***Documento de Análisis***

En este documento estara conformado por el análisis de complejidad de cada uno de los requerimientos del Reto 1 y a su vez, las pruebas de tiempos de ejecución de los mismos.

1. **Complejidad de los Algoritmos**

n = Tamaño del Catalogo de Artistas

m = Tamaño del Catalogo de Obras

**Req 1.**

**Graphical user interface, text

Description automatically generated**

El algotimo depende de n. Se recorre en un for loop la lista de los artistas organizada previamente por Begin Date, por lo que la complejidad del requerimiento es O(n)

**Complejidad:** O(n)

**Req 2.**

**Graphical user interface, text, application

Description automatically generated**

El algotimo depende de m. En la primera funcion, se recorre en un for loop la lista de las obras organizadas previamente por fecha de adquision, por lo que la complejidad de la funcion es O(m). En la segunda funcion se recorre la lista de obras dentro del rango ingresado por el usuario (z). Por lo que la complejidad de esta funcion es O(z). Al hacer la suma de las complejidades como (z) nunca va a ser mayor que (m), se concluye que la complejidad del requerimiento es O(m).

**Complejidad:** O(m)

**Req 3. – Implementado por Sofia Velasquez**

**Text

Description automatically generated**

El algotimo depende de n y de m. En la primera funcion se recorre en un for loop la lista de los artistas, en el peor caso el artista a buscar se encuentra en la ultima posicion, por lo que la complejidad de la funcion es O(n). En la segunda funcion se recorre en un for loop la lista de obras, por lo que la complejidad de esta funcion es O(m).La tercera funcion depende del tamaño de la lista de obras del artista (a) la cual se recorre en un for loop por lo que la complejidad de esta funcion es O(a). Al hacer la suma de las complejidades como (a) nunca va a ser mayor que (m), y no se sabe si (n) es mayor que (m) o viceversa se concluye que la complejidad del requerimiento en el peor caso es O(n + m).

**Complejidad:** O(n + m)

**Req 4.** Implentado por Valeria Caro

**Text

Description automatically generated with medium confidence**

El algotimo depende de m y n. Se recorre en un for loop la lista de obras por lo que la complejidad de la funcion es O(n) y luego, se realiza otro for loop por los artistas O(m). Posteriormente se organiza la lista, y por lo tanto, la complejidad es O(n + m).

**Complejidad:** O(n + m)

**Req 5.**

Graphical user interface, text, application, email

Description automatically generated

Text, timeline

Description automatically generated

El algotimo depende de m. En la primera funcion, se recorre en un for loop la lista de las obras organizadas previamente por fecha, por lo que la complejidad de la funcion es O(m). En la segunda funcion se recorre la lista de obras dentro del departamento ingresado por el usuario (d). Por lo que la complejidad de esta funcion es O(d). En el view, se hace un merge sort de la lista de obras dentro del departamento, el cual tiene una complejidad de O(d\*log(d)). Al hacer la suma de las complejidades como (d) nunca va a ser mayor que (m) pero no sabe si (d\*log(d)) es mayor que m, se concluye que la complejidad del requerimiento es O(m + d\*log(d)).

**Complejidad:** O(m + d\*log(d)).

**Req 6.**

Graphical user interface, text

Description automatically generated

El algotimo depende de m. Se recorre en un for loop la lista de obras por lo que la complejidad de la funcion es O(m). Se concluye que la complejidad del requerimiento es O(m).

**Complejidad:** O(m)

1. **Tiempos de Ejecución**

**Req 1.**

|  |  |  |
| --- | --- | --- |
| % de Muestra  (ARRAYLIST) | Tiempo(ms) | Sort\_List(ms)  (Merge Sort) |
| 0,50% | 0.0 | 31.25 |
| 5% | 0.0 | 93.75 |
| 10% | 0.0 | 140.62 |
| 20% | 15.62 | 171.87 |
| 30% | 15.62 | 218.75 |
| 50% | 15.62 | 312.5 |
| 80% | 15.62 | 325.0 |
| 100% | 31.25 | 343.75 |

**Req 2.**

|  |  |  |
| --- | --- | --- |
| % de Muestra  (ARRAYLIST) | Tiempo(ms) | Sort\_List(ms)  (Merge Sort) |
| 0,50% | 0.0 | 15.62 |
| 5% | 0.0 | 171.87 |
| 10% | 46.87 | 328.12 |
| 20% | 62.5 | 718.75 |
| 30% | 62.5 | 1171.87 |
| 50% | 156.25 | 1953.12 |
| 80% | 218.75 | 3203.12 |
| 100% | 298.87 | 3859.37 |

**Req 3.**

|  |  |
| --- | --- |
| % de Muestra  (ARRAYLIST) | Tiempo(ms) |
| 0,50% | 0.0 |
| 5% | 31.25 |
| 10% | 31.25 |
| 20% | 46.87 |
| 30% | 78.12 |
| 50% | 93.75 |
| 80% | 171.87 |
| 100% | 203.12 |

**Req 4.**

|  |  |
| --- | --- |
| % de Muestra  (ARRAYLIST) | Tiempo(ms) |
| 0,50% | 15.62 |
| 5% | 15.62 |
| 10% | 31.25 |
| 20% | 93.75 |
| 30% | 109.375 |
| 50% | 171.875 |
| 80% | 328.12 |
| 100% | 328.12 |

**Req 5.**

|  |  |  |
| --- | --- | --- |
| % de Muestra  (ARRAYLIST) | Tiempo(ms) | Sort\_List(ms)  (Merge Sort) |
| 0,50% | 31.25 | 15.62 |
| 5% | 93.75 | 156.25 |
| 10% | 187.5 | 375.0 |
| 20% | 390.62 | 671.87 |
| 30% | 593.75 | 1140.62 |
| 50% | 1062.5 | 1906.25 |
| 80% | 1781.25 | 3046.87 |
| 100% | 2234.37 | 3656.25 |

**Req 6.**

|  |  |
| --- | --- |
| % de Muestra  (ARRAYLIST) | Tiempo(ms) |
| 0,50% | 31.25 |
| 5% | 31.25 |
| 10% | 31.25 |
| 20% | 78.12 |
| 30% | 78.12 |
| 50% | 140.62 |
| 80% | 203.12 |
| 100% | 250.0 |

A continuacion se presenta el Anexo de las graficas de cada unos de los requerimientos.

Anexo de Graficas:

Req 1.

Req 2.

Req 3.

Req 4.

Req 5.

Req 6.

Grafica de Comparacion de todos los requerimientos: