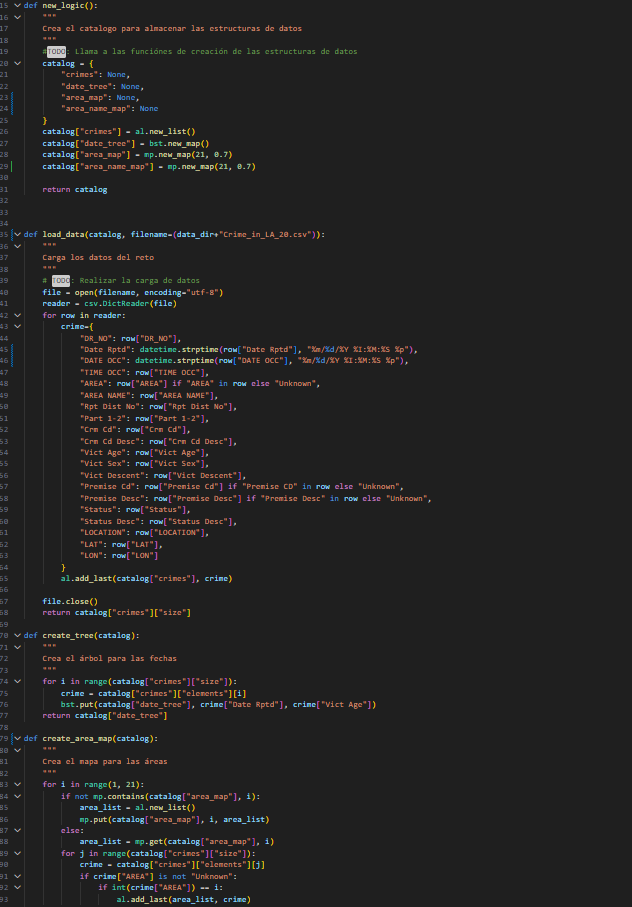
**ANÁLISIS DEL RETO**

Tomas Aponte, 202420148, [t.aponte@uniandes.edu.co](mailto:t.aponte@uniandes.edu.co)

Juan Diego García, [202423575jd.garcia12@uniandes.edu.co](mailto:202423575jd.garcia12@uniandes.edu.co)

# **Carga de datos**



La carga de datos está dividida en 3 funciones: new\_logic(), load\_data(), create\_tree(), create\_map(). New\_logic() se encarga de construir un catálogo general donde estarán guardadas una lista de arreglos (crimes), un árbol binario de búsqueda (date\_tree) y un mapa linear probing (area\_map). Load\_data() se encarga de llenar la lista de arreglos, generando un diccionario por cada fila del archivo csv, donde la llave del diccionario se nombra después de la columna de donde viene el dato. Una vez se tenga el diccionario con todos los datos, se agrega al catálogo a la lista de arreglos, y se procede a llenar el diccionario de la siguiente fila, quedando así la totalidad de los datos cargados. Por otro lado, create\_tree() se encarga de construir el árbol del catálogo, donde las llaves son la fecha del reporte de un crimen dado, y su valor la edad de la víctima del crimen; mientras que create\_map() se encarga de generar un mapa donde cada llave corresponde con el código de un área dada, y el valor de cada llave son los diccionarios asociados al área. Estas estructuras se crean por aparte debido a la complejidad temporal de la creación del árbol. Juntar todas las estructuras en una carga de datos aumentaría significativamente el tiempo de respuesta para cargar los primeros y últimos 5 registros del catálogo, que solo necesitan de la lista de diccionarios.

## **Descripción**

|  |  |
| --- | --- |
| **Entrada** | N/A |
| **Salidas** | Total de reportes, primeros y últimos 5 datos |
| **Implementado (Sí/No)** | Si, Juan Diego García |

## **Análisis de complejidad**

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Crea el catalogo para todas las estructuras de datos | O(1) |
| Extrae todos los registros a una lista | O(n) |
| Crea y llena las estructuras de datos necesarias | O(n) |
| Selecciona las primeros y últimos 5 | O(n) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

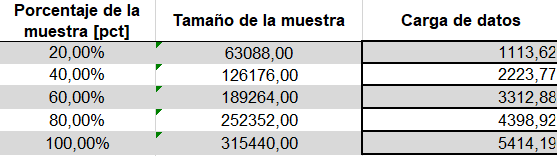
Procesador: Ryzen 7 7730U Radeon Graphics

