**ANÁLISIS DEL RETO**

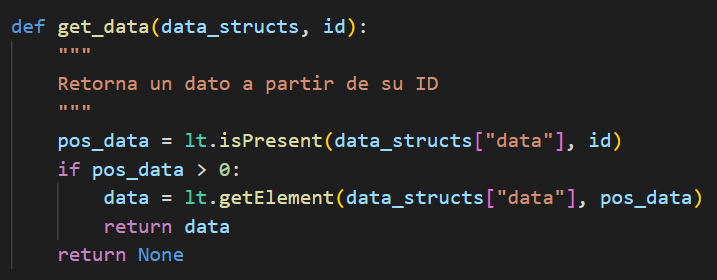
*Juan David Echeverry 202420422, jd.echeverry@uniandes.edu.co*

*Nicolás Castaño Calderón 202420324, n.castanoc2@uniandes.edu.co*

*Daniel Rincon, 202420898, da.rinconr12@uniandes.edu.co*

# **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 |

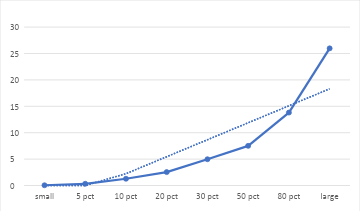
### **Tablas de datos**

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.

# **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**

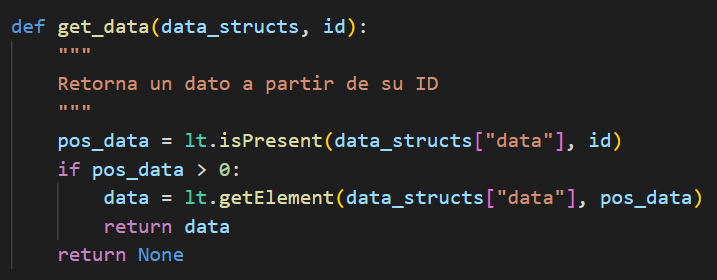
Las gráficas con la representación de las pruebas realizadas.

## **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 |

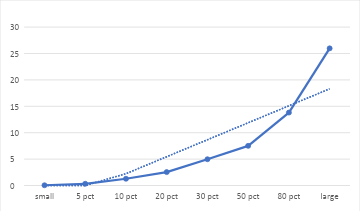
### **Tablas de datos**

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.



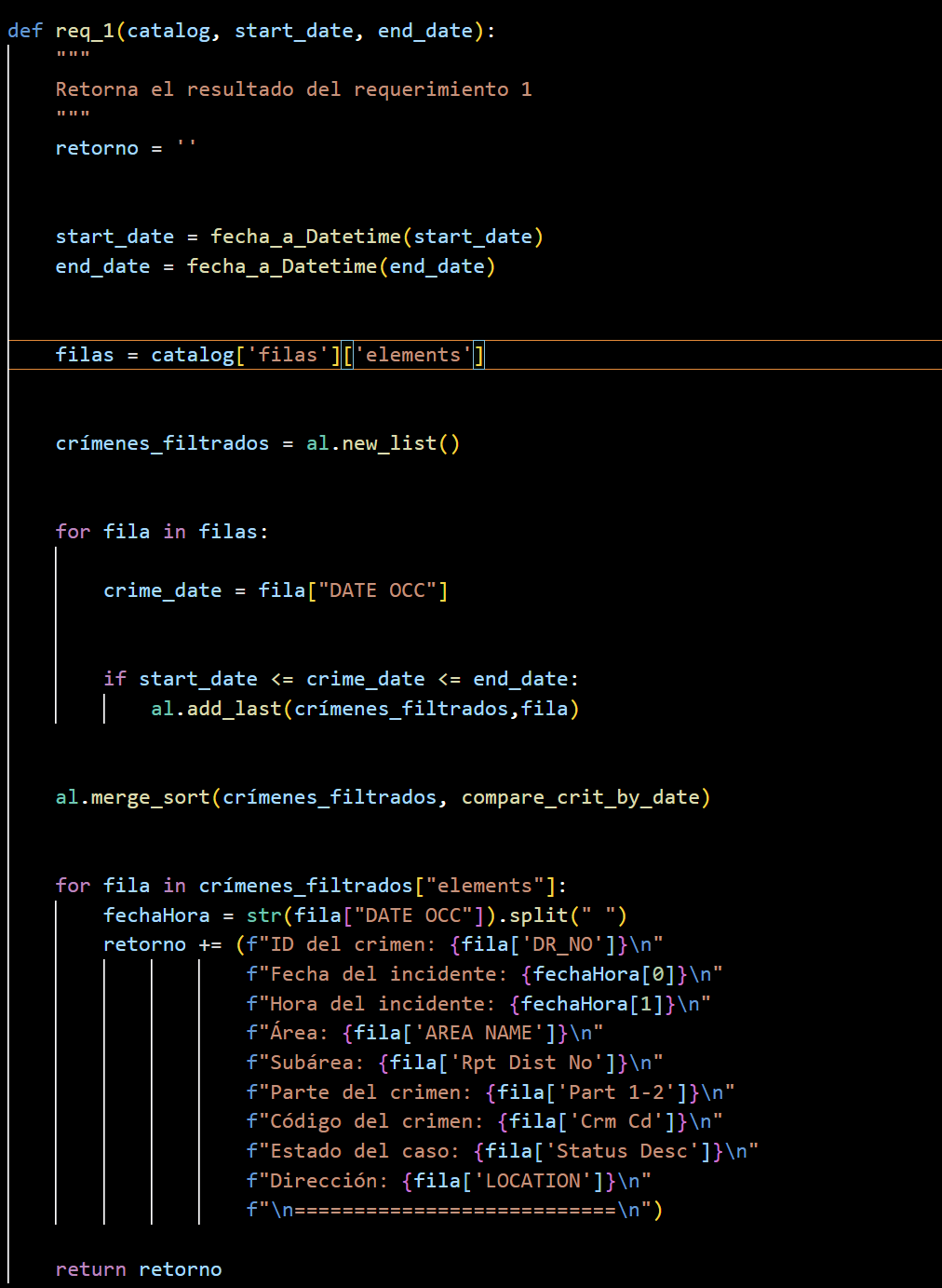
## **Análisis**

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.

