Estructuras de Datos y Algoritmos – Reto 2 – Grupo 13 Sección 2
Análisis de complejidad de los requerimientos

Req 3 - Juan Sebastián Núñez Cortes, 202021672, j.nunezc@uniandes.edu.co
Req 4 - Santiago Rodríguez Bernal, 202011182, s.rodriguez63@uniandes.edu.co

A continuación, se presenta un análisis de complejidad para cada requerimiento del Reto 2, considerando el código utilizado y las graficas que se obtuvieron a partir de las pruebas de tiempos. Las graficas junto con los tiempos se encuentran en el archivo 'Tablas Reto 2'.

Porcentaje de la muestra [pct]	Tamaño de la muestra (ARRAYLIST)	Requerimiento 1 [ms]	Requerimiento 2 [ms]	Requerimiento 3 [ms]	Requerimiento 4 [ms]	Requerimiento 5 [ms]	Requerimiento 6 [ms]	Carga de datos (1) [ms]
0,5% (small)	2716,00	15,62	31,25	0,00	0.0	46,88	46,88	1171,88
5,00%	12571,00	15,62	46,88	15,62	0.0	343,75	203,12	26468,75
10,00%	21664,00	15,62	46,88	0,00	0.0	750,00	265,62	75546,88
20,00%	38213,00	0,00	62,50	15,62	0.0	1734,38	375,00	193437,50
30,00%	53767,00	0,00	62,50	0,00	15.62	2876,24	425,89	335437,50
50,00%	83569,00	31,25	62,50	15,62	15.62	4390,62	531,25	666984,38
80,00%	125924,00	15,62	78,12	31,25	15.62	6984,38	640,63	1166515,62
100% (large)	153373,00	15,62	78,12	15,62	0.0	9000,00	687,50	1559812,50

Tabla 1. Resumen de los tiempos de ejecución del Reto 2

Carga de datos: La carga de datos se ocupa de añadir cada obra y cada artista al catalogo del MoMA. Adicionalmente, se crean cinco mapas para poder ser consultados al momento de probar los requerimientos. Asimismo, hay un sexto mapa dentro de uno de estos cinco índices. En general, si bien la toma de datos requiere más tiempo que en el anterior Reto, esto posibilitó que se redujeran considerablemente las complejidades y los tiempos de carga de todos los requerimientos.

Requerimiento 1: Para el requerimiento 1 se utilizó principalmente dos funciones. La primera es getAuthorsByDate, la cual obtiene las obras de un mapa dado un rango determinado por dos fechas. Al examinar el código, las asignaciones son constantes, y el for utilizado se emplea n veces, con un contador de m veces. La complejidad de esto es de O(n). Luego se utiliza la función sortArtists para ordenar por mergesort, por lo cual su complejidad es de O(nLogn). Asimismo, esta corresponde a la complejidad del requerimiento. Si bien es la misma que en el Reto 1, dado que se acceden a valores que ya están guardados, esta es mucho más rápida y eficiente.

Requerimiento 2: Para el requerimiento 2 se empleó dos funciones. La primera es artworksRange, que al igual que el requerimiento 1, obtiene las obras de un mapa dado un rango determinado por dos fechas (año, mes y día). En este caso, nuevamente se realiza un for sobre las llaves para obtener dicho rango, por lo cual la complejidad es de O(n). Acto seguido se ordena el rango bajo la función sortArtworks, y dado que se emplea mergesort, la complejidad general del requerimiento es de O(nLogn). Nuevamente, dado que se acceden a valores previamente cargados en el catalogo, es mucho más eficiente que el Reto 1.

Requerimiento 3: Para el requerimiento 3, se utilizaron dos funciones. La primera es artistMedium, la cual al revisar las técnicas de un artista en el mapa de un catalogo, calcula cual es la técnica más utilizada. Dado que se examinan n llaves, el for produce una complejidad de O(n). Por otro lado, la segunda función getArtworksByMedium, retorna todas las obras de una técnica. Debido a que estas obras están guardadas como una lista de una llave de un mapa, acceder a este valor es de una complejidad constante. Por lo tanto, la complejidad del código utilizado es de O(n). Con respecto al Reto 1, se evidencia que es más eficiente, más no por mucho dado que en el Reto 1 esta función ya era bastante rápida.

Requerimiento 4: Para el requerimiento 4 primero se creo una función llamada ‘‘nationalityArtistsinArtwork’’ (en la carga de datos) que extrae el ‘‘ConstituentID’’ de la obra para compararla con el ID único de cada artista. De esta manera la función extrae la nacionalidad, pues encuentra el artista asociado a la obra. La complejidad de esta función es de O(n), pues utiliza loops para ir comparando con cada artista si el ‘‘ConstitutentID’’ de la obra y su ID son iguales. Por ultimo la función ‘‘addArtworkNationality’’, se encarga de tomar estos datos entregados por la primera función ya mencionada y agrega los datos al mapa de nacionalidades, donde la llave corresponde a la nacionalidad y el valor las obras asociadas a esta nacionalidad, en este caso la complejidad es O(n) ya que tiene que ir comparando por cada una de las obras y agregarla a la nacionalidad correspondiente.

Requerimiento 5: Para el requerimiento 5, se emplearon 3 funciones. La primera artworksDepartment retorna las obras de un departamento, el costo del departamento, y los kilogramos a transportar. Dado que, a diferencia del Reto 1, ya se tiene las obras de cada departamento en un mapa, se obtienen dichas obras en un tiempo constante y se itera n veces sobre ella para hacer los cálculos respectivos, es decir la complejidad es de O(n). En segundo y tercer lugar, se utiliza las funciones sortDateArtworks y sortCostArtworks para ordenar las obras del departamento por costo y fecha respectivamente. Para ambos se utiliza mergesort, por lo cual la complejidad del código para este requerimiento es de O(nLogn), la cual es más eficiente que el O(n^2) que se había obtenido en el Reto 1.

Bono – Requerimiento 6: Para este último requerimiento, se utilizó 4 funciones. La primera es getArtistByDate, la cual retorna una lista con los artistas que nacieron en un rango. Se accede a esta información en el catalogo, y se itera sobre ella n veces en las llaves del mapa, por lo cual la complejidad es O(n). La segunda función getprolificArtist, retorna una lista ordenada en base a los criterios del artista más prolífico. Primero, se agrega la información del artista que le corresponde basado en su cantidad de obras y medios utlizados. Segundo, en base a esta información, ordena con mergesort la lista, la complejidad es de O(nLogn). La tercera función es artistMedium, que se ocupa de identificar la técnica más utilizada de un artista y de obtener sus obras. Al igual que en el requerimiento 3, la complejidad es de O(n). Por último, la función getArtworksByMedium es una consulta sobre un mapa, con complejidad constante. Así pues, la complejidad del código es de O(nLogn).