Análisis del Reto 1

Estudiante 1 Sergio Rincón Cod 201914107
Estudiante 2 Luis Tejón Cod 202113150

El estudiante 1 (Sergio Rincón) desarrolló el requerimiento 3.
El estudiante 2 (Luis Tejón) desarrolló el requerimiento 4.

Observaciones generales:

- En general se prefirió disminuir complejidad en las consultas a pesar de que se pudiera aumentar complejidad en la carga de datos, esto porque la carga de datos se realizaría solo una vez, por lo que una complejidad extra es justificable si luego disminuye el tiempo a la hora de hacer las consultas.

Análisis complejidad de los requerimientos:

- Requerimiento 1:
  Tipo de TAD usado: TAD lista (array_list)
  Carga de datos: Para este requerimiento se agregó en la carga inicial de datos un ordenamiento tipo Merge para crear una lista de autores ordenada por fecha de nacimiento.
  Complejidad agregada a la carga de datos: O(nlogn) con n el número de artistas.
  Requerimiento: En este requerimiento se emplearon dos búsquedas binarias, teniendo en cuenta que permiten encontrar el límite superior e inferior del intervalo en la lista que contiene los artistas dentro de los años especificados por entrada. A partir de esto, a la complejidad de este requerimiento se le agrega 2logn. Ahora bien, la cantidad de artistas se determina a partir de los límites y, como sólo es necesario imprimir 6 obras, la complejidad total es de O(logn)
  Complejidad de la consulta: O(logn)

- Requerimiento 2:
  Tipo de TAD usado: TAD lista (array_list)
  Carga de datos: Para este requerimiento se agregó en la carga inicial de datos un ordenamiento de tipo Merge para crear una sub lista de las obras, pero ordenada por la fecha de adquisición.
  Complejidad agregada a la carga de datos: $O(nlogn)$ con n el número de obras.
  Requerimiento: En el requerimiento al ya estar organizada la lista, hizo falta únicamente el uso de 2 búsquedas binarias para hallar las posiciones de la menor obra con la fecha inicial (o la menor obra con una fecha inmediatamente mayor) y la posición de la mayor obra con la fecha final (o la inmediatamente menor) con lo que, al ser de tipo array se pudo “cortar” la lista a

partir de esas posiciones en O (1). Al final en la lista resultante se hizo una iteración para ver cuántas obras son adquiridas por compra la cual tiene una complejidad de n.
Complejidad de la consulta: $tilda(2logn + m)$ o $O(n)$ con en n el número de obras totales en el catálogo y m el número de obras en el rango de fechas dado. Nótese que la complejidad es O(n) ya que en el peor de los casos n= m y en ese caso n crece mucho más rápido que 2logn por lo que podemos despreciar este término

- Requerimiento 3:
- **Tipo de TAD usado:** TAD lista (array_list)
- **Carga de datos:** Para este requerimiento se generó un diccionario de artistas que contenía listas de obras organizadas por técnicas. Para lograr esto, se leyó el archivo de artistas y el archivo de obras, así como se organizó mediante Merge sort cada sublista.
- **Complejidad agregada a la carga de datos:** $O(nlogn)$
- **Requerimiento:** La decisión de implementar un diccionario de artistas con listas de obras organizadas por técnicas se tomó debido a la dificultad de crear una lista dentro de otra lista y luego reemplazarla. Sabiendo esto, se procedió a usar el diccionario, teniendo en cuenta también que la implementación de este en Python se hace mediante tablas de Hash, por lo que favorece la eficiencia del programa. Contando con esto, la consulta funciona de la siguiente manera, el nombre ingresado se busca en una lista de Tags mediante el algoritmo de búsqueda binaria, obteniendo el ID del autor. Este ID posteriormente se usa como llave de un diccionario que tiene la técnica más usada y las obras del autor organizadas por técnicas. Con esta información, se usa de nuevo la búsqueda binaria con la técnica más usada, obteniendo las posiciones límites en la lista de obras entre las cuales se encuentran las de la técnica. Con esto, se devuelve la cantidad de obras de dicha técnica junto con los demás datos del requerimiento.
- **Complejidad de la consulta:** En promedio $(logn)$ pero en el peor de los casos (sólo hay un autor y todas sus obras son de sólo una técnica) $O(n)$