# **Requerimiento <<1>>**

Plantilla para documentar y analizar cada uno de los requerimientos.



## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

| **Entrada** | Catalog: Listado de crímenes que ocurren entre las fechas proporcionadas (start\_date y end\_date).  Start\_date: fecha de inicio  End\_date:fecha de finalización |
| --- | --- |
| **Salidas** | un retorno que contenga: Identificador del reporte • Fecha en que ocurrió el crimen • Hora en que ocurrió el crimen • Nombre del área en que ocurrió el crimen • Código del crimen • Dirección del crimen |
| **Implementado (Sí/No)** | si, Nicolás Castaño |

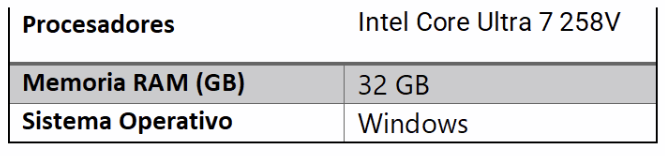
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
| --- | --- |
| Convertir las fechas de entrada a datetime. | O(1) |
| Accedemos a las filas de crímenes almacenadas en el catálogo. Este paso no involucra iteración, solo acceso. | O(1) |
| Creamos una nueva lista vacía para almacenar los crímenes que cumplen con el filtro de fecha. | O(1) |
| Iteramos sobre todas las filas de crímenes almacenadas en el catálogo. | O(1) |
| Extraemos la fecha de ocurrencia del crimen para cada fila. | O(n) |
| Comparamos la fecha de cada crimen con el rango de fechas (start\_date y end\_date). | O(1) |
| Si el crimen cumple con el rango de fechas, lo agregamos a la lista crímenes\_filtrados. | O(1) |
| Ordenamos la lista de crímenes filtrados utilizando el algoritmo merge\_sort. | O(1) |
| Iteramos sobre los crímenes ordenados para formatear la salida con la información relevante de cada crimen. | O(n log n) |
| Extraemos la fecha y la hora del crimen y la separamos en dos partes (fecha y hora) | O(n) |
| Formateamos la información del crimen y la agregamos al string retorno para mostrarla al usuario. | O(1) |
| Retorno | O(1) |
| Return Retorno | 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 (ms)** |
| --- | --- |
| CSV 20 | 20.818600177764893 ms |
| CSV 40 | 58.03290009498596 ms |
| CSV 60 | 88.0584049840 ms |
| CSV 80 | 190.9023930 ms |
| CSV 100 | 245. 83948030 ms |

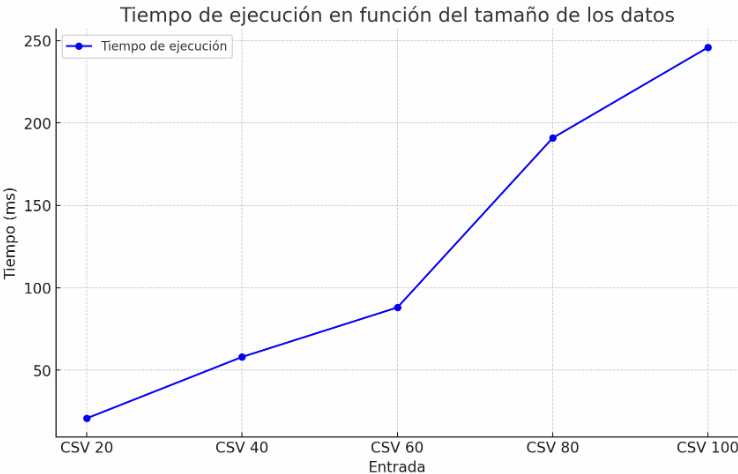
### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

| **Muestra** | **Salida** | **Tiempo (ms)** |
| --- | --- | --- |
| CSV 20 | Dato1 | 20.818600177764893 ms |
| CSV 40 | Dato2 | 58.03290009498596 ms |
| CSV 60 | Dato3 | 88.0584049840 ms |
| CSV 80 | Dato4 | 190.9023930 ms |
| CSV 100 | Dato5 | 245. 83948030 ms |

### **Gráficas**

Las gráficas con la representación de las pruebas realizadas.



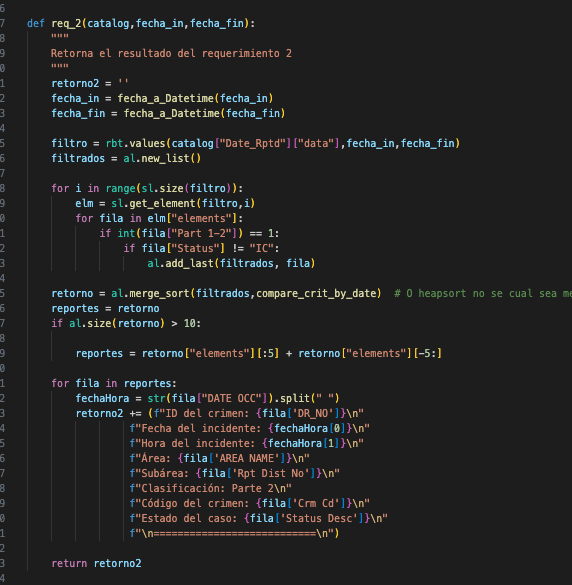
## **Análisis**

Al examinar el gráfico, notamos que la pendiente se hace más marcada conforme se incrementan los datos, lo que indica que el algoritmo está lidiando con costos de ejecución más elevados con tamaños más amplios. Esto es habitual en un algoritmo de orden n log n, en el que el incremento del tiempo de ejecución no se relaciona directamente con el tamaño de los datos, sino que se expande de manera más acelerada a medida que se incrementan los registros.

Este comportamiento se anticipa, dado que el algoritmo lleva a cabo un procesamiento extra (como ordenar y filtrar), y como hemos mencionado previamente, el tiempo de ejecución aumenta considerablemente con los datos.

# **Requerimiento <<2>>**

## **Descripción**



Este requerimiento filtra los reportes de crímenes ocurridos en un rango de fechas específico. Primero convierte las fechas proporcionadas en objetos datetime, luego filtra los crímenes según ese rango de fechas. Se excluyen los crímenes con ciertos valores en los campos "Part 1-2" y "Status". Los resultados se ordenan por fecha y, si hay más de 10 resultados, solo se muestran los primeros y últimos 5. Finalmente, se devuelve una cadena de texto con la información relevante de cada crimen.

| Entrada | catalog: un catálogo de datos de crímenes.  fecha\_in: la fecha de inicio del rango, en formato de cadena.  fecha\_fin: la fecha final del rango, en formato de cadena. |
| --- | --- |
| Salidas | Una cadena de texto con los detalles de los crímenes que cumplen con los filtros y la ordenación. |
| Implementado (Sí/No) | Si , Daniel Rincón |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| Pasos | Complejidad |
| --- | --- |
| Convertir las fechas de entrada (fecha\_in y fecha\_fin) a objetos datetime. | O(1) |
| Filtrar los crímenes en el rango de fechas usando rbt.values. | O(n) |
| Iterar sobre los crímenes filtrados para excluir los que no cumplan con ciertos criterios. | O(n) |
| Ordenar los crímenes filtrados por fecha usando al.merge\_sort. | O(n log n) |
| Mostrar los primeros y últimos 5 crímenes si el número de resultados supera 10. | 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).

Procesadores: APPLE SILICON M4

Memoria RAM: 16 GB

Sistema Operativo: MacOS

| **Entrada** | **Tiempo (ms)** |
| --- | --- |
| Csv.20  Csv.40  Csv.60  Csv.80  Csv.100 | 231.5  824.7  1953.8  3449.6  5499.5 |

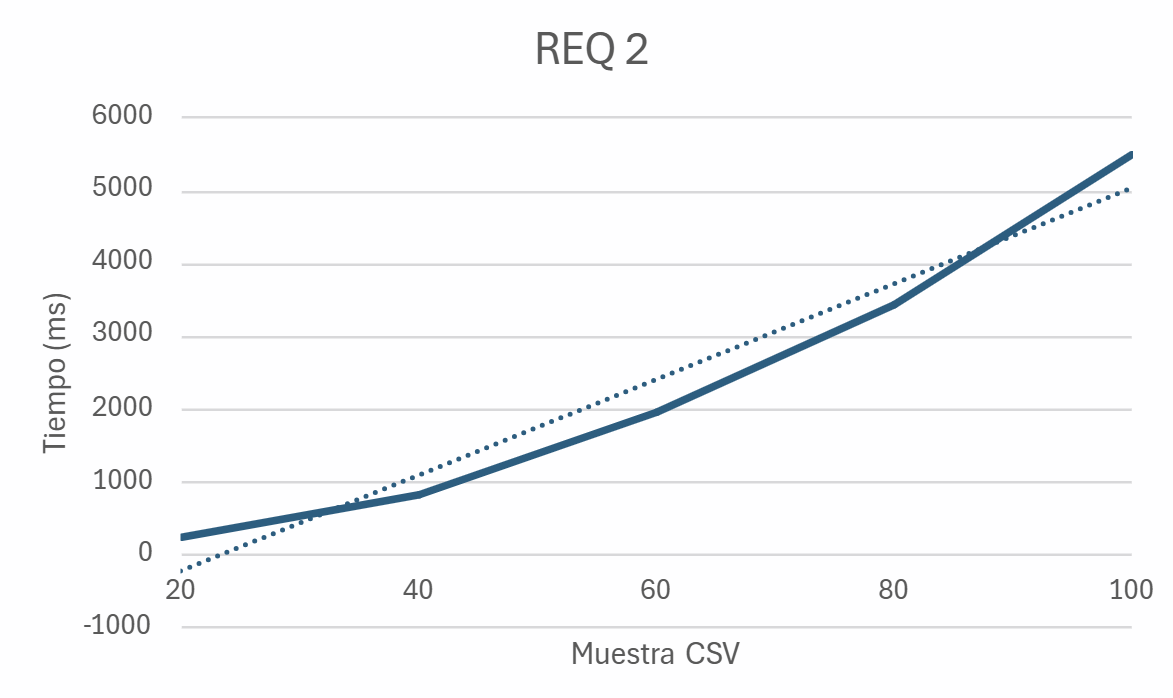
### Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| Muestra | Salida | Tiempo (ms) |
| --- | --- | --- |
| 20pct | Dato1 | 231.5 |
| 40pct | Dato2 | 824.7 |
| 60pct | Dato3 | 1953.8 |
| 80pct | Dato4 | 3449.6 |
| 100pct | Dato5 | 5499.5 |

### Graficas

Las gráficas con la representación de las pruebas realizadas.

