el principio y parar después de que se miran los primeros tres y luego empezar desde el final y parar después de que se miran los tres últimos (de forma que jamás recorre más de 6 artistas).

Nótese que aunque la complejidad temporal del requerimiento es O(n) en el peor de los casos, si se quisiera buscar un rango de 5 años, la complejidad es de O(1) para cada año, lo que equivale a únicamente 5 operaciones, un valor constante.

Esto es más rápido que la forma en la que se implementó el requerimiento en el reto anterior, pues era necesario ordenar la lista y luego buscar el rango. La menor complejidad para ordenar la lista se obtenía usando el algoritmo **mergesort** y era de O(n log (n)) y obtener los elementos del rango presenta una complejidad de O(n) en el peor de los casos (que el rango contenga a todos los años).

En el archivo **view.py**, se usa la función **printReq1()** para mostrar los resultados del requerimiento uno en una tabla, hecha con la librería **prettytable**. Esta función no interviene en el funcionamiento del requerimiento.

Requerimiento 2

Para el requerimiento dos, se usa una tabla de hash en la cual cada llave es un año y su valor correspondiente es una lista, concretamente una arrego, con las obras cuya fecha de adquisición **'DateAquired'** fue en ese año.

- Se obtiene el número total de obras en el rango sumando los tamaños de cada arreglo de obras, que se obtienen en O(1) con la operación **size** del API del TAD lista, al igual que se hace en el requerimiento 1.
- Se obtienen las primeras tres obras y las últimas tres obras con la función r**equirement2**, que está construida de manera muy similar a la del requerimiento 1: inicia desde el principio y para después de que se miran los primeros tres y luego va desde el final y para después de que se miran los tres últimos (de forma que jamás recorre más de 6 obras).

En el archivo **view.py**, se usa la función **printReq2()** para mostrar los resultados del requerimiento dos en una tabla, hecha con la librería **prettytable**. Esta función no interviene en el funcionamiento del requerimiento.

Requerimiento 3

Implementado por Federico Melo Barrero.

Para el requerimiento tres se usa una tabla de hash en la cual cada llave es el nombre de un artista, su **DisplayName**. El valor que corresponde a esa llave es una estructura que tiene:

- El número total de obras del artista.
- El número total de técnicas usadas por el artista.
- El nombre de la técnica más usada.
- Cuántas veces usa la técnica más utilizada.
- Una tabla de hash por medios, en donde cada una de las llaves es un medio o técnica y su valor es un arreglo con las obras relizadas con ese medio (y por el artista que es llave de la tabla de hash grande).

Esa estructura permite coger los datos que se tienen que rertornar en aproximadamente O(1), pues se hace un **get()** para el artista y los valores son los datos que se buscan. Para la lista de obras, se hace un **get()** al arreglo.

En el archivo **view.py**, se usa la función **printReq3()** para mostrar los resultados del requerimiento tres en una tabla, hecha con la librería **prettytable**. Esta función no interviene en el funcionamiento del requerimiento.

Requerimiento 4

Implementado por Juan Camilo Prieto Avella.

Para el requerimiento cuatro se usa una tabla de hash en la que las llaves son la nacionalidad de la obra y el valor es un array list que contiene todas las obras, luego aplicaremos un array que contiene y va guardando el top 10 de nacionalidades segun se va llenando la tabla de hash de nacionalidades.

En el archivo **view.py**, se usa la función **printReq4()** para mostrar los resultados del requerimiento cuatro en una tabla, hecha con la librería **prettytable**. Esta función no interviene en el funcionamiento del requerimiento.

Requerimiento 5

Para el requerimiento cinco se hace uso de una tabla de hash en donde cada llave es un departamento del MoMA y el valor que le corresponde es un arreglo con las obras que corresponden a ese departamento. Nótese que hacer un **get()** para tener el arreglo con todas las obras de un departamento es aproximadamente O(1).

