Camilo Ortiz Cruz 201821615 [c.ortizc@uniandes.edu.co](mailto:c.ortizc@uniandes.edu.co)

Kevin Fernando Gómez Camargo 202015120 [k.gomezc@uniandes.edu.co](mailto:k.gomezc@uniandes.edu.co)

Análisis de complejidad

Requerimiento 1:

A la hora de implementar el requerimiento 1 se tuvieron 2 opciones, para ambas se partió de ordenar desde un inicio el catálogo de acuerdo con la necesidad de búsqueda.

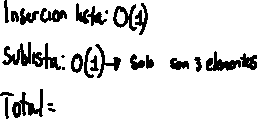
Opción 1:

Se pensó recorrer la lista hasta encontrar el primer valor que fuera mayor o igual que el limite inferior y seguir hasta que el valor se vuelva mayor que el límite superior, cuando los valores estén dentro del limite superior e inferior se realizarían las operaciones necesarias del requerimiento, que son sumar el numero total de artistas, y guardar los primero 3 encontrados, adicionalmente se mantiene un dato de cuál es la posición máxima de los valores dentro del rango, al finalizar esto se crea una sublista de las ultimas 3 posiciones del rango.

Opción 2:

Encontrar la posición mínima haciendo una búsqueda de ceil() y a partir de esto buscar hasta el límite máximo, internamente ir sumando el número de artistas, agregar los 3 primero e ir buscando la máxima posición del rango, Finalmente crear una sublista de las ultimas 3 posiciones del rango.

Complejidad:



De las dos opciones se escogió la 2nda, esto se debe a que al ser log(n) + m y n el valor de log(n) en casi todos los casos será menor que m, la complejidad queda en o(m) y debido a esto en los casos promedio como en el del ejemplo del reto que habían 545 artistas en el rango de años, la opción 2 tendrá un m = 545 mientras que la opción 1 tendrá que recorrer desde 1 hasta la última posición del límite superior, el rango dado se encuentra casi al final del arreglo de artistas, tendrá que recorrer casi 15000.

Pruebas de tiempo:

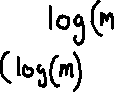
Requerimiento 2.

Para el requerimiento 2 se planteó solo se planteó una forma de hacerlo y esta es:

1. Buscar la pos del ceil de la fecha inicial
2. Recorrer desde pos del ceil hasta la pos del máximo valor dentro del límite superior
3. Para cada obra sumar en una variable si fue comprada y sumar al número de obras en el rango y sumar al número de artistas y tener una variable que calcule la máxima posición de las obras dentro del rango



1. Por cada obra y cada id de la obra hacer búsqueda binaria del Id y encontrar el nombre del artista y agregarlo a la obra.
2. Agregar las primeras 3 obras a una lista
3. Recorre las ultimas 3 posiciones del rango
4. Por cada obra y cada id hacer búsqueda binaria para encontrar los nombres de los artistas y agregarlos a la obra



Como se puede ver se logro hacer un algoritmo efeciente y la razon por la cual no se miro otra opción es que despues de analizar el problema y darnos cuenta que se debia hacer la busqueda del nombre del artista, consideramos que intentar optimizar mas podria causar demoras en otras partes como en la carga de datos, donde se podria agregar a las obras los datos de los artististas pero consideramos que la complejidad tanto de tiempo (nlog(m) o incluso nm ) y la complejidad de implementar no lo ameritaban.

Pruebas de tiempos:

Requerimiento 3 (Kevin Fernando):

Requerimiento 4 (Camilo Ortiz):

El algoritmo planteado para resolver el requerimiento 4 fue el siguiente:

1. Recorrer todas las obras



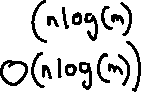
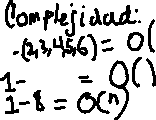
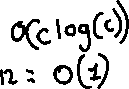
1. Por cada obra recorrer los ids
2. Por cada id hacer una busquedaBinaria
3. Agregar el nombre de los artistas a la obra



1. Guardar la nacionalidad



1. Sumar las nacionalidades de los artistas de la obra
2. Recorrer las nacionalidades de la obra que se recorrio y agregar a la lista que contiene la obras unicas de una nacionalidad
3. Recorrer las nacionalidades de la obra y sumar el numero de artistas al total de obras por nacionalidad
4. Crear una lista de tuplas que contengan el nombre del pais y el numero de obras por nacionalidad
5. Hacerle sort a la lista de tuplas
6. Obtener el primer elemento de la lista de tuplas ordenada
7. Obtener la lista del pais con mas obras unicas.



Al igual que en el punto anterior consideramos que lograr un comportamiento linearitmico es un orden de crecimiento aceptable el cual cumple con las necesidades de la aplicación, sin necesidad de desarrollar un algoritmo muy complejo.

Pruebas de tiempo:

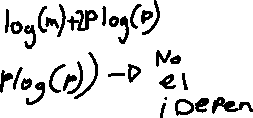
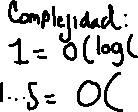
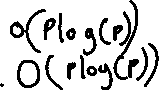
Requerimiento 5:

Para el requerimiento 5 se hizo el siguiente algoritmo:

1. Buscar el departamento con búsqueda binaria
2. Recorrer el sub-arreglo hasta que el departamento cambie
3. Por cada obra sumar al total, sumar al peso, calcular costo y sumarlo al total de costo y al costo de la obra
4. Por cada obra recorrer los ids y hacer búsqueda binaria de la posición y agregar los artistas a la obra



1. Agregar a 2 listas separadas, 1 para los antiguos y otra para los más caros cada obra, para antiguos revisar si la fecha > 0
2. Hacer sort por fecha a los antiguos y sort por costo a los más caros
3. Crear sublistas de los primeros 5 de los antiguos y los más caros



El requerimiento 5 también se logró hacer en tiempo linearitmico, en teoría en el promedio de casos debe ser igual o mejor que el requerimiento 4 ya que con un rango que contenga menos de la totalidad de los datos debería ser un poco más rápido.

Prueba de tiempo: