**ANÁLISIS DEL RETO**

Juan Camilo Cancelado, 202410123, j.canceladod@uniandes.edu.co

Pedro Archila, p.archila@uniandes.edu.co

Gabriela Gomez, g,gomezh2@uniandes.edu.co

# **Requerimiento 5**

## **Descripción**

Consultar las N áreas con mayor cantidad de crímenes no resueltos ocurridos en un rango de fechas

|  |  |
| --- | --- |
| **Entrada** | • La cantidad N de áreas a consultar  • La fecha inicial del periodo a consultar  • La fecha final del periodo a consultar |
| **Salidas** | • Área en que ocurrió el crimen  • Nombre del área en que ocurrió el crimen  • Cantidad de crímenes no resueltos reportados en el rango de fechas  • Fecha del primer crimen  • Fecha del último crimen |
| **Implementado (Sí/No)** | Si se implementó, Juan Camilo Cancelado |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Paso** | **Complejidad y Variables** |
| **1. Parseo de fechas de entrada** | **O(1)**: Las fechas inicial y final se convierten a objetos datetime. No depende del tamaño de los datos. |
| **2. Obtener crímenes en el rango de fechas** | **O(log(n) + k)**: Se realiza una búsqueda en el árbol rojo-negro fecha\_occ para obtener los valores en el rango. Aquí, n es el número total de fechas en el árbol y k es el número de fechas en el rango. |
| **3. Filtrar crímenes no resueltos** | **O(m)**: Se itera sobre los crímenes obtenidos en el rango de fechas (k fechas con m crímenes en total) y se filtran aquellos con estado "IC". |
| **4. Crear el árbol**area\_crime\_count | **O(1)**: Inicialización del árbol rojo-negro para almacenar la cantidad de crímenes por área. |
| **5. Iterar sobre los crímenes no resueltos** | **O(m \* log(a))**: Para cada crimen no resuelto (m crímenes), se actualizan los datos en el árbol area\_crime\_count. Aquí, a es el número de áreas únicas. |
| **6. Actualizar fechas de primer y último crimen por área** | **O(m \* log(a))**: Para cada crimen, se comparan y actualizan las fechas de primer y último crimen en el árbol area\_crime\_count. |
| **7. Obtener valores del árbol**area\_crime\_count | **O(a)**: Se obtienen los valores del árbol, que son mapas con información de cada área. |
| **8. Ordenar las áreas por cantidad de crímenes** | **O(a \* log(a))**: Se utiliza merge\_sort para ordenar las áreas según el criterio definido. Aquí, a es el número de áreas únicas. |
| **9. Obtener las N áreas con más crímenes** | **O(N)**: Se utiliza ar.sub\_list para obtener las primeras N áreas de la lista ordenada. |
| **10. Preparar la respuesta** | **O(N)**: Se formatea la información de las N áreas con más crímenes para la salida. |
| **TOTAL** | **O(log(n) + k + m \* log(a) + a \* log(a) + N)** |

## **Pruebas Realizadas**

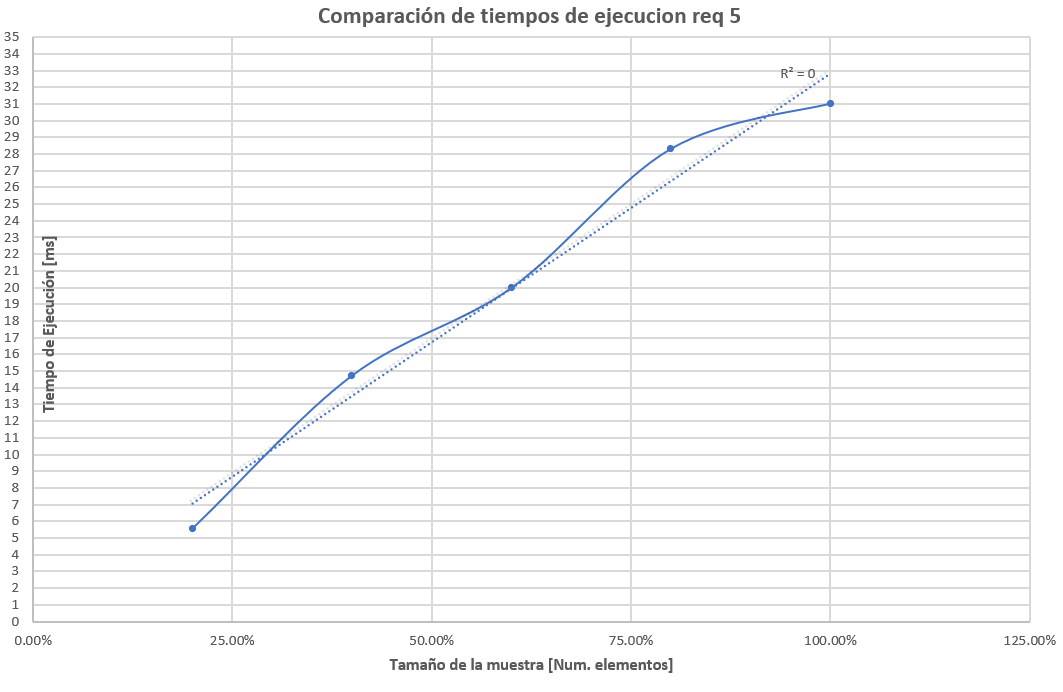
Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron N = 5, f\_i = 2020-01-01, f\_f = 2020-01-03.

