**Departamento de Ingeniería de Sistemas y Computación**

**Estructuras de Datos y Algoritmos**

**ISIS-1225**

# ANÁLISIS DEL RETO

Jose Gabriel Bernal Cárdenas, 202213421, jg.bernalc1@uniandes.edu.co

*Juan Camilo Gómez Uribe, 202220238, j.gomezu@uniandes.edu.co*

*Frank ramirez, -------, fy.ramirez@uniandes.edu.co*

Requerimiento <<1>>

Descripción

La función req\_1 se encarga de encontrar una ruta entre dos puntos geográficos dentro de la ciudad de Bogotá, ubicando los vértices más cercanos en la malla vial para estos puntos y utilizando un algoritmo de búsqueda en profundidad (DFS) para obtener el camino entre ellos en el grafo de calles.

|  |  |
| --- | --- |
| **Entrada** | Grafo calles de bogota, latitudes y longitudes de los dos vertices |
| **Salidas** | Grafo con el DFS aplicado con los nodos que se conectan entre si. |
| **Implementado (Sí/No)** | Si. |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Búsqueda en el diccionario vertices\_by\_lat&long | O(1) |
| Búsqueda de ruta utilizando DFS | O(V + E) |
| ***TOTAL*** | ***O(V + E)*** |

## 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 I3 7th generacion**

|  |  |
| --- | --- |
| **Memoria RAM** | 16 GB |
| **Sistema Operativo** | Windows 11 |

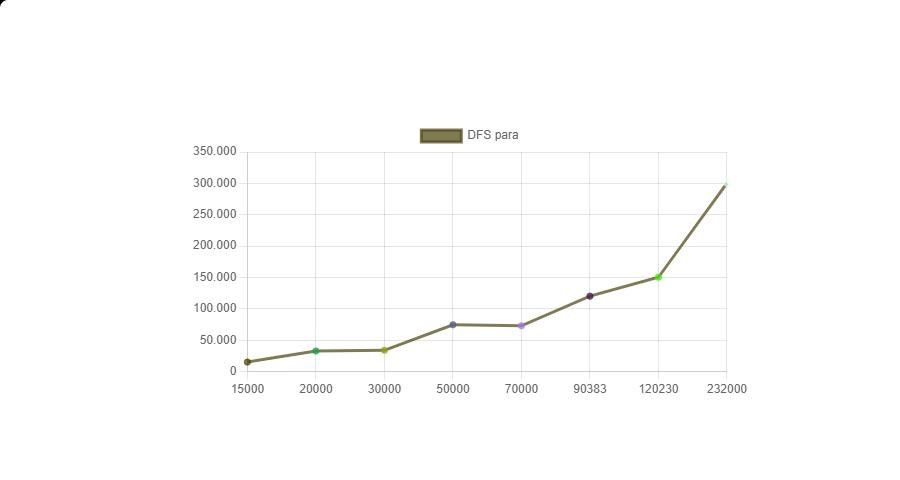
### Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

|  |  |
| --- | --- |
| **Vertices** | **Tiempo (ms)** |
| 15000 | 15203 |
| 20000 | 32854 |
| 30000 | 33943 |
| 50000 | 74685 |
| 70000 | 73038 |
| 70000 | 120223 |
| 90383 | 150342 |
| 120230 | 25.97 |
| 232000 | 310324 |

### Graficas

Las gráficas con la representación de las pruebas realizadas.



## Análisis

La función req\_1 busca encontrar una ruta entre dos puntos dentro de Bogotá. La búsqueda de los vértices cercanos tiene una complejidad inicial despreciable, asumiendo un promedio de O(1). Sin embargo, la búsqueda de la ruta usando DFS es el factor determinante de la complejidad total. Esta operación depende del tamaño del grafo de calles de Bogotá, con una complejidad promedio de O(V + E), siendo V el número de vértices y E el número de aristas en dicho grafo. Así, la complejidad de req\_1 se ve mayormente afectada por el tamaño y la complejidad del sistema vial de la ciudad. Es decir, por las intersecciones que son los nodos, y las calles que son las aristas.