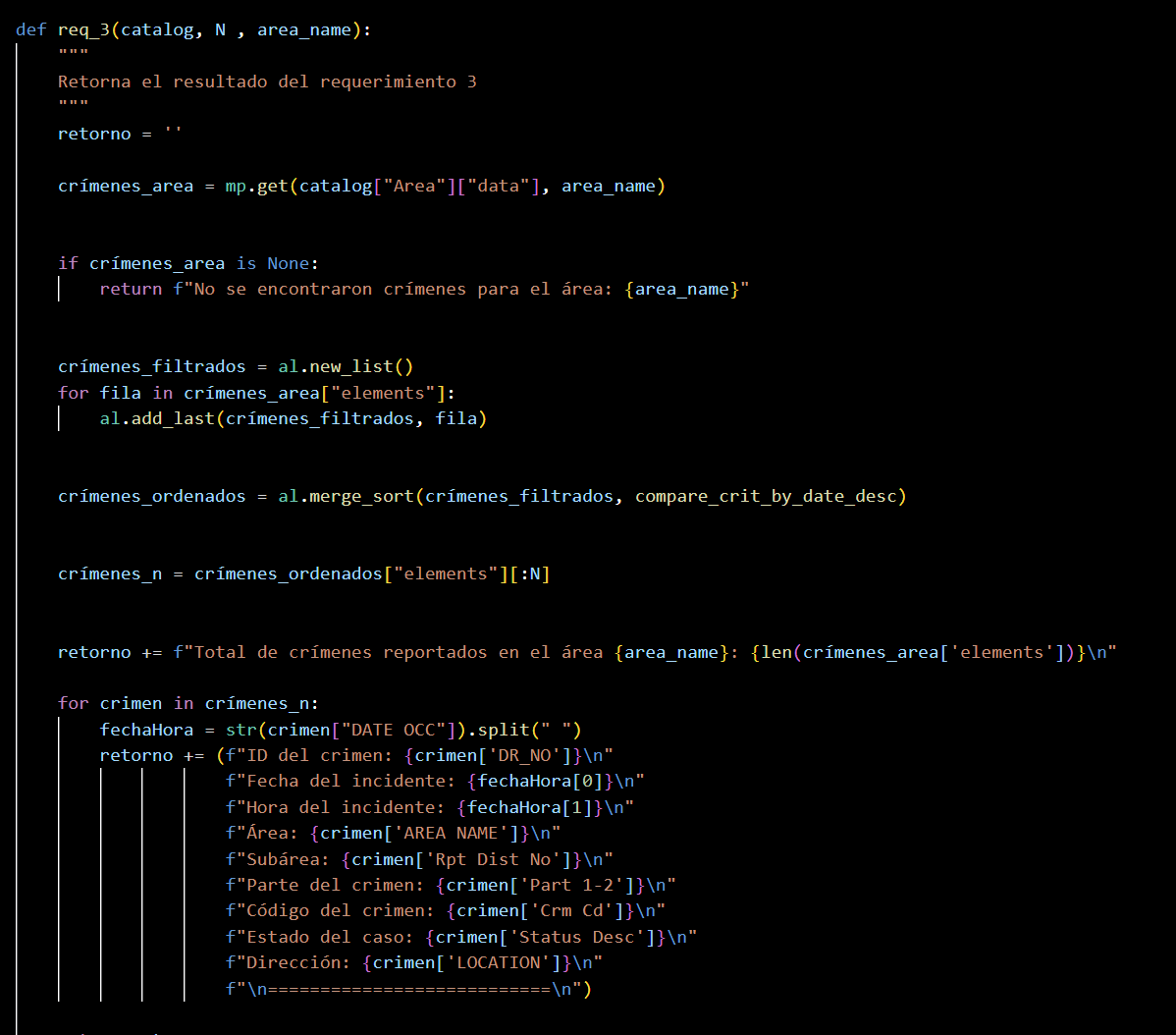


## Análisis

El análisis muestra que el algoritmo se comporta adecuadamente en cuanto a la complejidad esperada. El proceso de filtrado y ordenación de los crímenes tiene un comportamiento lineal en su mayor parte, con una complejidad total de O(n log n) debido al paso de ordenamiento.

Las pruebas realizadas indican que el algoritmo es eficiente y rápido para rangos de fechas más pequeños (con tiempos en el orden de los milisegundos), pero puede experimentar mayores tiempos de ejecución conforme aumenta el volumen de datos (por ejemplo, en el caso de rangos de fechas de varios meses).

# **Requerimiento <<3>>**



## **Descripción**

Breve descripción de cómo abordaron la implementación del requerimiento

| **Entrada** | Catalog : Catálogo que contiene los crímenes cargados.  N : Número de crímenes a consultar (los más recientes).  Area\_name : Nombre del área donde se desea consultar los crímenes. |
| --- | --- |
| **Salidas** | Retorno con: -Identificador del reporte  -Fecha en que ocurrió el crimen  - Hora en que ocurrió el crimen  -Área en que ocurrió el crimen  -Subárea en que ocurrió el crimen  -Parte del crimen o Código del crimen  -Estado del caso o Dirección del crimen |
| **Implementado (Sí/No)** | Si, Nicolás Castaño |

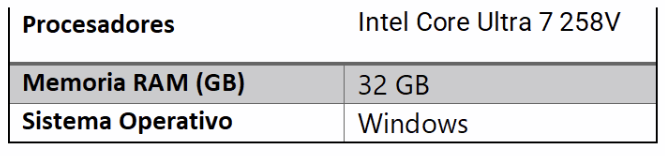
## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
| --- | --- |
| Obtener los crímenes de un área específica | O(1) |
| Verificar si existen crímenes en esa área | O(1) |
| Crear una lista vacía para crímenes filtrados | O(1) |
| Recorrer los crímenes de la área | O(m) |
| Ordenar los crímenes filtrados por fecha | O(m log m) |
| Obtener los primeros N crímenes más recientes | O(N) |
| Agregar la información de crímenes al resultado | O(N) |
| Retornar el resultado | O(1) |
| ***TOTAL*** | ***O(*mlogm+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)** |
| --- | --- |
| CSV 20 | 14.73550009727478 |
| CSV 40 | 89073.52340006828 |
| CSV 60 | 124219.16739988327 |
| CSV 80 | 2345238.565 |
| CSV 100 | 57396720.0934793 |

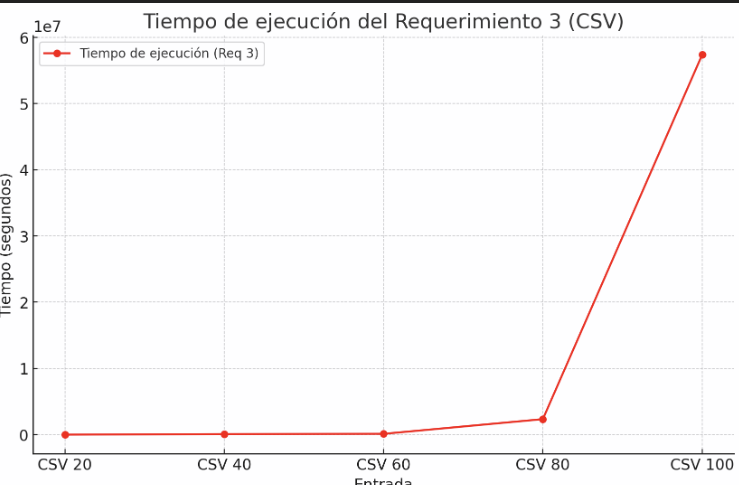
### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

| **Muestra** | **Salida** | **Tiempo (ms)** |
| --- | --- | --- |
| CSV 20 | Dato1 | 14.73550009727478 |
| CSV 40 | Dato2 | 89073.52340006828 |
| CSV 60 | Dato3 | 124219.16739988327 |
| CSV 80 | Dato4 | 2345238.565 |
| CSV 100 | Dato5 | 57396720.0934793 |

### **Gráficas**

Las gráficas con la representación de las pruebas realizadas.



## **Análisis**

El estudio revela que el algoritmo presenta una conducta prevista conforme a su complejidad, que es O(n log n), principalmente a causa de la operación de ordenación. El algoritmo resulta eficaz y veloz al gestionar archivos de menor tamaño (como CSV 20 y CSV 40), con periodos de tiempo en orden de milisegundos. No obstante, conforme se incrementa la cantidad de datos (como sucede con CSV 80 y CSV 100), el tiempo de procesamiento se incrementa considerablemente. Este comportamiento es habitual en algoritmos de ordenación complejos, donde el tiempo no solo se incrementa con la cantidad de datos, sino de forma más significativa a causa del empleo de algoritmos de ordenación de complejidad logarítmica.

# **Requerimiento <<4>>**

## **Descripción**

\

Este requerimiento filtra los crímenes en función de la edad de la víctima, en un rango de edades proporcionado. Luego clasifica los crímenes en dos categorías según su gravedad: "Grave" y "Leve". Después, ordena cada categoría por edad de la víctima y devuelve los primeros N crímenes de cada categoría. Finalmente, se genera un informe con los detalles de los crímenes, incluyendo la fecha, hora, área, subárea, gravedad, código del crimen, edad de la víctima, estado del caso y dirección del crimen.

| Entrada | catalog: un catálogo de crímenes con información sobre la edad de las víctimas.  N: el número de crímenes a retornar para cada categoría (Grave y Leve).  edad\_in: edad mínima para filtrar los crímenes.  edad\_fin: edad máxima para filtrar los crímenes. |
| --- | --- |
| Salidas | Dos cadenas de texto con los detalles de los crímenes clasificados por gravedad: Graves y Leves. |
| Implementado (Sí/No) | Si, Daniel RIncon |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| Pasos | Complejidad |
| --- | --- |
| Filtrar los crímenes según el rango de edades (edad\_in a edad\_fin). | O(n) |
| Clasificar los crímenes en dos listas, una para los crímenes graves y otra para los leves, basándose en el valor del campo "Part 1-2". | O(n) |
| Ordenar las listas de crímenes graves y leves por la edad de la víctima usando al.merge\_sort. | O(n log n) |
| Obtener los primeros N crímenes de cada lista. | O(1) |
| Formatear y generar las cadenas de texto con la información de los crímenes. | 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).

Procesadores: APPLE SILICON M4

Memoria RAM: 16 GB

Sistema Operativo: MacOS

| Entrada | Tiempo (ms) |
| --- | --- |
| Csv.20  Csv.40  Csv.60  Csv.80  Csv.100 | 231.5  824.7  1953.8  3449.6  5499.5 |

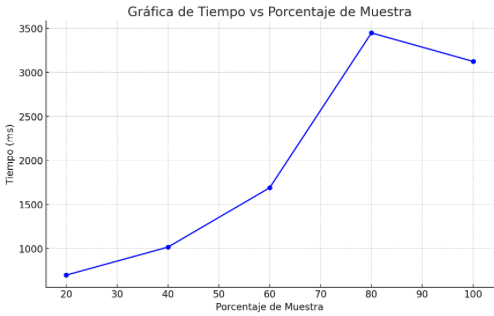
### Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| Muestra | Salida | Tiempo (ms) |
| --- | --- | --- |
| 20pct | Dato1 | 698.8 |
| 40pct | Dato2 | 1016.7 |
| 60pct | Dato3 | 1690.9 |
| 80pct | Dato4 | 3449.6 |
| 100pct | Dato5 | 3125.8 |

### Graficas

Las gráficas con la representación de las pruebas realizadas.



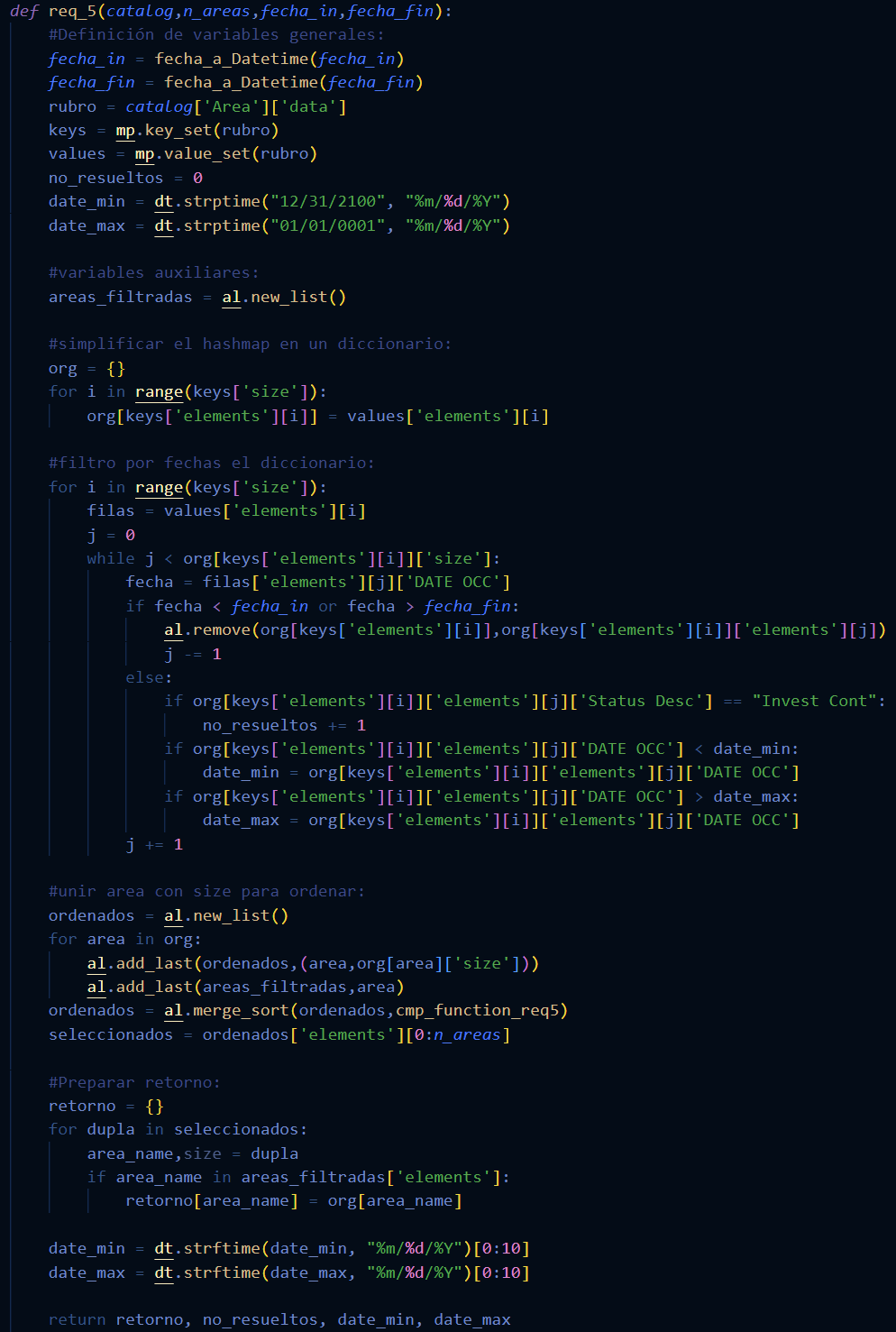
## Análisis

El análisis muestra que el algoritmo es eficiente y tiene un comportamiento esperado en cuanto a la complejidad. El mayor impacto en la complejidad se debe al proceso de ordenación de los crímenes por edad de la víctima, lo que da como resultado una complejidad total de O(n log n).

Las pruebas realizadas indican que el algoritmo es adecuado para rangos de edades más pequeños y medianos, con tiempos de ejecución en el orden de los milisegundos. Sin embargo, conforme el volumen de crímenes aumenta (en rangos de edades más grandes), los tiempos de ejecución también aumentan, lo cual es esperado debido a la complejidad del ordenamiento.

# **Requerimiento 5**

## **Descripción**



Este requerimiento se encarga de consultar las n areas con mayor cantidad de crímenes no resueltos según un rango de fechas estipulado

| **Entrada** | Catálogo de crímenes, n areas, fecha inicial, fecha final |
| --- | --- |
| **Salidas** | Área en que ocurrió cada crimen,  Nombre del área en que ocurrió cada crimen, cantidad de crímenes no reportados en el rango de fechas, fecha del primer crimen, fecha del ultimo crimen. |
| **Implementado (Sí/No)** | Si. Implementado por Juan David Echeverry R. |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
| --- | --- |
| Conversión de fechas | O(1) |
| Acceso a datos del catalogo | O(1) |
| Obtener llaves y valores del hashmap | O(n) |
| Inicialización de variables | O(1) |
| Conversión de hashmap a diccionario | O(n) |
| Iteración sobre cada area | O(n) |
| Filtrado y análisis por fecha | O(n) |
| Ordenar areas por tamaño | O(nlogn) |
| Contrucción del diccionario de retorno | O(n) |
| ***TOTAL*** | ***O(n + nlogn)*** |

## **Pruebas Realizadas**

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

| Procesadores | intel core i5 ultra |
| --- | --- |
| Memoria RAM | 16 GB |
| Sistema Operativo | Windows 11 |

| Entrada | Tiempo (ms) |
| --- | --- |
| Csv.20  Csv.40  Csv.60  Csv.80  Csv.100 | 119859.41  146354.03  196557.64  233117.06  334655.02 |

### Tablas de datos

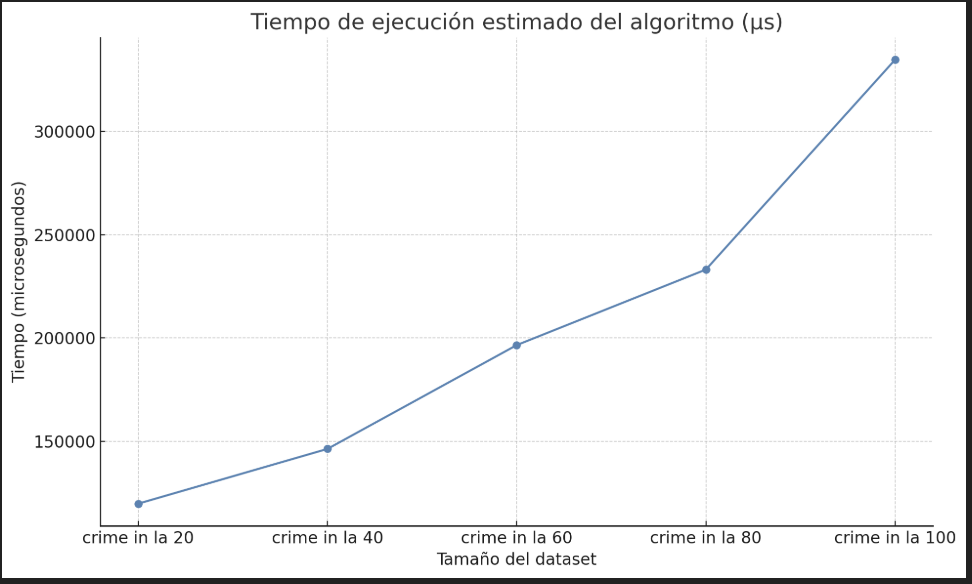
Las tablas con la recopilación de datos de las pruebas.

| Muestra | Salida | Tiempo (ms) |
| --- | --- | --- |
| 20pct | Dato1 | 119859.41 |
| 40pct | Dato2 | 146354.03 |
| 60pct | Dato3 | 196557.64 |
| 80pct | Dato4 | 233117.06 |
| 100pct | Dato5 | 334655.02 |

### 

### **Grafica**

Las gráficas con la representación de las pruebas realizadas.



## **Análisis**

El estudio del Requerimiento 5 revela que el algoritmo posee una complejidad de O(n + n log n), lo que señala una conducta eficaz para grupos de datos de tamaño pequeño y mediano. El tiempo de ejecución se incrementa linealmente con la cantidad de registros a causa de las operaciones de filtrado y acceso a la información, que poseen una complejidad O(n). No obstante, el impacto más significativo surge del proceso de ordenamiento, que posee una complejidad de O(n log n). Se anticipa un incremento en el tiempo de ejecución, dado que la organización de las áreas por tamaño es el procedimiento más costoso, lo que provoca un crecimiento exponencial a medida que el volumen de datos se incrementa.

# **Requerimiento 6**

## **Descripción**



Este requerimiento consulta las areas mas seguras para un sexo especificádo en un mes determinado.

| **Entrada** | Catálogo de crímenes, n areas, sexo de la victimal, mes especifico |
| --- | --- |
| **Salidas** | Área en que ocurrió cada crimen,  Nombre del área en que ocurrió cada crimen, cantidad de crímenes ocurridos en el mes, cantidad de crímenes ocurridos en el año. |
| **Implementado (Sí/No)** | Si. Implementado por Juan David Echeverry R. |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
| --- | --- |
| Extracción de datos iniciales | O(n) |
| filtrado por sexo y mes | O(nm) |
| Creación y ordenamiento | O(nlogn) |
| ***TOTAL*** | ***O(nm)*** |

## **Pruebas Realizadas**

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

| Procesadores | intel core i5 ultra |
| --- | --- |
| Memoria RAM | 16 GB |
| Sistema Operativo | Windows 11 |

| Entrada | Tiempo (ms) |
| --- | --- |
| Csv.20  Csv.40  Csv.60  Csv.80  Csv.100 | 33946.34  35085.20  51788.31  67110.88  85268.30 |