Se usa una función adicional que calcula el costo de transportar cada una de las obras y su peso. Se realizan los dos cálculos de forma independiente por cada obra (esto es inevitable).

- El total de obras para transportar se obtiene con el tamaño del arreglo haciendo uso de la operación **size** del API del TAD lista.
- El precio estimado en USD del servicio se obtiene sumando los precios individuales para transportar cada una de las obras de arte.
- El peso estimado de las obras se obtiene sumando los pesos individuales obtenidos para cada obra.

En el archivo **view.py**, se usa la función **printReq5()** para mostrar los resultados del requerimiento cinco en una tabla, hecha con la librería **prettytable**. Esta función no interviene en el funcionamiento del requerimiento.

Pruebas de tiempos de ejecución de los requerimientos

Máquina utilizada:

Reto 1:

	Máquina
Procesadores	1,6 GHz Intel Core i5 de dos núcleos,
Memoria RAM (GB)	8.00 GB.
Sistema Operativo	MacOS Big Sur 64-bits

Reto 2:

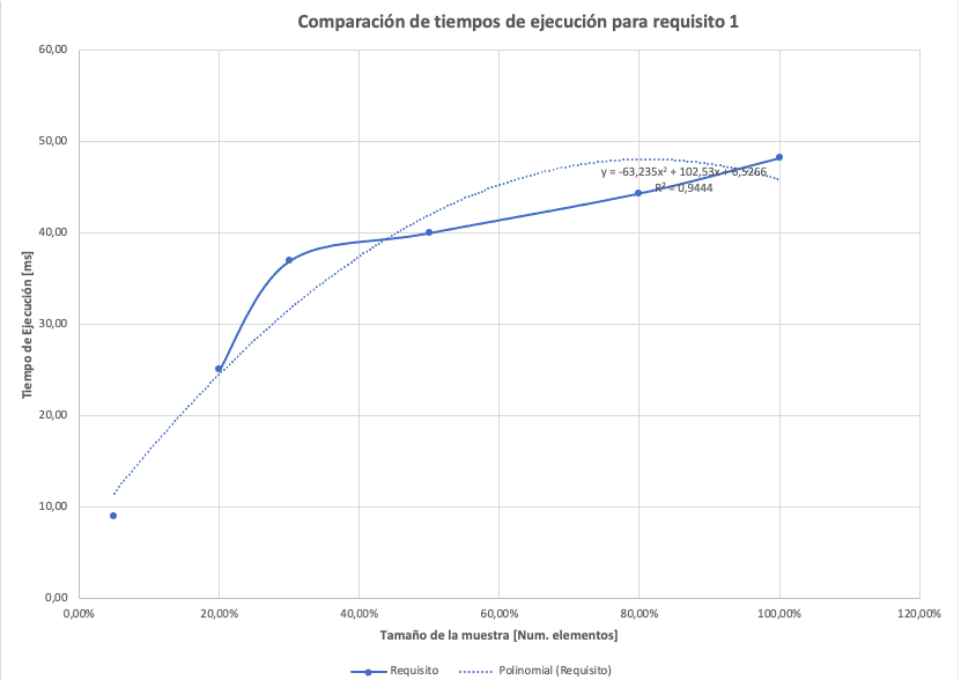
	Máquina
Procesadores	Intel® Core™ i5-6200U CPU @ 2.30GHz × 4
Memoria RAM (GB)	3.70 GB.
Sistema Operativo	Linux, Ubuntu 20.04.3 LTS

Requerimiento 1:

Requerimiento 1, reto 1

Porcentaje de la muestra [pct]	Merge Sort [ms]
small	8.907
20%	25.056
30%	36.948
50%	40.008
80%	44.347
100.00%	12.1327