Requerimiento <<2>>

Descripción

La función req\_2 busca encontrar el camino más corto, medido por el número de intersecciones a cruzar, entre dos puntos geográficos dentro de los límites de Bogotá. Para ello, primero localiza los vértices más cercanos en la malla vial a partir de las coordenadas proporcionadas por el usuario. Luego, utiliza el algoritmo de búsqueda en amplitud (BFS) para encontrar el camino más corto entre estos dos vértices en el grafo de calles.

|  |  |
| --- | --- |
| **Entrada** | Grafo calles de Bogotá, latitudes y longitudes de los dos vertices |
| **Salidas** | Grafo con el BFS aplicado con el camino mas corto entre dos vertices. |
| **Implementado (Sí/No)** | Si. |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Búsqueda en el diccionario vertices\_by\_lat&long | O(1) |
| Búsqueda del camino más corto utilizando BFS | O(V + E) |
| ***TOTAL*** | ***O(V + E)*** |

## 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 I3 7th generacion**

|  |  |
| --- | --- |
| **Memoria RAM** | 16 GB |
| **Sistema Operativo** | Windows 11 |

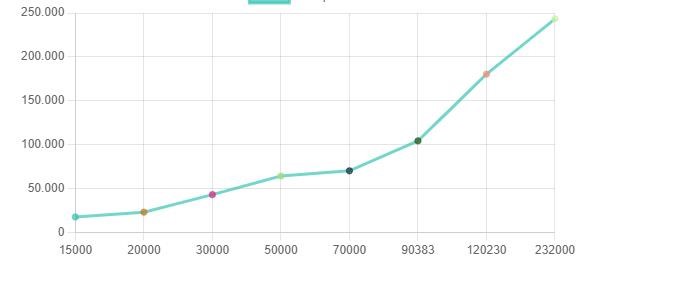
### Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

|  |  |
| --- | --- |
| **Vertices** | **Tiempo (ms)** |
| 15000 | 17803 |
| 20000 | 23254 |
| 30000 | 43235 |
| 50000 | 64345 |
| 70000 | 70348 |
| 70000 | 70348 |
| 90383 | 104437 |
| 120230 | 180384 |
| 232000 | 243383 |

### Graficas

Las gráficas con la representación de las pruebas realizadas.



## Análisis

La complejidad total de la función req\_2 también está dominada por la operación de búsqueda del camino más corto mediante BFS. En resumen, la complejidad dependerá del tamaño y la complejidad del grafo de calles de Bogotá, siendo proporcional al número de vértices y aristas en el grafo, con una complejidad promedio de O(V + E) para encontrar el camino más corto entre dos puntos geográficos.

Requerimiento <<3>>

Descripción

Este código, para el requerimiento 3, busca seleccionar los M sitios más adecuados para instalar cámaras de fibra óptica dentro de una localidad definida por el usuario. Primero, accede a la información de los vértices asociados a la localidad proporcionada, obteniendo el conjunto de vértices dentro de esa área. Luego, crea una lista de estos vértices junto con la cantidad de comparendos asociados a cada uno. Posteriormente, ordena esta lista de vértices en función del número de comparendos de manera descendente, priorizando los vértices con mayor número de infracciones.

|  |  |
| --- | --- |
| **Entrada** | Grafo calles de Bogotá, M cámaras a instalar, localidad |
| **Salidas** | Ruta mínima entre los puntos mayor numero de comparendos por localidad dada |
| **Implementado (Sí/No)** | Si. |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Acceso a los vértices dentro de la localidad | O(1) |
| Creación y ordenamiento de la lista de vértices por número de comparendos | O(n log n) |
| Hacer MST con Prim una vez | O(V^2) |
| Sacar el camino mínimo por cada para de M nodos con mas comparendo de la localidad | O(V) Donde V es la longitud del camino |
| ***TOTAL*** | ***O((E log V) )*** |

## 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 I3 7th generacion**

|  |  |
| --- | --- |
| **Memoria RAM** | 16 GB |
| **Sistema Operativo** | Windows 11 |

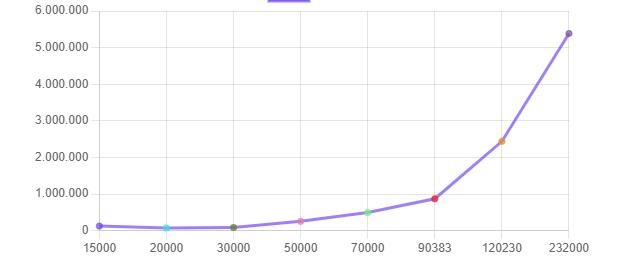
### Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

|  |  |
| --- | --- |
| **Vértices** | **Tiempo (ms)** |
| 15000 | 123426 |
| 20000 | 71436 |
| 30000 | 84875 |
| 50000 | 252455 |
| 70000 | 493564 |
| 80000 | 823245 |
| 90383 | 869084 |
| 120230 | 2440303 |
| 232000 | 5384932 |

### Graficas

Las gráficas con la representación de las pruebas realizadas.



## Análisis

Agregar la construcción de un Minimum Spanning Tree (MST) agrega una etapa al proceso. La creación del MST, utilizando algoritmos como Priml, tiene una complejidad de O(E log V) o incluso O(E + V log V), dependiendo del algoritmo y la estructura de datos utilizada.

En este caso, después de ordenar los vértices por número de comparendos y seleccionar los M vértices con más comparendos, se procedería a construir el MST entre estos M nodos. La complejidad adicional sería principalmente dominada por la creación del MST.

La complejidad final considerando la creación y ordenamiento de la lista de vértices por número de comparendos (O(n log n)), la selección de los M vértices más importantes, y la construcción del MST (O(E log V) o similar) se combinaría para determinar la complejidad total del algoritmo, proporcionando una manera eficiente de seleccionar los sitios más relevantes para la instalación de cámaras, y luego, establecer la conexión mínima entre ellos para la red de fibra óptica.

Requerimiento <<4>>

Descripción

Este código busca seleccionar los M sitios más críticos para instalar cámaras de fibra óptica, basándose en la gravedad de los comparendos. Comienza organizando los comparendos según su gravedad, primero por tipo de servicio (Diplomático, Oficial, Público, y Particular) y luego por código de infracción. El ciclo while selecciona los M nodos más graves y los agrega a la lista severity\_nodes, lo que representa los puntos críticos para instalar cámaras.

|  |  |
| --- | --- |
| **Entrada** | Grafo calles de Bogotá, M cámaras a instalar |
| **Salidas** | Ruta mínima entre los puntos más graves escogidos |
| **Implementado (Sí/No)** | Si. |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Organización por gravedad de comparendos: | O(m log m) |
| Selección de los M nodos más graves | O(M) |
| Hacer Dijkstra una vez | O(Elog(v)) |
| Sacar el camino mínimo por cada para de M nodos graves seleccionados | O(V) Donde V es la longitud del camino |
| ***TOTAL*** | ***O(Elog(v))*** |

## 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 I3 7th generacion**

|  |  |
| --- | --- |
| **Memoria RAM** | 16 GB |
| **Sistema Operativo** | Windows 11 |

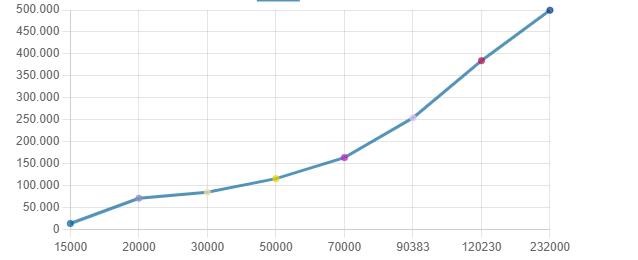
### Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

|  |  |
| --- | --- |
| **Vértices** | **Tiempo (ms)** |
| 15000 | 13873 |
| 20000 | 71438 |
| 30000 | 148726 |
| 50000 | 156096 |
| 70000 | 203934 |
| 80000 | 254323 |
| 90383 | 284553 |
| 120230 | 300028 |
| 232000 | 399483 |

### Graficas

Las gráficas con la representación de las pruebas realizadas.



