

Camilo Ortiz Cruz 201821615 c.ortizc@uniandes.edu.co

Kevin Fernando Gómez Camargo 202015120 k.gomezc@uniandes.edu.co

Análisis de complejidad

Maquinas

	Kevin Fernando	Camilo
<i>Procesador</i>	2,5 GHz Intel Core i5 de dos núcleos	11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz
<i>Memoria RAM</i>	4 GB	16 GB
<i>Sistema Operativo</i>	macOS Catalina 10.15.7	Windows 10 Home 64-bits

Pruebas de tiempo

Tabla pruebas de tiempo de Camilo.

size	artists	artworks	req1p	req2p	req5p	req1exp	req2exp	req4exp	req5exp	req6exp
small	1948	768	862	316	394	10	10	10	15	0,00
5%	4996	7572	2334	3247	4109	10	10	80,4	162,75	3,12
20%	8724	29489	4352	12880	15983	13,324	15	326,56	726,38	15,62
50%	12137	71432	6117	32009	38888	15,625	50,78	703,34	1849,04	34,37
100%	15223	138150	76644	61484	76117	15,62	74,22	1409,75	3937	106,25

Tabla pruebas de tiempo de Kevin Fernando.

Size	Req1 (ms)	Req2 (ms)	Req3 (ms)	Req4 (ms)	Req5 (ms)	Req6 (ms)
Small	4.20	7.99	1.48	34.44	49.80	5.36
50%	27.55	297.94	237.24	2477.70	6670.87	14.60
Large	30.28	479.15	541.84	5191.49	13196.23	172.95

Para todas las pruebas de tiempo los encabezados req1p hacen referencia al subconjunto de los valores en el rango.

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 límite inferior y seguir hasta que el valor se vuelva mayor que el límite superior, cuando los valores estén dentro del límite superior e inferior se realizarían las operaciones necesarias del requerimiento, que son sumar el número 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:

Opción 1:

Recorrido del arreglo: $O(n)$

Suma total: $O(1)$

Insercion lista: $O(1)$

Sublista: $O(1) \rightarrow$ solo son 3 elementos

Total: $O(n)$

Opción 2:

Busqueda ceil: $O(\log(n))$

Recorrido sub: $O(m)$

Insercion lista: $O(1)$

Sublista: $O(1) \rightarrow$ solo son 3 elementos

Total: $O(\log(n) + m)$
 $= O(m)$

De las dos opciones se escogió la 2da, 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.

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
4. 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.
5. Agregar las primeras 3 obras a una lista
6. Recorre las ultimas 3 posiciones del rango
7. Por cada obra y cada id hacer búsqueda binaria para encontrar los nombres de los artistas y agregarlos a la obra

Complejidad:

n = Obras m = artistas p = sublista obras en rango

1. $O(\log(n))$

2,3,4,5. $O(p \log(m))$ $m > n$

6,7. $O(\log(m))$

$$Tot = O(\log(n) + p \log(m) + \log(m))$$
$$= O(p \log(m))$$

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 ($n \log(m)$ o incluso nm) y la complejidad de implementar no lo ameritaban.

Requerimiento 3 (Kevin Fernando):

Se planteo la siguiente solución para el requerimiento 3:

1. Crear dos arreglos, uno para los medios (AM) y otro para las obras (AO)
2. Encontrar el ConstituentID del artista por busqueda binaria
3. Confirmar que el artista existe
4. Si existe, preguntar para cada obra si el ConstituentID se encuentra en los autores
5. Si sí se encuentra, sumar 1 al numero de obras
6. Si el medio ya está en AM, crear un diccionario con la información de la obra y agregarlo al final del arreglo correspondiente al medio dentro del AO
7. Si el medio no está, se crea el diccionario, se agrega el medio al final del AM, se crea un arreglo al final del AO, y se agrega el diccionario al final del arreglo creado en el AO
8. Se calcula el tamaño del arreglo en AO (número de obras) correspondiente a cada medio y se compara con una variable "mayor" que inicia en 0
9. Si el tamaño es mayor a "mayor", se guarda el medio y el arreglo con las obras de ese medio
10. Se crea un diccionario con las respuestas para facilitar la impresión de las respuestas

Complejidad:

Paso 1: $O(1)$

Paso 2: $O(\log(n))$ n : número de artistas

Paso 3: $O(1)$

Pasos 4 – 7: $O(m \times a \times k)$

m : número de obras

a : número de artistas por obra, es pequeño

k : constante relacionada a la cantidad de medios de cada artista, se establece una constante porque la longitud del arreglo medios no siempre es la misma y no sabemos cómo cambia, entonces no se sabe exactamente el orden del isPresent en términos de los datos.