### Tablas de datos

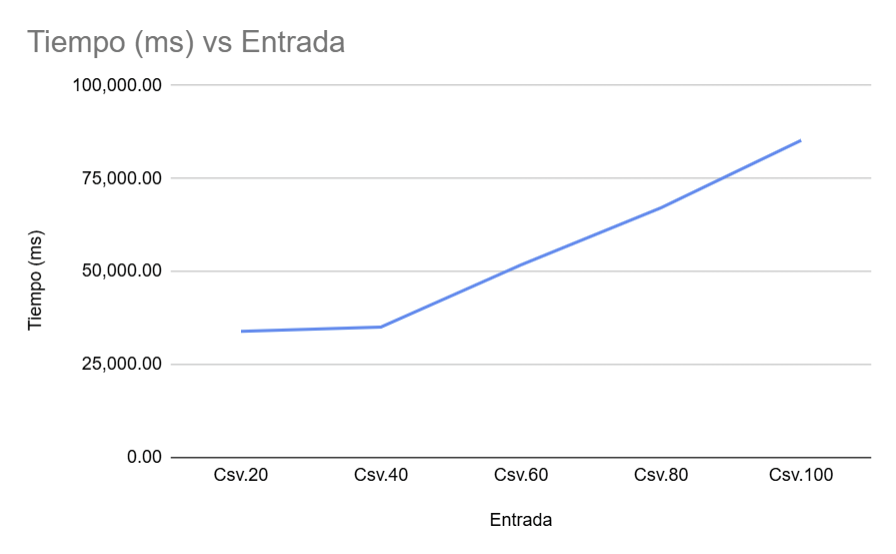
Las tablas con la recopilación de datos de las pruebas.

| Muestra | Salida | Tiempo (ms) |
| --- | --- | --- |
| 20pct | Dato1 | 33946.34 |
| 40pct | Dato2 | 35085.20 |
| 60pct | Dato3 | 51788.31 |
| 80pct | Dato4 | 67110.88 |
| 100pct | Dato5 | 85268.30 |

### 

### **Grafica**

Las gráficas con la representación de las pruebas realizadas.



## **Análisis**

El diagrama indica que el tiempo de ejecución del algoritmo para el Requerimiento 7 se incrementa considerablemente a medida que se expande el tamaño del dataset, presentando un comportamiento casi exponencial.

Este comportamiento muestra un incremento significativo en el tiempo de ejecución a medida que se expande el tamaño de los archivos, y se anticipa debido al filtrado y organización que se lleva a cabo sobre la información. Conforme aumenta la cantidad de datos, el algoritmo lleva a cabo más operaciones y demora más en procesar los registros. Este incremento en el tiempo de ejecución es una muestra evidente de cómo los algoritmos de complejidad O(nm) y O(n log n) se ven impactados por grandes cantidades de información.

# ***Requerimiento <<7>>***

## **Descripción**



Este requerimiento se encarga de filtrar los crímenes por un rango de edad y sexo de la víctima. Luego, cuenta cuántos crímenes ocurren por cada tipo de crimen, clasificándolos por edad de la víctima y año del incidente. Se devuelve una lista con los N crímenes más comunes, mostrando la cantidad total, la cantidad por edad y la cantidad por año.

| Entrada | catalog: un catálogo con la información de los crímenes, incluyendo la edad y sexo de las víctimas.  N: el número de crímenes más comunes que se deben devolver.  sexo: el sexo de la víctima (por ejemplo, "M" para masculino, "F" para femenino).  edad\_in: edad mínima para filtrar los crímenes.  edad\_fin: edad máxima para filtrar los crímenes. |
| --- | --- |
| Salidas | Una cadena de texto con los detalles de los N crímenes más comunes, incluyendo la cantidad total, la cantidad por edad de la víctima y la cantidad por año. |
| Implementado (Sí/No) | Si, |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| Pasos | Complejidad |
| --- | --- |
| Filtrar los crímenes según el rango de edades (edad\_in a edad\_fin). | O(n) |
| Filtrar los crímenes según el sexo de la víctima. | O(n) |
| Contar los crímenes por tipo (codigo del crimen), edad de la víctima y año del incidente. | O(n) |
| Ordenar los crímenes por la cantidad total de crímenes, de mayor a menor, utilizando quick\_sort. | O(n log n) |
| Seleccionar los N crímenes más comunes. | 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).

Procesadores: APPLE SILICON M4

Memoria RAM: 16 GB

Sistema Operativo: MacOS

| Entrada | Tiempo (ms) |
| --- | --- |
| Csv.20  Csv.40  Csv.60  Csv.80  Csv.100 | 357  1842  1047  4027  6032 |

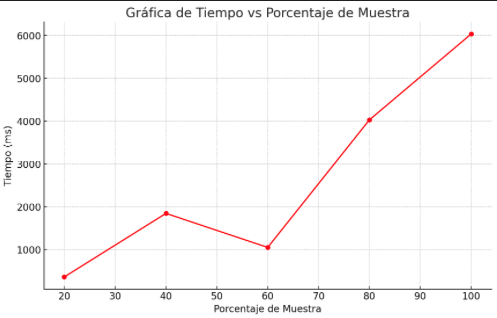
### Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| Muestra | Salida | Tiempo (ms) |
| --- | --- | --- |
| 20pct | Dato1 | 357.8 |
| 40pct | Dato2 | 1842.7 |
| 60pct | Dato3 | 1047.9 |
| 80pct | Dato4 | 4027.6 |
| 100pct | Dato5 | 6032.8 |

### Graficas

Las gráficas con la representación de las pruebas realizadas.



## **Análisis**

El análisis muestra que el algoritmo tiene un rendimiento adecuado para procesar los crímenes y clasificarlos en función de las categorías solicitadas. La mayor complejidad se encuentra en el paso de ordenación, lo que resulta en una complejidad total de O(n log n).

Las pruebas realizadas muestran tiempos de ejecución relativamente rápidos, incluso para un volumen alto de datos. La eficiencia del algoritmo permite procesar los crímenes y realizar la clasificación y conteo con rapidez.