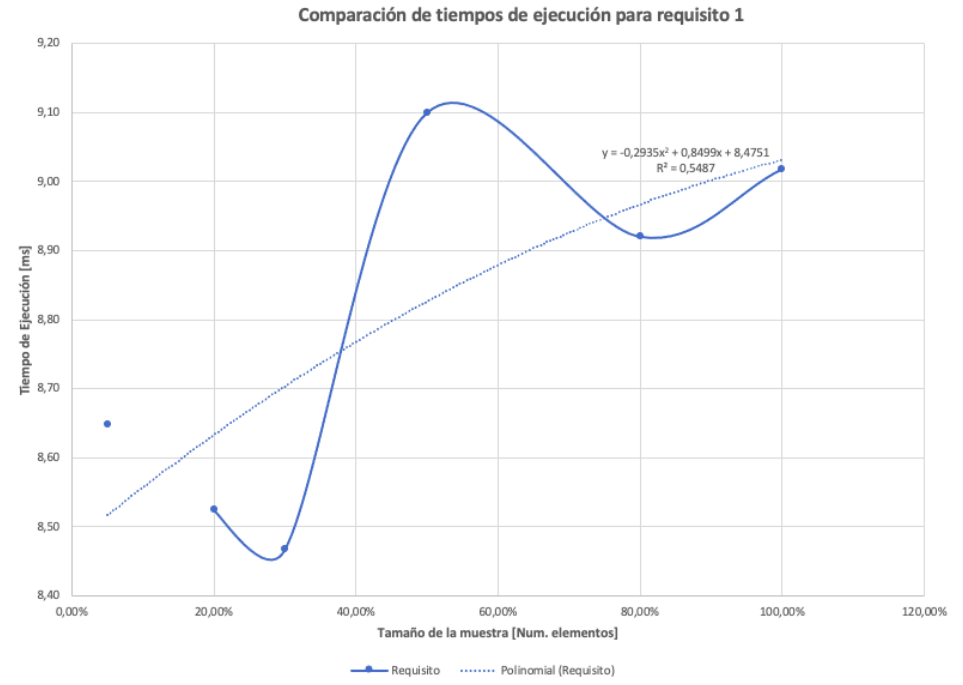
Gráfica:



Requerimiento 1, reto 2

Porcentaje de la muestra [pct]	Número de artistas	Demora [ms]
small	1948	8.6488
20%	8724	8.5244
30%	10063	8.4673
50%	12137	9.0998
80%	14143	8.9203
100.00%	15223	9.0185

Gráfica:



Requerimiento 2

Requerimiento 2, reto 1

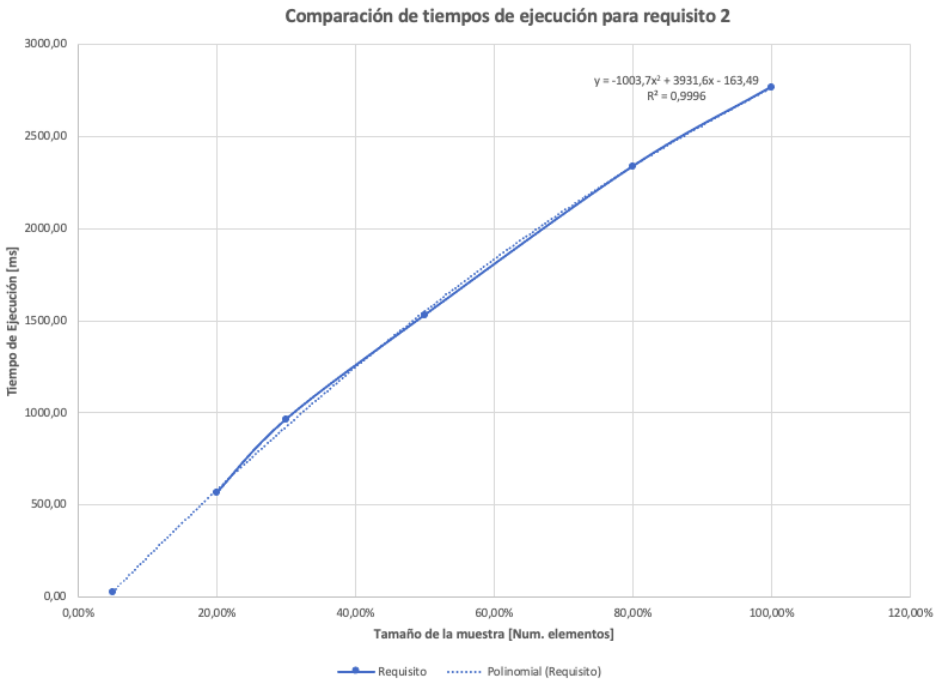
Porcentaje de la muestra [pct]	Merge Sort [ms]
small	26.932
20%	565.22
30%	965.828
50%	1530.945
80%	2337.218
100.00%	2768.036