RAM: 16GB

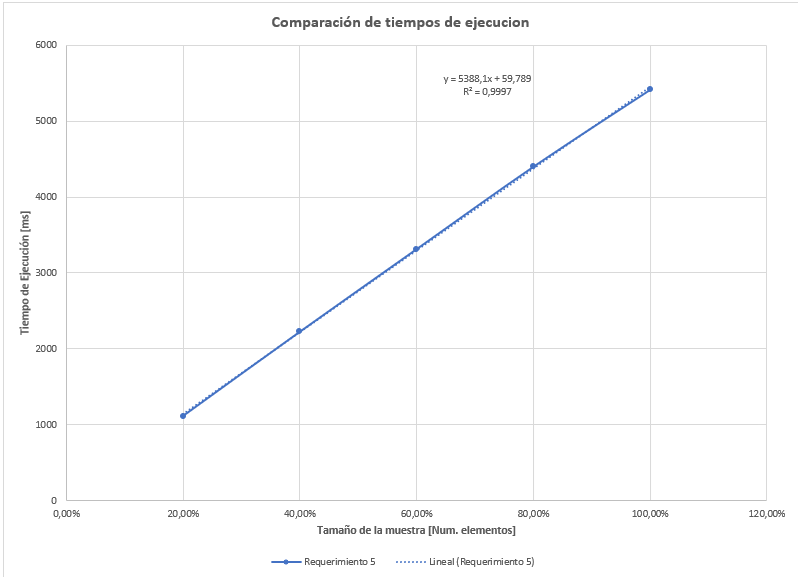
SO: Windows 11

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| 20% | 1113,62 |
| 40% | 2223,77 |
| 60% | 3312,88 |
| 80% | 4398,92 |
| 100% | 5414,19 |

### **Tablas de datos**



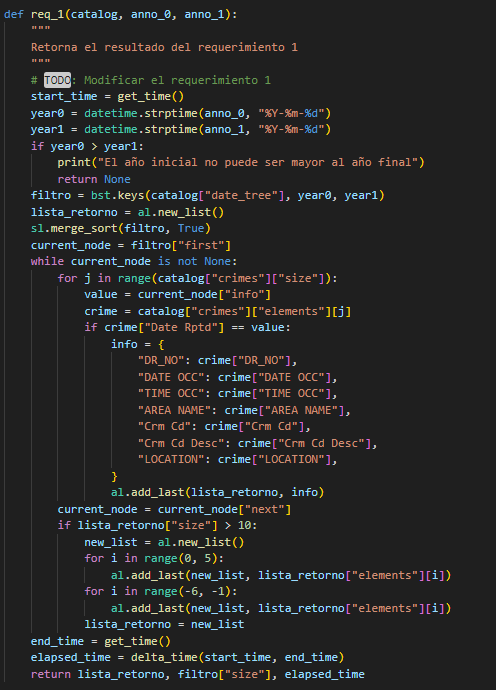
### **Graficas**



## **Análisis**

La grafica confirma la complejidad temporal propuesta anteriormente O(n). Esto se debe a que la longitud de su ejecución depende de la cantidad de datos que contienen los registros para cargarse en su estructura asignada.

# **Requerimiento 1**



Este requerimiento se encarga de recibir dos fechas para las cuales retorna todos los registros que están dentro de este periodo de tiempo. Primero se formatean las fechas, después se descartan fechas invalidas, se filtran y organizan los datos, se crea una lista de retorno la cual extrae los registros relevantes para el criterio, finalmente se acorta la lista si es necesario.

## **Descripción**

|  |  |
| --- | --- |
| **Entrada** | Catálogo, fecha inicio, fecha fin |
| **Salidas** | Reportes dentro del periodo de tiempo |
| **Implementado (Sí/No)** | Si, Juan Diego García |

## **Análisis de complejidad**

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Formatea los valores de fecha y verifica validez | O(1) |
| Extrae las llaves del mapa | O(n) |
| Organiza los datos | O(n log n) |
| Extrae reportes relevantes a una lista | O(n) |
| ***TOTAL*** | ***O(n log n)*** |

