Documentación por cada Requerimiento

|  |
| --- |
| **Identificador y Título**: A1. Obtener el camino de costo mínimo entre dos ubicaciones geográficas por distancia |
| **Descripción**:  Para encontrar el camino de costo mínimo se debe tomar la distancia haversiana en cada arco como medida base. El punto de origen y destino son ingresados por el usuario como latitudes y longitudes (debe validarse que dichos puntos se encuentren dentro de los límites encontrados de la ciudad). Estas ubicaciones deben aproximarse a los vértices más cercanos en la malla vial. |
| **Datos de Entrada**:   1. Latitud del punto de origen 2. Longitud del punto de origen 3. Latitud del punto de destino 4. Longitud del punto de destino |
| **Datos de Salida**:   1. Muestre el camino resultante en Google Maps 2. Sumatoria de distancias harvesianas en Km 3. Menor distancia haversiana 4. Vértices del camino 5. Cantidad de vértices del camino |
| **Estructura(s) Genérica(s) de Datos a utilizar**:   1. GrafoNoDirigido 2. RedBlackBST |
| **Parametrización de Estructura(s) Genérica(s) de Datos a utilizar**   1. La Estructura GrafoNoDirigido<K, V> se va utilizar con K = Integer y V = String.   La llave K corresponde al atributo Id de los vértices que se cargarán en él.   1. La Estructura RedBlackBST<K, V> se va utilizar con K igual a la clase Double y V igual a la clase Vertice.   La llave K corresponde al atributo latitud del vértice |
| **Complejidad Temporal:**  La complejidad temporal es O((V + A) \* log(V)) (V = Número de vértices, A = Número de arcos) porque comparado al tiempo que tomará el algoritmo de Dijkstra, el tiempo de ajustar las posiciones al nodo es irrelevante. |

|  |
| --- |
| **Identificador y Título**: A2. Determinar la red de comunicaciones que soporte la instalación de cámaras de video en los M puntos donde se presentan los comparendos de mayor gravedad. |
| **Descripción**:  El distrito quiere instalar una red de comunicaciones que le permita la instalación de cámaras de video en M sitios; sin embargo, se requiere que esta red tenga el menor costo de instalación posible. El costo de instalación de la red es de U$10000 por cada kilómetro extendido. Con la finalidad de que la red sea eficiente se seleccionaron como puntos de supervisión los M vértices donde se presentan los comparendos de mayor gravedad. Para saber si un comparendo es más grave que otro primero se mira el tipo de servicio: Público es más grave que Oficial y Oficial es más grave que Particular; si dos comparendos tienen el mismo tipo de servicio se compara el código de la infracción (campo INFRACCION) usando el orden lexicográfico (forma de comparación de los Strings en Java, A12 es más grave que A11 y B10 es más grave que A10). |
| **Datos de Entrada**:   1. Cantidad M de comparendos |
| **Datos de Salida**:   1. Tiempo que toma el algoritmo en encontrar la solución 2. Los vértices, los arcos y el costo total de implementar esta red 3. Muestre en un mapa en Google Maps la red de comunicaciones propuesta. |
| **Estructura(s) Genérica(s) de Datos a utilizar**:  GrafoNoDirigido<K, V> |
| **Parametrización de Estructura(s) Genérica(s) de Datos a utilizar**   1. La Estructura GrafoNoDirigido<K, V> se va utilizar con K = Integer y V = String.   La llave K corresponde al atributo Id de los vértices que se cargarán en él. |
| **Complejidad Temporal:**  La complejidad temporal es O((V + A) \* log(V)) (V = Número de vértices, A = Número de arcos) porque es el tiempo que le tomará al algoritmo de Dijkstra encontrar el camino con menor costo entre todos los vértices |

|  |
| --- |
| **Identificador y Título**: C1. Obtener los caminos más cortos para que los policías puedan atender los M comparendos más graves.. |
| **Descripción**:  Se espera que cada ubicación de un comparendo grave debe ser atendida por la estación de policía más cercana (mínima distancia de desplazamiento usando la malla vial). Asuma que cualquier policía en una estación puede atender un comprendo y que inicialmente todos los policías están en sus estaciones. |
| **Datos de Entrada**:   1. Cantidad M de comparendos |
| **Datos de Salida**:   1. Tiempo que toma el algoritmo en encontrar la solución 2. Los vértices, los arcos y el costo total de recorrer estas rutas para atender los comparendos 3. Muestre en un mapa en Google Maps las rutas. |
| **Estructura(s) Genérica(s) de Datos a utilizar**:  GrafoNoDirigido<K, V> |
| **Parametrización de Estructura(s) Genérica(s) de Datos a utilizar**   1. La Estructura GrafoNoDirigido<K, V> se va utilizar con K = Integer y V = String.   La llave K corresponde al atributo Id de los vértices que se cargarán en él. |
| **Complejidad Temporal:**  La complejidad temporal es O(M((V + A) \* log(V))) (V = Número de vértices, A = Número de arcos) porque es el tiempo que le tomará al algoritmo de Dijkstra encontrar el camino con menor costo entre una infracción y la que le sigue |

|  |
| --- |
| **Identificador y Título**: C2. Identificar las zonas de impacto de las estaciones de policía. |
| **Descripción**: Para poder identificar las zonas de impacto de cada una de las estaciones de policía, se debe asignar la estación de policía que puede atender cada comparendo en el menor tiempo posible; es decir, la estación que tenga la ruta más corta (mínima distancia de desplazamiento usando la malla vial) al lugar del comparendo. Tomando como base los caminos identificados anteriormente (entre cada comparendo y la estación de policía que lo atiende) cree un Grafo tomando únicamente los vértices y arcos involucrados en dichos caminos. Defina un esquema JSON para persistir su grafo. |
| **Datos de Entrada**:  Una serie de comparendos. |
| **Datos de Salida**:   1. Tiempo que toma el algoritmo en encontrar la solución 2. El número de vértices y arcos que tiene el grafo no dirigido 3. La cantidad de comparendos que atiende cada una de las estaciones de policía. 4. Para cada componente conexa imprima: el color, el ObjectId de las estaciones de policía dentro del componente, y el número de vértices incluidos |
| **Estructura(s) Genérica(s) de Datos a utilizar**:  GrafoNoDirigido<K, V>  RedBlackBST<K,V> |
| **Parametrización de Estructura(s) Genérica(s) de Datos a utilizar**   1. La Estructura GrafoNoDirigido<K, V> se va utilizar con K = Integer y V = String.   La llave K corresponde al atributo Id de los vértices que se cargarán en él.   1. La Estructura RedBlackBST<K, V> se va utilizar con K igual a la clase Double y V igual a la clase Vertice.   La llave K corresponde al atributo latitud del vértice |
| **Complejidad Temporal:**  La complejidad temporal es O(M((V + A) \* log(V))) (V = Número de vértices, A = Número de arcos, M = Número de comparendos ingresados) porque es el tiempo que le tomará al algoritmo de Dijkstra encontrar el camino con menor costo entre una estación y la infracción y se ejecutará M veces |