|  |  |
| --- | --- |
| Procesadores | Intel core i7 7th gen |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 10 |

### **Tablas de datos**

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| 20 pct | 5.6 |
| 40 pct | 14.74 |
| 60 pct | 19.99 |
| 80 pct | 28.31 |
| 100 pct | 31.02 |

### **Graficas**



## **Análisis**

# El comportamiento cercano a O(N) en el requerimiento 5 puede explicarse porque las operaciones lineales, como el filtrado de crímenes no resueltos y la iteración sobre los datos, dominan el tiempo de ejecución, especialmente cuando el número de áreas únicas (a) o el rango de fechas (k) es pequeño en comparación con el número total de crímenes procesados (m). Además, las estructuras como árboles rojo-negro y mapas están optimizadas, lo que reduce el impacto de las operaciones logarítmicas. Esto, combinado con una distribución uniforme de los datos, hace que el tiempo de ejecución sea proporcional al tamaño de los datos procesados, asemejándose a un comportamiento O(N) en la práctica.

# **Requerimiento 7**

## Determinar los crímenes más comunes para las víctimas de un sexo en un rango de edad dado

## **Descripción**

|  |  |
| --- | --- |
| **Entrada** | • El número N de los crímenes más comunes a calcular  • El sexo de la víctima  • Edad inicial del rango a consultar (con formato entero en años)  • Edad final del rango a consultar (con formato entero en años) |
| **Salidas** | o Código el crimen  o Cantidad de crímenes cometidos  o Cantidad de crímenes ocurridos por edad de la víctima, en formato (crímenes, edad)  o Cantidad de crímenes ocurridos para cada año, en formato (crímenes, año) |
| **Implementado (Sí/No)** | Si se implementó y quien lo hizo. |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Paso** | **Complejidad y Variables** |
| **Parseo de edades de entrada** | **O(1)**: |
| **Obtener crímenes en el rango de edades** | **O(log(n) + k)**: Se realiza una búsqueda en el árbol rojo-negro [edad](vscode-file://vscode-app/c:/Users/juank/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) para obtener los valores en el rango. Aquí, [n](vscode-file://vscode-app/c:/Users/juank/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) es el número total de edades en el árbol y [k](vscode-file://vscode-app/c:/Users/juank/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) es el número de edades en el rango. |
| **Filtrar crímenes por sexo de la víctima** | **O(m)**: Se itera sobre los crímenes obtenidos en el rango de edades ([k](vscode-file://vscode-app/c:/Users/juank/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) edades con m crímenes en total) y se filtran por el sexo de la víctima. |
| **Crear un mapa para contar crímenes por código** | **O(1)** |
| **Iterar sobre los crímenes filtrados** | **O(m \* log(c))**: Para cada crimen filtrado (m crímenes), se actualizan los datos en el mapa [crime\_count\_map](vscode-file://vscode-app/c:/Users/juank/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html). Aquí, c es el número de códigos únicos de crímenes. |
| **Actualizar conteo de crímenes por edad** | **O(m \* log(e))**: Para cada crimen, se actualiza el conteo en el árbol rojo-negro age\_count. Aquí, e es el número de edades únicas en los crímenes filtrados. |
| **Actualizar conteo de crímenes por año** | **O(m \* log(y))**: Para cada crimen, se actualiza el conteo en el árbol rojo-negro year\_count. Aquí, y es el número de años únicos en los crímenes filtrados. |
| **Obtener valores del mapa** | **O(c)** |
| **Ordenar los crímenes por cantidad, años y código** | **O(c \* log(c))**: Se utiliza [merge\_sort](vscode-file://vscode-app/c:/Users/juank/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) para ordenar los crímenes según el criterio definido. Aquí, c es el número de códigos únicos de crímenes. |
| **Obtener los N crímenes más comunes** | **O(N)**: Se utiliza [ar.sub\_list](vscode-file://vscode-app/c:/Users/juank/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) para obtener los primeros [N](vscode-file://vscode-app/c:/Users/juank/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) crímenes de la lista ordenada. |
| **Preparar la respuesta** | **O(N)**: Se formatea la información de los [N](vscode-file://vscode-app/c:/Users/juank/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) crímenes más comunes para la salida. |
| **TOTAL** | **O(log(n) + k + m \* (log(c) + log(e) + log(y)) + c \* log(c) + N)** |