## **Pruebas Realizadas**

Procesador: Ryzen 7 7730U Radeon Graphics

RAM: 16GB

SO: Windows 11

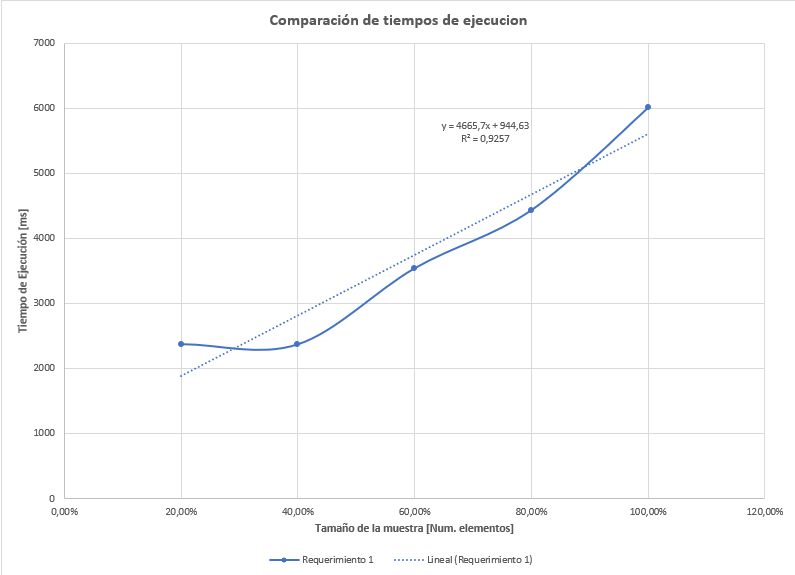
Fecha inicio: 2020-01-01

Fecha fin: 2020-05-02

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| 20% | 2372,80 |
| 40% | 2369.22 |
| 60% | 3538.62 |
| 80% | 4432.99 |
| 100% | 6006.62 |

### **Tablas de datos**

### **Graficas**



## **Análisis**

Es posible evidenciar en la grafica unos cambios notables en el comportamiento de el tiempo de ejecución en cuanto a la cantidad de datos. Esto puede deberse a la complejidad O(n log n) de merge\_sort y al recorrido que se le da al mapa para extraer las llaves de sus respectivos valores y así realizar comparaciones sencillamente.

# **Requerimiento 3**



Este requerimiento se encarga de retornar una cantidad de registros especificada en un área específica desde el catálogo. Se guardan en una variable todos los registros que coinciden con el área deseada por el usuario. Se organiza la lista con merge\_sort y se corta a la longitud deseada. Finalmente se guardan todos los datos del registro que coinciden con el criterio en una lista final.

## **Descripción**

|  |  |
| --- | --- |
| **Entrada** | Catálogo, numero de registros deseado, nombre de un área. |
| **Salidas** | Numero de reportes deseado para el área elegida. |
| **Implementado (Sí/No)** | Si, Tomas Aponte |

## **Análisis de complejidad**

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Obtiene los valores del mapa | O(n) |
| Organiza los datos | O (n log n) |
| Acorta la lista | O(1) |
| Extrae reportes relevantes a una lista | O(n) |
| ***TOTAL*** | ***O (n log n)*** |

