**ANÁLISIS DEL RETO**

David Laverde, 202225363, d.laverde@uniandes.edu.co

Ivan Alejandro Maldonado,202310631, i.maldonado@uniandes.edu.co

Estudiante 3, código 3, email 3

# **Requerimiento <<1>>**

Plantilla para el documentar y analizar cada uno de los requerimientos.



## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Estructura de datos, Numero de partidos a consultar, Nombre del equipo a consultar y Condición del equipo (home o away) |
| **Salidas** | Lista con los datos deseados y ya filtrado por primeros 3 y últimos 3. |
| **Implementado (Sí/No)** | Si, por Alejandro Maldonado |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Recorrido de nueva lista (for) | O(n) |
| Agregar cosas a la lista (addlast) | O(1) |
| Ordenar la lista (mergesort) | O(n log n) |
| Creación de una sublista (sublist) | O(3) |
| Copiar los elementos a la nueva lista (for) | O(n) |
| ***TOTAL*** | ***O(n log n)*** |

## **Pruebas Realizadas**

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

|  |  |
| --- | --- |
| Procesador | AMD Ryzen 7 4800H 2.90GHz |
| RAM | 8,00 GB 3200hz |
| Sistema operativo | Windows 11 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| small | 27.33 |
| 5pct | 99.66 |
| 10pct | 196.90 |
| 20pct | 375.27 |
| 30pct | 537.03 |
| 50pct | 847.06 |
| 80pct | 1204.63 |
| large | 1544.14 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

la complejidad total del código depende del número de elementos en el conjunto de datos, y en el peor caso, estará dominada por el paso de ordenamiento, que es O(n log n). Otras operaciones dentro del bucle también contribuirán a la complejidad lineal en función del número de elementos que cumplen las condiciones. Por lo tanto, en general, se puede considerar que el código tiene una complejidad en el orden de O(n log n) en el peor caso.

# **Requerimiento <<2>>**

Plantilla para el documentar y analizar cada uno de los requerimientos.

Texto

Descripción generada automáticamente

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Estructura de datos, numero de goles a consultar, nombre de jugador a consultar. |
| **Salidas** | Lista de goles anotados por el jugador. |
| **Implementado (Sí/No)** | Si, implementado por Alejandro Maldonado |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Paso 1 | O(...) |
| Paso 2 | O(...) |
| Paso …. | O(...) |
| ***TOTAL*** | ***O(...)*** |

## **Pruebas Realizadas**

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

|  |  |
| --- | --- |
| Procesador | AMD Ryzen 7 4800H 2.90GHz |
| RAM | 8,00 GB 3200hz |
| Sistema operativo | Windows 11 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
|  |  |
|  |  |
|  |  |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

# **Requerimiento <<3>>**

Plantilla para el documentar y analizar cada uno de los requerimientos.

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Nombre del equipo a consultar, fecha inicial del periodo, fecha final del periodo. |
| **Salidas** | Lista de partidos que jugo el equipo a consultar en el periodo especificado. |
| **Implementado (Sí/No)** | Si, implementado por Alejandro Maldonado |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Paso 1 | O(...) |
| Paso 2 | O(...) |
| Paso …. | O(...) |
| ***TOTAL*** | ***O(...)*** |

## **Pruebas Realizadas**

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

|  |  |
| --- | --- |
| Procesador | AMD Ryzen 7 4800H 2.90GHz |
| RAM | 8,00 GB 3200hz |
| Sistema operativo | Windows 11 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
|  |  |
|  |  |
|  |  |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el analisis de complejidad.

# **Requerimiento Ejemplo**

## **Descripción**



Este requerimiento se encarga de retornar un dato de una lista dado su ID. Lo primero que hace es verificar si el elemento existe. Dado el caso que exista, retorna su posición, lo busca en la lista y lo retorna. De lo contrario, retorna None.

|  |  |
| --- | --- |
| **Entrada** | Estructuras de datos del modelo, ID. |
| **Salidas** | El elemento con el ID dado, si no existe se retorna None |
| **Implementado (Sí/No)** | Si. Implementado por Juan Andrés Ariza |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Buscar si el elemento existe (isPresent) | O(n) |
| Obtener el elemento (getElement) | O(1) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el ID 1.

|  |  |
| --- | --- |
| Procesadores | AMD Ryzen 7 4800HS with Radeon Graphics |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 10 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| small | 0.05 |
| 5 pct | 0.33 |
| 10 pct | 1.28 |
| 20 pct | 2.54 |
| 30 pct | 4.98 |
| 50 pct | 7.51 |
| 80 pct | 13.81 |
| large | 25.97 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (ms)** |
| small | Dato1 | 0.05 |
| 5 pct | Dato2 | 0.33 |
| 10 pct | Dato3 | 1.28 |
| 20 pct | Dato4 | 2.54 |
| 30 pct | Dato5 | 4.98 |
| 50 pct | Dato6 | 7.51 |
| 80 pct | Dato7 | 13.81 |
| large | Dato8 | 25.97 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.