- Requerimiento 4:
- **Tipo de TAD usado:** TAD lista (array_list)
- **Carga de datos:** Para este requerimiento al cargar los datos de los artistas se hizo organizándolos con respecto a su 'ConstituentID' usando nuevamente un Merge, luego se iteró sobre las obras, con lo que obra por obra se buscaban los ID de sus artistas en la lista de artistas haciendo uso de una búsqueda binaria, y se agregaba la obra a una lista de obras de esa nacionalidad.
- **Complejidad agregada a la carga de datos:** $tilda(mlogm + 2nlogm)$ o $O(nlogm)$ con n el número de obras y m el número de artistas.
- **Requerimiento:** En este requerimiento simplemente se consulta el catálogo y se imprime los resultados.
- **Complejidad de la consulta:** $O(1)$

- Requerimiento 5:
  Tipo de TAD usado: TAD lista (array_list)

Carga de datos: En este caso, la carga de datos implica crear diccionarios de los departamentos llenándolos con listas de obras en orden cronológico, además de guardar las obras cuyo transporte es más costos y el precio total.
Complejidad agregada a la carga de datos: $O(nlogn)$
Requerimiento: El requerimiento se responde mediante los datos guardados en el diccionario durante la carga de datos junto con las 5 primeras posiciones de la lista organizada por antigüedad, por lo que se puede decir que la complejidad tiende a ser $O(1)$. La respuesta funciona recibiendo el nombre del departamento, que se usa como llave en un diccionario que contiene la información necesaria para dar la respuesta además de la lista de obras
Complejidad de la consulta: $O(1)$

- Requerimiento 6:
  Tipo de TAD usado: TAD lista (array_list)
  Carga de datos: Para esta función en la carga de datos se creó una lista de obras de dos dimensiones, en la que se iteró sobre la lista de obras verificando que la obra tuviera fecha, ancho y alto.
  Complejidad agregada a la carga de datos: $tilda(3n)$ o $O(n)$ con n el número de obras
  Requerimiento: En el requerimiento se iteró sobre la lista de obras de dos dimensiones, comparando su fecha con las fecha inicial y final del rango, si la obra se encontraba en el rango de fechas se agregaba a una lista de la que luego se leerían los datos. Por último, en la lista de obras en ese rango de fechas, se iteró para ir sumando las áreas de cada obra hasta superar el área dada por parámetro.
  Complejidad de la consulta: $tilda(2n + m)$ con n el número de obras en 2D y m el número de obras en ese rango de fechas dado por parámetro o $O(j)$ con j el número total de datos (obras y artistas).

Ambientes de pruebas

	Máquina 1	**Máquina 2**
Procesadores	AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz	Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 2.61GHz
Memoria RAM (GB)	16GB	12GB
Sistema Operativo	Windows 10 home 64-bits	Windows 10 home 64-bits

Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

Maquina 1 Resultados

Archivo “large”

Carga de Datos	103.1875 seg
Req. 1	0.0 ms
Req. 2	0.0 ms
Req. 3	0.0 ms
Req. 4	0.0 ms
Req. 5	78.125
Req. 6	0.0 ms

Tiempo de carga en segundos de los requerimientos en la máquina 1.

Maquina 2 Resultados

	Archivo “large”
Carga de Datos	105.71875 sec
Req. 1	15.625 ms
Req. 2	0.0 ms
Req. 3	0.0 ms
Req. 4	0.0 ms
Req. 5	46.875 ms
Req. 6	0.0 ms

Tiempo de carga en segundos de los requerimientos en la máquina 2.