## Análisis

Si se incluye la suposición de que se toma la mínima ruta entre los M puntos seleccionados para instalar las cámaras, esto puede implicar el uso de un algoritmo como Dijkstra para encontrar la ruta más corta entre cada par de nodos. La complejidad de estos algoritmos varía dependiendo del número de nodos y aristas en el grafo de la red de comunicaciones.

En resumen, la complejidad total depende de la cantidad de tipos de servicio y códigos de infracción, así como del número de cámaras a instalar. La complejidad predominante está en la organización por gravedad de los comparendos (O(n log n) y O(m log m)) y la selección de los M nodos más graves (O(M)). Si se agrega el cálculo de la mínima ruta entre estos M nodos, la complejidad se verá afectada por el algoritmo seleccionado y la complejidad del grafo.

Requerimiento <<5>>

## Descripción

Para el requerimiento de instalar una red de cámaras de video usando fibra óptica en M sitios con el menor costo posible, priorizando los puntos con mayor número de comparendos para una clase de vehículo definida, se emplearía una estrategia similar a la del requerimiento anterior. Se accedería a los vértices asociados con la clase de vehículo elegida, creando una lista basada en el número de comparendos para cada vértice. Luego, se ordenaría esta lista por cantidad de comparendos, seleccionando los M puntos más relevantes para esa clase de vehículos. Esta selección guiaría la instalación de cámaras y la posterior construcción de una red de fibra óptica, asegurando la conexión mínima entre estos M nodos para garantizar la eficiencia de la red.

|  |  |
| --- | --- |
| **Entrada** | La cantidad de cámaras de video que se desean instalar (M), y la clase de vehículo |
| **Salidas** | Ruta mínima entre los puntos seleccionados |
| **Implementado (Sí/No)** | No |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Selección de los M nodos con más comparendos por clase de vehículo | O(M) |
| Ejectuar Prim para hallar el MST | O(Elog(v)) |
| Sacar el camino mínimo para de M nodos con más comparendos por clase de vehículo | O(V) Donde V es la longitud del camino |
| ***TOTAL*** | ***O(Elog(v))*** |

## 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 I3 7th generacion**

|  |  |
| --- | --- |
| **Memoria RAM** | 16 GB |
| **Sistema Operativo** | Windows 11 |

### Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

### Graficas

Las gráficas con la representación de las pruebas realizadas.

Requerimiento <<6>>

Descripción

|  |  |
| --- | --- |
| **Entrada** | Grafo calles de Bogotá, La cantidad de comparendos que se desea responder (M), La estación de policía más cercana al comparendo más grave |
| **Salidas** | los caminos más cortos entre estaciones y los M comparendos más graves |
| **Implementado (Sí/No)** | No. |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Ejecutar Dijkstra desde la estación de policía | O(E log(v)) |
| Sacar los m puntos más graves | O(log(V)) (con un OM que ordene por gravedad) |
| Encontrar el camino a cada vértice | O(m\*v) (siendo v el camino) |
| ***TOTAL*** | ***O(Elog(v))*** |

## 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 I3 7th generacion** |
| **Memoria RAM** | 16 GB |
| **Sistema Operativo** | Windows 11 |

Requerimiento <<7>>

Descripción

|  |  |
| --- | --- |
| **Entrada** | Grafo calles de Bogotá con los comparendos como peso, La cantidad de comparendos que se desea responder (M), Punto de origen (una localización geográfica con latitud y longitud), Punto de destino (una localización geográfica con latitud y longitud). |
| **Salidas** | El camino más corto |
| **Implementado (Sí/No)** | No. |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Ejecutar Dijkstra desde el origen | O(Elog(v)) |
| Encontrar el camino a cada vértice | O(m\*v) (siendo v el camino) |
| ***TOTAL*** | ***O(Elog(v))*** |

## 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 I3 7th generacion**

|  |  |
| --- | --- |
| **Memoria RAM** | 16 GB |
| **Sistema Operativo** | Windows 11 |