## **Pruebas Realizadas**

Procesador: Ryzen 7 7730U Radeon Graphics

RAM: 16GB

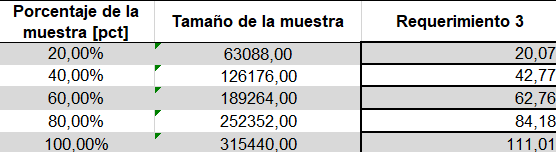
SO: Windows 11

Nombre de área = West LA

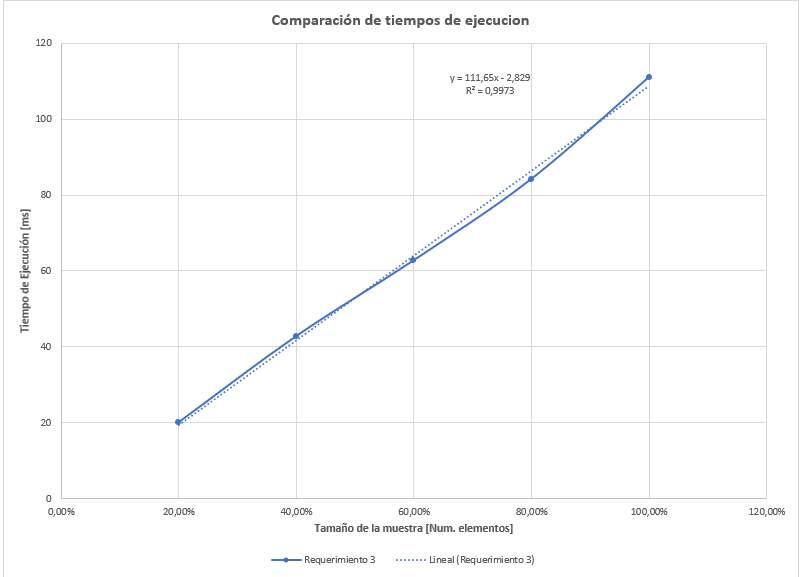
Numero de registros = 10

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| 20% | 20,07 |
| 40% | 42,77 |
| 60% | 62,76 |
| 80% | 84,18 |
| 100% | 111,01 |

### **Tablas de datos**



### **Graficas**



## **Análisis**

En esta grafica se puede evidenciar un claro comportamiento O(n) esto se debe a que la mayoría de los procesos llevados a cabo en el algoritmo son de esta complejidad temporal. Al ser merge\_sort estable no afecta mucho la complejidad temporal y garantiza resultados constantes.

# **Requerimiento 5**



Este requerimiento se encarga de retornar una cantidad de áreas definida por el usuario dentro de un margen de tiempo especificado. El código formatea los datos de fecha, verifica su validez, obtiene los datos relevantes del mapa y los inserta en una lista. Después, se organizan los datos y finalmente se acorta al tamaño deseado.

## **Descripción**

|  |  |
| --- | --- |
| **Entrada** | Catálogo, numero de registros deseado, nombre de un área. |
| **Salidas** | Numero de reportes deseado para el área elegida. |
| **Implementado (Sí/No)** | Si, Tomas Aponte |

## **Análisis de complejidad**

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Formatea y valida la fecha | O(1) |
| Extrae los elementos útiles del mapa | O(n) |
| Recorre la lista en búsqueda de coincidencias | O (n log n) |
| Extrae reportes relevantes a una lista | O(n) |
| Organiza y recorta los registros | O(n log n) |
| ***TOTAL*** | ***O (n log n)*** |

## **Pruebas Realizadas**

Procesador: Ryzen 7 7730U Radeon Graphics

RAM: 16GB

SO: Windows 11

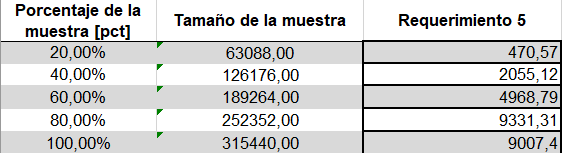
Fecha inicio: 2020-01-10

Fecha fin: 2020-06-06

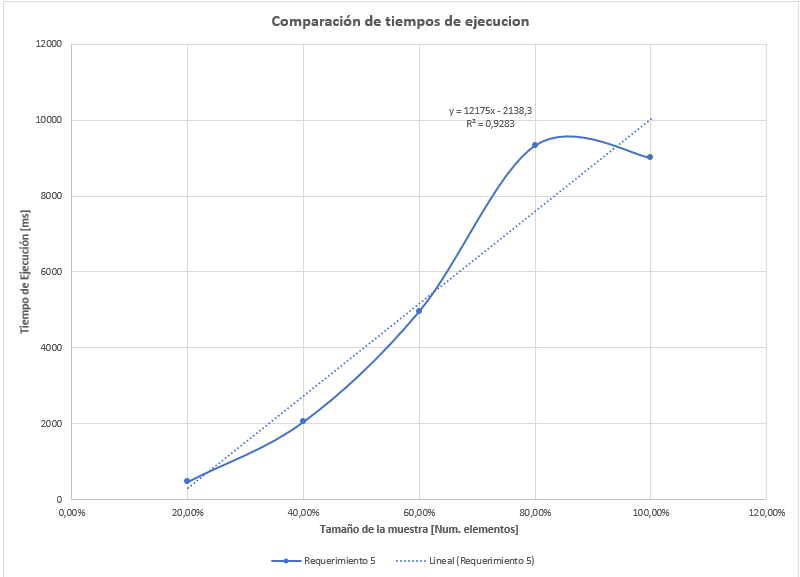
Numero de áreas: 20

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| 20% | 470,57 |
| 40% | 2055,12 |
| 60% | 4968,79 |
| 80% | 9331,31 |
| 100% | 9007,40 |

### **Tablas de datos**



### **Graficas**



## **Análisis**

Esta es la grafica que tiene el comportamiento más ambiguo en la creación de los requisitos. Se podría atribuir el recorrido de la lista a su comportamiento, así como el ordenamiento de datos pero no parece haber nada fuera de lo normal que pudiera causar este cambio repentino en su estructura.