Gráfica:

Requerimiento 2, reto 2

Porcentaje de la muestra [pct]	Número de obras	Demora [ms]
small	768	27.7895
20%	29489	61.4830
30%	43704	80.5871
50%	71432	93.6373
80%	111781	128.2081
100.00%	138150	152.4710

Gráfica:



Requerimiento 3

Requerimiento 3, reto 1

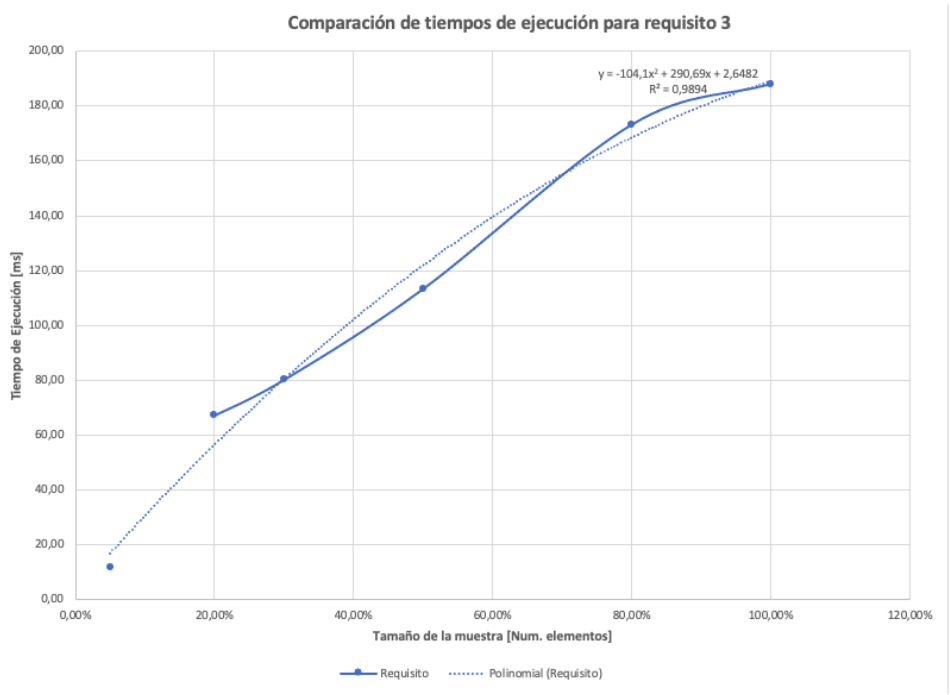
Porcentaje de la muestra [pct]	Merge Sort [ms]
small	3.079
20%	16.822
30%	19.924
50%	28.089
80%	45.849
100.00%	49.198

Gráfica:

Requerimiento 3, reto 2

Porcentaje de la muestra [pct]	Número de artistas	Número de obras	Demora [ms]
small	1948	768	11.6611
20%	8724	29489	67.2750
30%	10063	43704	80.2654
50%	12137	71432	113.3889
80%	14143	111781	173.2196
100.00%	15223	138150	188.0300

Gráfica:



Requerimiento 4

Requerimiento 4, reto 1

Porcentaje de la muestra [pct]	Merge Sort [ms]
small	8.537
20%	8.568
30%	7.359
50%	7.665
80%	9.480
100.00%	7.263

Gráfica:

Requerimiento 4, reto 2

Porcentaje de la muestra [pct]	Número de artistas	Número de obras	Demora [ms]
small	1948	768	
20%	8724	29489	
30%	10063	43704	
50%	12137	71432	
80%	14143	111781	
100.00%	15223	138150	

Gráfica:

Requerimiento 5

Requerimiento 5, reto 1

Porcentaje de la muestra [pct]	Merge Sort [ms
small	22.690
20%	18783.318
30%	44077.052

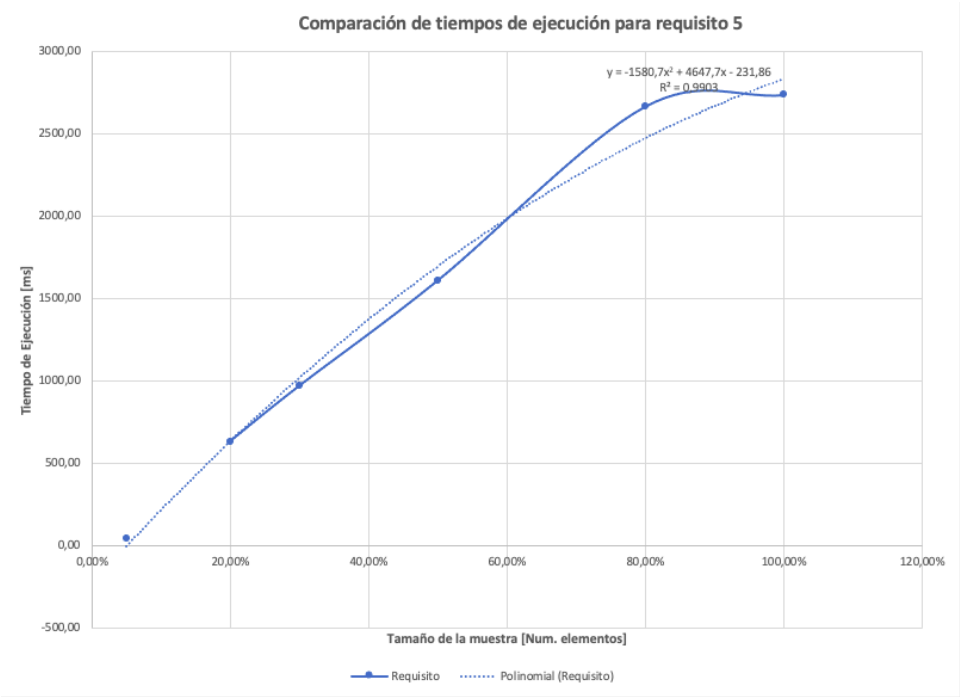
50%	136853.936
80%	328783.518
100.00%	509095.672

Gráfica:

Requerimiento 5, reto 2

Porcentaje de la muestra [pct]	Número de artistas	Número de obras	Demora [ms]
small	1948	768	41.6980
20%	8724	29489	631.8684
30%	10063	43704	969.3307
50%	12137	71432	1609.4605
80%	14143	111781	2667.1462
100.00%	15223	138150	2738.2595

Gráfica:


Comparación de tiempos de ejecución para requisito 5
y = -1580,7x² + 4647,7x - 231,86
R² = 0,9903
3000,00
2500,00
2000,00
1500,00
1000,00
500,00
0,00
-500,00
0,00%
20,00%
40,00%
60,00%
80,00%
100,00%
120,00%
Tiempo de Ejecución [ms]
Tamaño de la muestra [Num. elementos]
Requisito
Polinomial (Requisito)