Análisis del Reto

Grupo 2 Sección 4

Requerimiento 3 - Daniel Rodriguez - ds.rodriguezf1@uniandes.edu.co - 202014760  
Requerimiento 4 - Santiago Forero - s.forerog2@uniandes.edu.co - 202111446

N = Artists.csv

M = Artworks.csv

**Requerimiento 1**

* Análisis complejidad temporal:

La función SortArtists tiene una complejidad temporal de O(n). Inicialmente se realiza un recorrido del rango de años ingresado por el usuario para poder extraer cada sublista de artistas, tomando como índice los años del rango, y ya que el recorrido el rango de años va del primer año al último, no hay necesidad de usar ningún algoritmo de ordenamiento, pues va a quedar ordenado a medida que se extraigan los datos. Dentro de este ciclo se recorre cada artista por sub lista de artistas, para agregarlo a una lista más grande que se retorna al usuario. Así que la complejidad temporal del primer requerimiento es:

O(n)

* Pruebas de tiempo – Tablas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Small | 20 pct | 50 pct | 80 pct | Large |
| Reto 1 | 62.5 | 359.375 | 531.25 | 671.875 | 710.93 |
| Reto 2 |  |  |  |  |  |

* Gráfica

Se puede ver que tiende a ser logarítmica, tal como se esperaría teniendo en cuenta la teoría

**Requerimiento 2**

* Análisis complejidad temporal:

La función SortArtworks tiene una complejidad temporal de O(m log(m)) + O(m). Inicialmente filtra las llaves (fechas de adquisición de cada obra) en el rango de fechas que ingrese el usuario, teniendo en el peor de los casos O(m). Posteriormente usa la función MergeSort para ordenar que tiene una complejidad de O(m log(m)). Finalmente se recorre cada llave y para cada llave se recorren las obras asociadas para agregarlas a una lista más grande que se retorna al usuario, esto es también complejidad O(m).

Así que la complejidad temporal del segundo requerimiento es:

O(m log(m)) + O(m)

* Pruebas de tiempo – Tablas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Small | 20 pct | 50 pct | 80 pct | Large |
| Reto 1 | 765.625 | 81093.75 | 311593.75 | NA | NA |
| Reto 2 |  |  |  |  |  |

* Gráficas
* Por cuestiones del procesador del computador, las pruebas no dan resultados muy reveladores.

**Requerimiento 3 – Daniel Rodriguez**

* Análisis complejidad temporal:
* Pruebas de tiempo – Tablas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Small | 20 pct | 50 pct | 80 pct | Large |
| Reto 2 |  |  |  |  |  |

* Gráficas

**Requerimiento 4 – Santiago Forero**

* Análisis complejidad temporal:

P = Número de nacionalidades

La función ArtworksNacionalidad tiene una complejidad temporal de O(p) + O(p log(p)). Inicialmente recorre cada nacionalidad a la cual le extrae la cantidad de obras asociadas con la función size, por lo que la complejidad de esta parte es O(p), posteriormente se realiza un ordenamiento de Merge Sort, de done se obtiene el O(p log(p)). Finalmente por medio del mapa, se extrae la lista de las obras asociadas a la nacionalidad con más obras, con función O(1), de la cual posteriormente se imprime en pantalla las primeras y últimas 3 posiciones. Así que la complejidad temporal del cuarto requerimiento es:

O(p) + O(p log(p))

* Pruebas de tiempo – Tablas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Small | 20 pct | 50 pct | 80 pct | Large |
| Req 4 | 4135.42 | 493015.625 | NA | NA | NA |

* Gráficas

Por cuestiones del procesador del computador, las pruebas no dan resultados muy reveladores.

**Requerimiento 5**

* Análisis complejidad temporal:

R = Primeras 5 obras más costosas

S = Primeras 5 obras más antiguas

T = Cantidad de obras del departamento ingresado por el usuario

* La función CostoTransDept tiene una complejidad temporal de O(t\*r\*s) + O(r log(r)) + O(s log(s)). Esto debido a que realiza un recorrido por todas las obras mientras crea listas más pequeñas con las 5 obras más antiguas y costosas respectivamente. No obstante, ya que cada una de estas listas tiene únicamente 5 elementos siempre, se podría escribir también la complejidad temporal como O(t). En todo caso, durante este recorrido cada lista r y s se ordena por medio de Merge Sort una sola vez al inicio con complejidades de O(r log(r)) + O(s log(s)) o 2\*O(5 log(5)). Finalmente, cabe resaltar que con el uso de mapas la complejidad se reduce bastante ya que no hay que recorrer todas las obras, sino únicamente las del departamento ingresado por el usuario. Así que la complejidad temporal del quinto requerimiento es:

O(t\*r\*s) + O(r log(r)) + O(s log(s))

o

O(t)

* Pruebas de tiempo – Tablas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Small | 20 pct | 50 pct | 80 pct | Large |
| Req 5 | 15.625 | 148.437 | 317.7 | 484.375 | 601.56 |

* Gráficas

Se puede ver que tiende a ser lineal, tal como en la teoría

**Requerimiento 6**

* Análisis complejidad temporal:
* U = Medios usados por un artista

La función artistsProlific tiene una complejidad temporal de O(m) + O(n) + O(n log n). Inicialmente llama a la función del requerimiento 1, con complejidad O(n). Posteriormente, por cada artista de la lista ordenada y filtrada recorre todas sus obras asociadas por medio del índice del mapa, así que como resultado de esta parte queda una complejidad de O(m) + O(n), esto debido a que, aunque pudiese parecer que se recorre una dentro de la otra, realmente solo se buscan las obras asociadas a un artista específico, por lo que después de recorrer todos los artistas cada obra habrá sido recorrida más o menos 1 vez. Después, al recorrer cada obra del artista se crea un nuevo mapa que tiene como llaves los medios que utiliza un artista, y como valor la lista asociada de ese medio y ese artista, por lo que al recorrerse y ordenarse tiene una complejidad de O(u) + O(u logu). Finalmente, se ordenan los artistas buscando cuales son los más prolíficos con complejidad O(n log n), para ya retornar al usuario. Así que la complejidad temporal del sexto requerimiento es:

O(m) + O(n\*(u + u log u)) + O(n log n)

o

O(m) + O(n) + O(n log n)

* Pruebas de tiempo – Tablas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Small | 20 pct | 50 pct | 80 pct | Large |
| Req 6 |  |  |  |  |  |

* Gráficas

Se puede ver que tiende a ser lineal, tal como en la teoría