# **Requerimiento 6**



El requerimiento recibe un numero de áreas y un margen de tiempo para el que debe obtener los registros que cumplan el criterio propuesto. Se encarga de hacer esto obteniendo los valores oportunos de un mapa con la información de los registros, creando una lista para probar los requisitos, guardando los registros que aciertan con el criterio, organizando los registros y retornando el resultado acortado por área.

## **Descripción**

|  |  |
| --- | --- |
| **Entrada** | Catálogo, numero de área deseados, rango de tiempo. |
| **Salidas** | Numero de áreas deseado, información de los registros acordes. |
| **Implementado (Sí/No)** | Si, Juan Diego García |

## **Análisis de complejidad**

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Extrae las llaves del mapa | O(n) |
| Recorre la lista en busca de coincidencias | O(n) |
| Organiza los datos | O (n log n) |
| Extrae reportes relevantes a una lista y la acorta | O(n) |
| ***TOTAL*** | ***O (n log n)*** |

## **Pruebas Realizadas**

Procesador: Ryzen 7 7730U Radeon Graphics

RAM: 16GB

SO: Windows 11

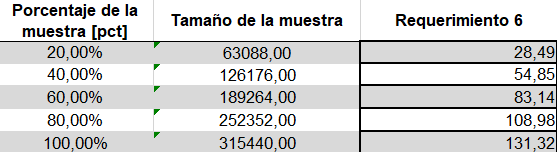
Mes: 05

Género: F

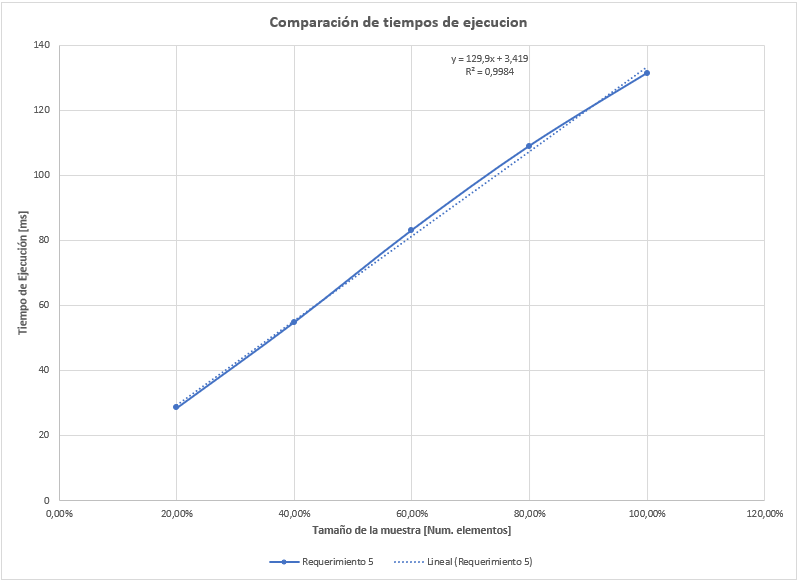
Numero de áreas: 15

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| 20% | 28,49 |
| 40% | 54.85 |
| 60% | 83.14 |
| 80% | 108.98 |
| 100% | 131.32 |

### **Tablas de datos**



### **Graficas**



## **Análisis**

La grafica demuestra el comportamiento definido por los procesos que se ejecutan dentro del código del requisito. La leve curva que se observa en la grafica puede ser gracias a el recorrido de el diccionario en búsqueda de registros que coincidieran con el criterio, la extracción de llaves del mapa o la organización de los datos. Pero pareciera tender a O(n) mas que otra cosa.

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| 20% |  |
| 40% |  |
| 60% |  |
| 80% |  |
| 100% |  |