**ANÁLISIS DEL RETO**

Estudiante 1, código 1, email 1

Isabella González López,202223594,i.gonzalezl@uniandes.edu.co

Estudiante 3, código 3, email 3

# **Requerimiento <<n>>**

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

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Parámetros necesarios para resolver el requerimiento. |
| **Salidas** | Respuesta esperada del algoritmo. |
| **Implementado (Sí/No)** | Si se implementó y quien lo hizo. |

## **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).

|  |  |
| --- | --- |
| **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.

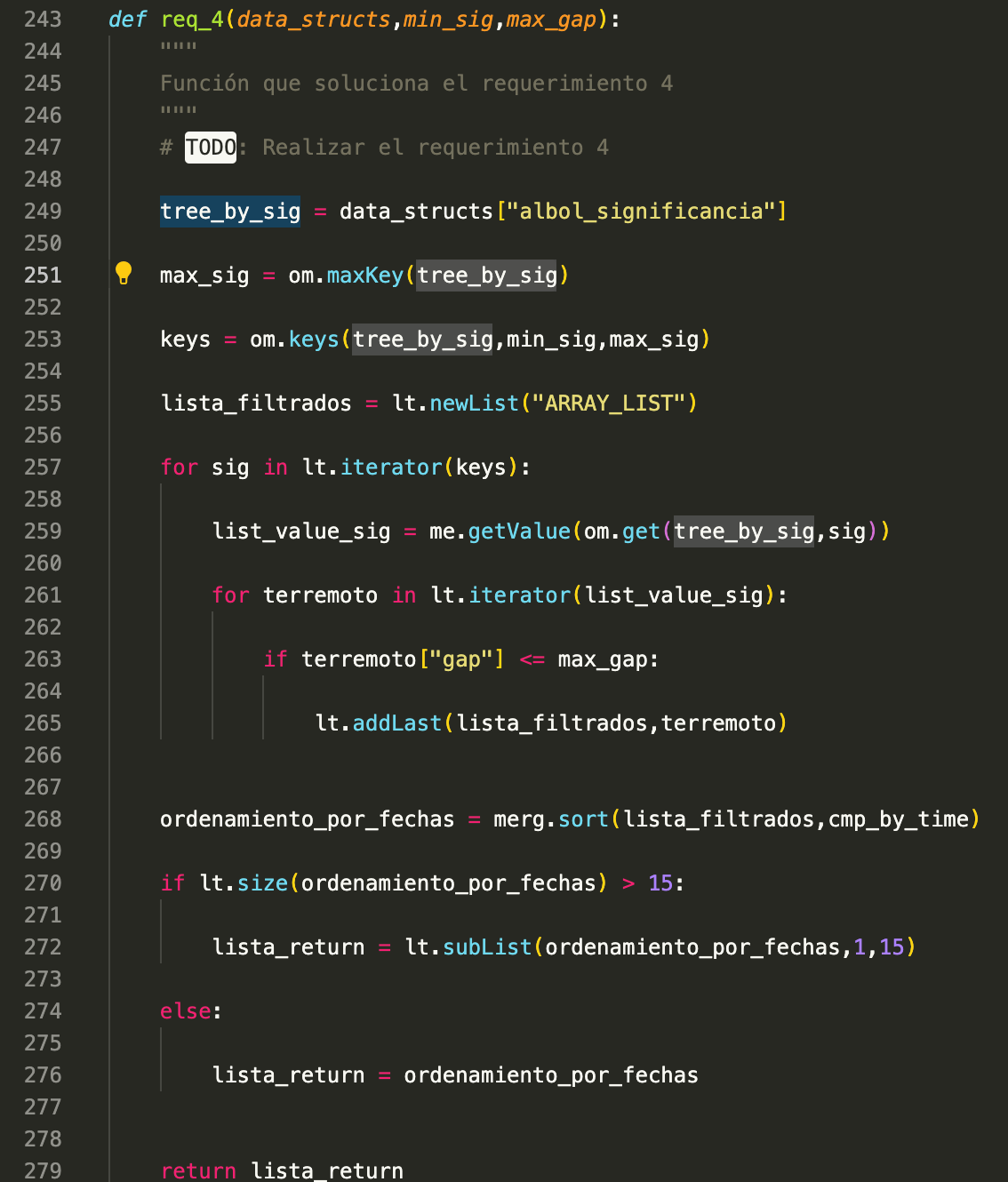
|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (ms)** |
| small | Dato1 | 0.05 |
| 5 pct | Dato2 | 0.33 |
| 10 pct | Dato3 | 1.28 |
| 20 pct | Dato4 | 2.54 |
| 30 pct | Dato5 | 4.98 |
| 50 pct | Dato6 | 7.51 |
| 80 pct | Dato7 | 13.81 |
| large | Dato8 | 25.97 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

# **Requerimiento 4**

## **Descripción**



La función toma tres parámetros. El código toma un árbol del data\_structs. Utiliza funciones de operaciones para obtener datos dentro de unos rangos. Luego, filtra y organiza eventos de acuerdo con la brecha especificada y finalmente devuelve una lista ordenada por fecha d elos eventos, esto hasta los 15 primeros si la lista supera este tamaño.d

|  |  |
| --- | --- |
| **Entrada** | La estructura de datos,la significancia mínima del evento y la distancia azimutal máxima del evento |
| **Salidas** | Los quince (15) eventos cronológicamente más recientes que cumplan con los parámetros especificados. |
| **Implementado (Sí/No)** | Si, Isabella González López |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Linea 249 | O(1) |
| Linea 251 y Linea 253 | O(n) |
| Linea 255 | O(1) |
| Linea 257-Linea 261 | O(n) |
| Linea 263 y Linea 265 | O(1) |
| Linea 268 | O(n log (n)) |
| Linea 270-Linea 276 | O(1) |
| ***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).

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

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

### **Graficas**

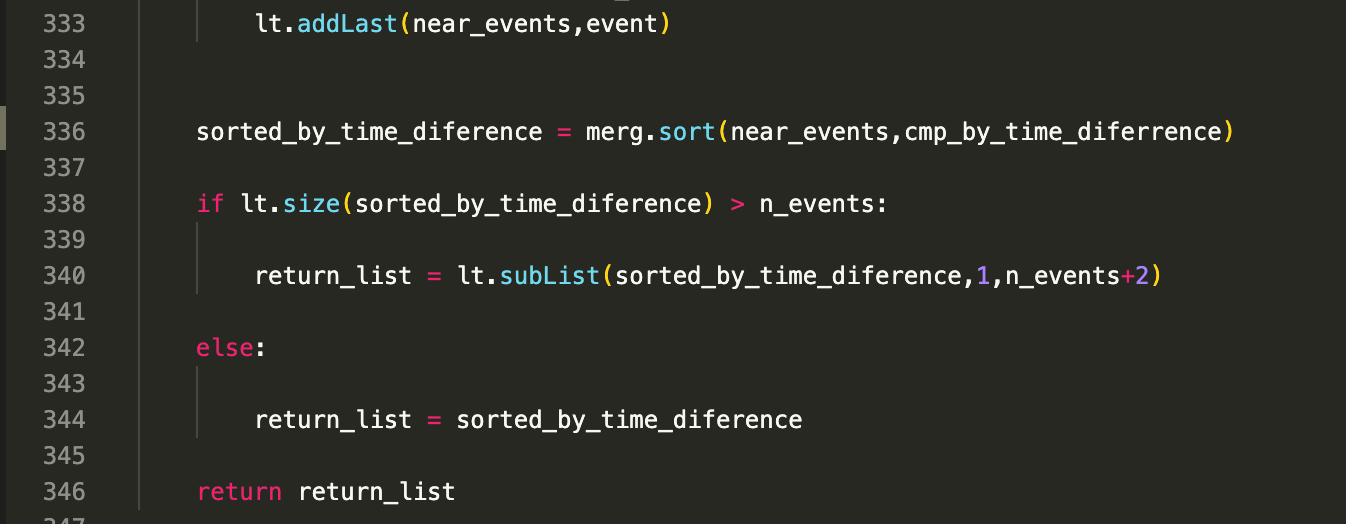
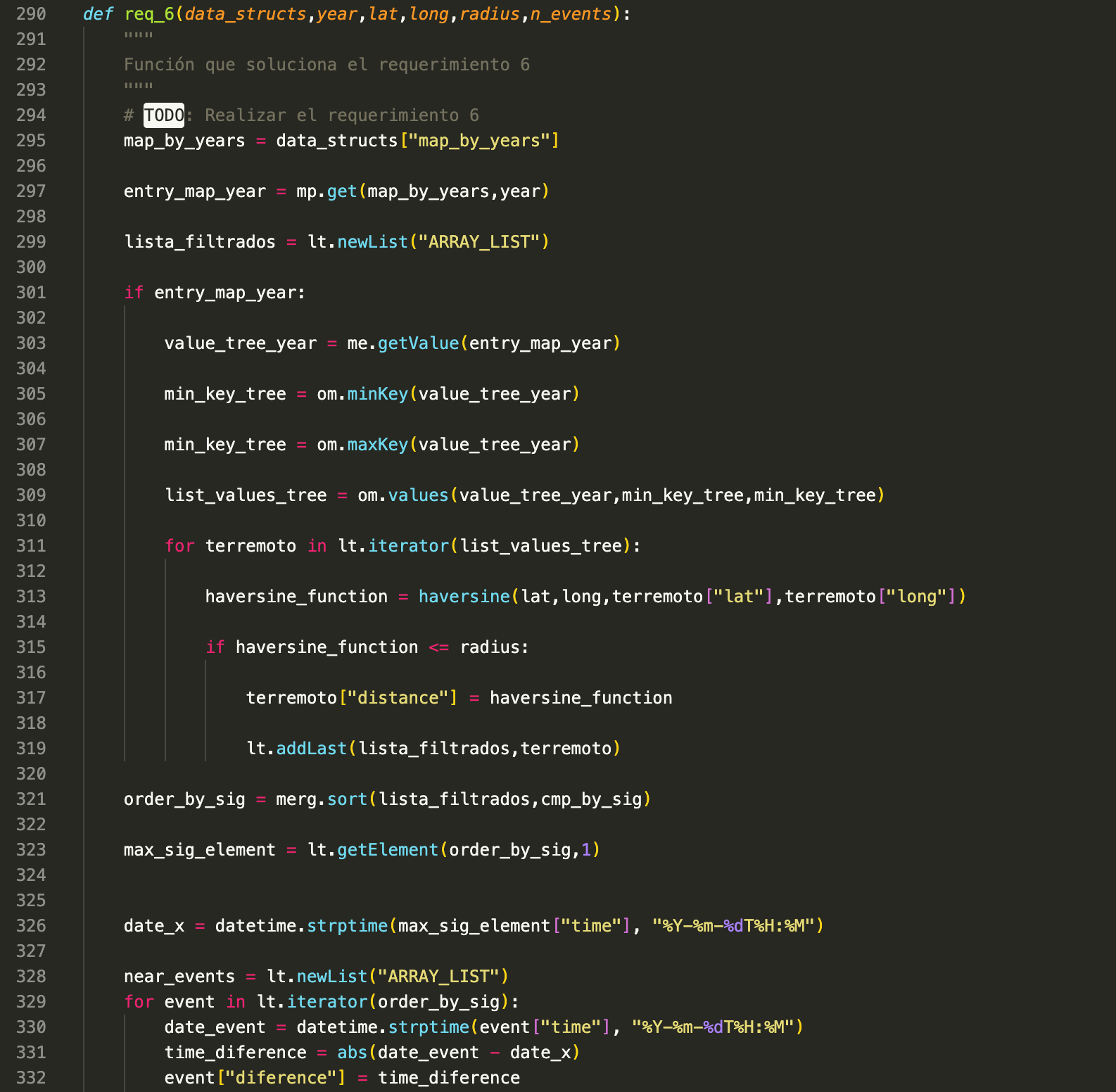
Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

