**ANÁLISIS DEL RETO**

Gabriela Gómez, 202420506, g.gomezh2@uniandes.edu.co

# **Requerimiento <<4>>**

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

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | El catalogo que en este caso es un árbol rojo negro, n, que es la cantidad de datos, edad\_i que es la edad inicial y el edad\_f que es la edad final. |
| **Salidas** | El tiempo de carga de los datos en la función, el total de crímenes filtrados y una lista llamada respuesta con la info de cada caso filtrado. |
| **Implementado (Sí/No)** | Si Gabriela Gomez |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| |  |  | | --- | --- | | Convertir las edades de los parámetros a enteros | O(1) | | O(1) |
| |  |  | | --- | --- | | Obtener el árbol rojo-negro de edades |  | | O(1) |
| |  |  | | --- | --- | | Obtener los valores en el rango de edades |  | | O(L) |
| |  |  | | --- | --- | | Crear listas para almacenar los crímenes graves  y menores | O(1) | | O(1) |
| |  | | --- | | Filtrar los crímenes según su gravedad | | O(E) |
| |  |  | | --- | --- | | Ordenar las listas de crímenes por el criterio dado | O(G log G + M log M) | | O(G log G + M log M) |
| |  | | --- | | Combinar las listas ordenadas | | |  | | --- | |  |  |  | | --- | | O(G + M) | |
| |  |  | | --- | --- | | Crear la respuesta con los primeros N crímenes | O(N) | | O(N) |
| |  |  | | --- | --- | | Calcular el tiempo de ejecución |  | | O(1) |
| |  | | --- | | Calcular el total de crímenes | | O(1) |
| ***TOTAL*** | ***O(L + E + G log G + M log M + N)*** |

 **L** es el número de valores en el rango de edades (rbt.values).

 **E** es el número total de crímenes en el rango de edades.

 **G** es el número de crímenes graves ("Part 1").

 **M** es el número de crímenes menores ("Part 2").

 **N** es el número de crímenes solicitados para la respuesta.

## **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).

* N = 10
* Edad\_i = 20
* Edad\_f = 45

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| 20 | 345,65 |
| 40 | 567,04 |
| 60 | 802,33 |
| 80 | 964,08 |
| 100 | 1143,58 |

### **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.

El comportamiento de la gráfica es muy cercano a lineal dado a la complejidad logarítmica de varias operaciones relacionadas con el ordenamiento y la búsqueda en árboles pero también a los recorridos O(N). Además, el uso de dos ciclos (for) para recorrer por completo las lista de todos los datos entre las edades filtradas hace que con muchos datos la gráfica crezca rápidamente constante (pendiente,rapidez). En conclusión, la función es muy cercana a la línea de tendencia lo cual nos dice que su complejidad es muy parecida a O(N), sino que en este caso por la cantidad de recorridos en diferentes estructuras enlazadas se usan mas variables para su expresion.

# **Requerimiento <<6>>**

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

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | El catalogo que en este caso es un binario, n que es la cantidad de datos, genero que es un filtro y mes que es otro filtro. |
| **Salidas** | El tiempo de carga de los datos en la función, y el total de top de áreas siendo una lista con info de cada dato |
| **Implementado (Sí/No)** | Si Grupal |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Obtener árbol de fechas | O(1) |
| Crear mapa para contar los crímenes por area | O(1) |
| Recorrer todas las fechas en el árbol | O(F) |
| Iterar sobre cada lista de crímenes | O(C) |
| Filtrar por mes y genero | O(1) |
| Obtener y procesar área y nombre del área O(1) | O(1) |
| Inicializar área en el mapa si no existe | O(log(A)) |
| Poner datos del área en el mapa | O(1) |
| Actualizar conteo de crímenes | O(1) |
| Actualizar conteo por año | O(log(Y)) |
| Convertir mapa de áreas a lista | O(A) |
| Ordenas las areaS | |  | | --- | |  |  |  | | --- | | O(A log A) | |
| Obtener las N áreas mas seguras | O(N) |
| Calcular tiempo trasncurrido | O(1) |
| ***TOTAL*** | ***O(F\*C+A log A +N)*** |

* **F** es el número de fechas en el árbol de fechas de ocurrencia.
* **C** es el número de crímenes por fecha.
* **A** es el número de áreas.
* **Y** es el número de años de los crímenes.
* **N** es el número de áreas más seguras que se necesitan.

## **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).

* Numero de registros: 8
* Sexo: F
* Mes: 6

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| 20 | 255.43 |
| 40 | 505.22 |
| 60 | 1305.06 |
| 80 | 2357.74 |
| 100 | 3591.63 |

### **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.

El comportamiento de la gráfica es entre lineal y cuadrático dado a la complejidad logarítmica de varias operaciones relacionadas con el ordenamiento y la búsqueda en árboles. Además, el uso de dos ciclos (for) para recorrer por completo las lista de todos los datos entre el género y el mes filtrado hace que con muchos datos la gráfica crezca rápidamente. En conclusión, la función no esta alejada de la línea de tendencia lo cual nos dice que su complejidad no esta lejos de O(N), sino que en este caso por la cantidad de recorridos en diferentes estructuras enlazadas se usan mas variables.

# **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.