## **Pruebas Realizadas**

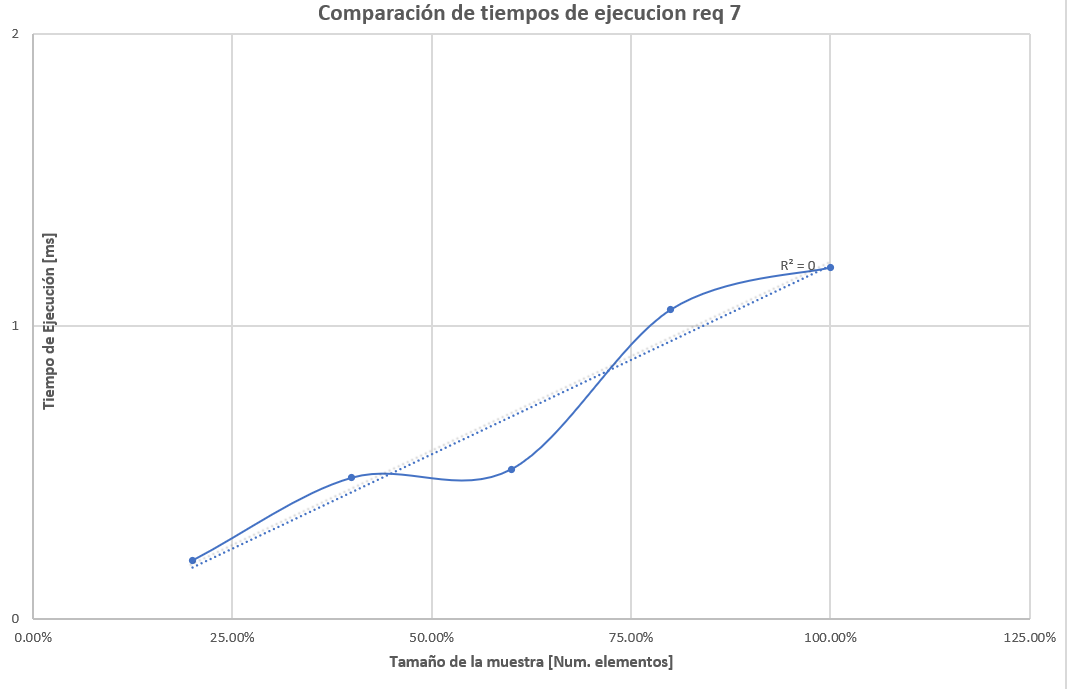
Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron N = 5, sexo = M, edad\_min = 20, edad\_max = 25

|  |  |
| --- | --- |
| Procesadores | Intel core i7 7th gen |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 10 |

### **Tablas de datos**

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| 20 pct | 198.61 |
| 40 pct | 482.53 |
| 60 pct | 511.49 |
| 80 pct | 1058.86 |
| 100 pct | 1202.9 |

### **Graficas**



## **Análisis**

El comportamiento cercano a O(N) observado en el requerimiento 7 puede explicarse por la relación lineal entre el número de crímenes procesados y el tiempo de ejecución. Esto ocurre porque las operaciones más costosas, como el filtrado de crímenes por sexo y la actualización de conteos en estructuras como mapas y árboles, dependen directamente del número de crímenes en el rango (m). Aunque teóricamente hay términos logarítmicos asociados a las búsquedas y actualizaciones en árboles y mapas, en la práctica, estos términos tienen un impacto menor debido a tamaños pequeños de subconjuntos (como códigos únicos de crímenes o años únicos). Además, la eficiencia de las estructuras de datos implementadas y la distribución uniforme de los datos contribuyen a que el tiempo de ejecución sea proporcional al tamaño de los datos procesados.