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

# **Requerimiento 6**

## **Descripción**



La función toma como entrada varios datos, entre eso un mapa organizado por años del evento.Luego, organiza los eventos del año en especifico dentor de un radio según los datos dados.Luego filtra según la magnitud sismica para luego tomar el mayor.Luego, calcula la diferencia de tiempo entre el evento seleccionado y los demás eventos filtrados, ordena los datos según esta diferencia. Retorna una lista con los eventos más cercanos en tiempo al evento con mayor magnitud, retorna los n\_events.

|  |  |
| --- | --- |
| **Entrada** | La estructura de datos,el año relevante (en formato “%Y”),la Latitud de referencia para el área de eventos (lat), la longitud de referencia para el área de eventos (long), el radio [km] del área circundante (float) y el número de los N eventos de magnitud más cercana a mostrar. |
| **Salidas** | Los N eventos sísmicos más cercanos en tiempo, antes y después, al evento más significativo organizados cronológicamente desde el más reciente. |
| **Implementado (Sí/No)** | Si, Isabella González López |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Linea 295-Linea 303 | O(1) |
| Linea 305-Linea 311 | O(n) |
| Linea 313-Linea317 | O(1) |
| Linea 319 | O(n) |
| Linea 321 | O(n log(n)) |
| Linea 323-Llinea 328 | O(1) |
| Linea 329 | O(n) |
| Linea 330-Linea333 | O(1) |
| Linea 326 | O(n log(n)) |
| Linea 338-Linea 344 | O(1) |
| ***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).

|  |  |
| --- | --- |
| **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.

## **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.