Paso 8: $O(y)$ y : número de medios

Complejidad total:

$$O(1) + O(\log(n)) + O(m \times a \times k) + O(y)$$

$O(m \times a \times k)$, pero a es pequeño porque generalmente una obra tiene pocos artistas

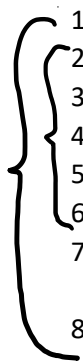
$$O(m \times k)$$

$$O(m)$$

El orden logrado es lineal porque generalmente un artista no tiene una cantidad de medios tan significativa con respecto al número de obras. Es un orden de crecimiento adecuado que funciona bien para lo que se quiere haciendo una implementación con únicamente listas.

Requerimiento 4 (Camilo Ortiz):

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

- 
1. Recorrer todas las obras
 2. Por cada obra recorrer los ids
 3. Por cada id hacer una búsqueda Binaria
 4. Agregar el nombre de los artistas a la obra
 5. Guardar la nacionalidad
 6. Sumar las nacionalidades de los artistas de la obra
 7. Recorrer las nacionalidades de la obra que se recorrió y agregar a la lista que contiene la obras únicas de una nacionalidad
 8. Recorrer las nacionalidades de la obra y sumar el número de artistas al total de obras por nacionalidad
 9. Crear una lista de tuplas que contengan el nombre del país y el número de obras por nacionalidad
 10. Hacerle sort a la lista de tuplas
 11. Obtener el primer elemento de la lista de tuplas ordenada
 12. Obtener la lista del país con más obras únicas.

Complejidad: $n = \text{obras}$ $m = \text{artistas}$
 $c = \text{nacionalidades}$
 $1-(2,3,4,5,6) = O(n \log(m))$

$$1-7 = O(n)$$

$$1-8 = O(n)$$

Ordenamiento:
MergeSort

$$1-\{2...8\} = O(n \log(m))$$

$$9. = O(c)$$

$$10. O(c \log(c))$$

$$c < m \text{ y } m < n$$

$$11 \text{ y } 12 = O(1)$$

$$T_{\text{ot}} = O(n \log(m) + c \log(c))$$

$$= O(n \log(m))$$

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

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

Complejidad:

$$1 = O(\log(n))$$

$$2..5 = O(P \log(m))$$

$$6. O(P \log(P))$$

$$7. O(1)$$

p: sublista de los elementos de 1 departamento

n: obras

m: artistas

$$n > m$$

$$n \geq P$$

$$T_{ot} = O(\log(n) + P \log(m) + 2P \log(P))$$

$$= O(P \log(m) + P \log(P)) \rightarrow \text{No se puede determinar el menor } i \text{ depende del input!}$$

El requerimiento 5 también se logró hacer en tiempo lineal, 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.

Requerimiento 6:

1. Búsqueda de la fecha mínima
2. Recorrer subarreglo entre el rango de fechas
3. Si esta en el rango de fechas revisar si cumple el requisito de área
4. Calcular área
5. Revisar si la suma de esa área sobrepasa el área máxima
6. Si no sobrepasa recorrer los ids de la obra y hacer búsqueda binaria en artistas para conseguir el nombre y agregar los nombres a la obra
7. Si se pasa terminar ciclo
8. Crear sublistas de los 5 primeros y 5 últimos

Complejidad:

$$1 = O(\log(n))$$

$$2..7 = O(P \log(m))$$

$$8 = O(1)$$

n = obras

m = artistas

p = sublista de obras en rango

$$T_{ot} = O(\log(n) + P \log(m))$$

