# **Requerimientos funcionales:**

|  |  |
| --- | --- |
| Nombre: | **Cargar Información entrega diseño** |
| Descripción: | Se deben cargar el archivo de comparendos, el de estaciones de policía y el grafo de la malla vial  **Los archivos solo deben poder cargarse una vez.** |
| Datos de Entrada: | Archivos de persistencia |
| Datos de Salida: | Se muestran el total de comparendos, el comparendo con mayor ObjectID, el total de estaciones de policía, la información de la estación de policía con el mayor ObjectID, el total de vértices de la malla vial de bogota, la información del vértice con el mayor ObjectID encontrado el total de arcos, y la información de los arcos del vértice mayor. |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | Los comparendos son añadidos en un Queue<Item> con item igual a Comparendo, las estaciones de policía son añadidas en un Queue<Item> con Item igual a EstacionesPolicia, y el archivo de persistencia del grafo es leído, y la información de esta es cargada en un GrafoNoDirigido<K, V> con K igual a Integer y V igual a String |
| Complejidad Temporal y justificación: | Como se leen archivos de persistencia uno por uno, las complejidades serán O(N) para cada estructura de datos. |

|  |  |
| --- | --- |
| Nombre: | **Requerimientos Parte Inicial 1** |
| Descripción: | Dada una localización geográfica con latitud y longitud, encontrar el Id del vértice de la malla vial más cercano por distancia haversiana. |
| Datos de Entrada: | Valores de latitud y longitud |
| Datos de Salida: | ID del vértice mas cercano a la ubicación geográfica ingresada como parámetro |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | GrafoNoDirigido<K, V> con K igual a Integer y V igual a String y.  LinearProbing<K, Vertice<K, V> > con K igual a Integer y V igual a String |
| Complejidad Temporal y justificación: | En el grafo se recorrerán todos sus vértices para poder comparar la distancia ingresada con la ubicación geográfica de cada uno, por lo tanto, la complejidad resulta O(N). |

|  |  |
| --- | --- |
| Nombre: | **Requerimientos Parte Inicial 2** |
| Descripción: | Adicionar la información de cada uno de los comparendos del año 2018 al grafo de la malla vial. Para cada comparendo se buscará el vértice más cercano y se añadirá la información del comparendo a este vértice. |
| Datos de Entrada: | Ninguno |
| Datos de Salida: | Ninguno, pero el grafo es actualizado |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | GrafoNoDirigido<K, V> con K igual a Integer y V igual a String y.  LinearProbing<K, Vertice<K, V> > con K igual a Integer y V igual a String |
| Complejidad Temporal y justificación: | Este requerimiento tiene la complejidad más alta, porque para cada comparendo se tienen que revisar todos los vértices, entonces la complejidad es O(C\*V) donde C es el numero de comparendos y V es el número de vértices del grafo. |

|  |  |
| --- | --- |
| Nombre: | **Requerimientos Parte Inicial 3** |
| Descripción: | Agregar al grafo información de costo. El grafo tendrá 2 costos en sus arcos. El primer costo es referente a la distancia harversiana entre dos vértices y el segundo es respecto al numero de comparendos entre los dos vértices |
| Datos de Entrada: | Ninguno |
| Datos de Salida: | Ninguno, pero el grafo es actualizado |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | GrafoNoDirigido<K, V> con K igual a Integer y V igual a String y.  LinearProbing<K, Vertice<K, V> > con K igual a Integer y V igual a String  Arco<K,V> con K igual a Integer y V igual a String |
| Complejidad Temporal y justificación: | Para cada vértice del grafo se tendrán que visitar sus arcos adyacentes entonces la complejidad será O(V\*E) |

|  |  |
| --- | --- |
| Nombre: | **Requerimientos Parte Inicial 4** |
| Descripción: | Adicionar la información de cada una de las estaciones de policía al grafo, nuevamente adicionando al vértice más cercano la información de la estación. |
| Datos de Entrada: | ninguno |
| Datos de Salida: | Ninguno, pero se actualiza el grafo |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | GrafoNoDirigido<K, V> con K igual a Integer y V igual a String y.  LinearProbing<K, Vertice<K, V> > con K igual a Integer y V igual a String |
| Complejidad Temporal y justificación: | Nuevamente se tendrá que visitar todos los vértices del grafo para determinar cual es el mas cercano a la estación de policía y este proceso se tendrá que repetir cuantas estaciones de policía haya, la complejidad resultará entonces O(E\*V) donde E es el número es estacione de policía y V es el numero de vértices en el grafo. |

|  |  |
| --- | --- |
| Nombre: | **Guardar grafo con toda la nueva información añadida a un archivo de persistencia** |
| Descripción: | Se adiciona toda la información del grafo actualizado a un archivo de persistencia json |
| Datos de Entrada: | ninguno |
| Datos de Salida: | ninugno |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | GrafoNoDirigido<K, V> con K igual a Integer y V igual a String y.  LinearProbing<K, Vertice<K, V> > con K igual a Integer y V igual a String  Arco<K,V> con K igual a Integer y V igual a String |
| Complejidad Temporal y justificación: | Se tienen que recorrer todos los arcos y vértices del grafo para poder consultar toda la información relevante que tendrá el archivo de persistencia de grafo. Por lo tanto la complejidad será O(V\*E) |

|  |  |
| --- | --- |
| Nombre: | **Cargar el nuevo grafo actualizado** |
| Descripción: | Carga el archivo de persistencia del nuevo grafo guardado y crea inicializa un nuevo grafo con la información del archivo |
| Datos de Entrada: | Ninguno |
| Datos de Salida: | Ninguno |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | GrafoNoDirigido<K, V> con K igual a Integer y V igual a String y.  LinearProbing<K, Vertice<K, V> > con K igual a Integer y V igual a String  Arco<K,V> con K igual a Integer y V igual a String |
| Complejidad Temporal y justificación: | Re recorre la información del archivo recorriendo los objetos json uno por uno, por lo tanto la complejidad es O(N) |

|  |  |
| --- | --- |
| Nombre: | **Requerimiento 1 parte A** |
| Descripción: | Obtener el camino de costo mínimo entre dos ubicaciones geográficas por distancia |
| Datos de Entrada: | Punto de origen y destino, ubicaciones geográficas ingresadas por el usuario |
| Datos de Salida: | En consola se muestra el camino a seguir informando:  el total de vértices, sus vértices (Id, latitud, longitud), el costo mínimo (menor distancia haversiana) y la distancia estimada (sumatoria de distancias harvesianas en Km).  En el mapa se muestra el camino resultante en Google maps |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | GrafoNoDirigido<K, V> con K igual a Integer y V igual a String y.  LinearProbing<K, Vertice<K, V> > con K igual a Integer y V igual a String  Arco<K, V> con K igual a Integer y V igual a String |
| Complejidad Temporal y justificación: | Se utilizará Dijkstra para encontrar el camino más corto hasta el destino.  O(E log V) |

|  |  |
| --- | --- |
| Nombre: | **Requerimiento 2 parte A** |
| Descripción: | Obtener el camino de costo mínimo entre dos ubicaciones geográficas por número de comparendos |
| Datos de Entrada: | Ninguno |
| Datos de Salida: | En consola se debe mostrar el tiempo que toma el algoritmo en ejecutarse y la información de la red propuesta los vértices (identificadores) y los arcos incluidos, y el costo (monetario) total.  En el mapa se muestra la red de comunicaciones propuesta |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | GrafoNoDirigido<K, V> con K igual a Integer y V igual a String y.  LinearProbing<K, Vertice<K, V> > con K igual a Integer y V igual a String  Cola de prioridad MaxHeapCP<Item> con item igual a comparendo y con criterio de comparación por gravedad requerido |
| Complejidad Temporal y justificación: | Se utilizará un algoritmo de árbol de recubrimiento mínimo más concretamente el algoritmo Prim eager O(E log V) |

|  |  |
| --- | --- |
| Nombre: | **Requerimiento 1 parte B** |
| Descripción: | Obtener el camino de costo mínimo entre dos ubicaciones geográficas por número de comparendos |
| Datos de Entrada: | ninguno |
| Datos de Salida: | En consola se muestra la información del camino a seguir, en mapa se muestra el camino resultante |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | GrafoNoDirigido<K, V> con K igual a Integer y V igual a String y.  LinearProbing<K, Vertice<K, V> > con K igual a Integer y V igual a String  Arco<K, V> con K igual a Integer y V igual a String |
| Complejidad Temporal y justificación: | Se utilizará Dijkstra para encontrar el camino más corto hasta el destino.  O(E log V) |

|  |  |
| --- | --- |
| Nombre: | **Requerimiento 2 parte B** |
| Descripción: | 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. |
| Datos de Entrada: | El numero M de comparendos que se requieren |
| Datos de Salida: | En consola se muestra el tiempo que toma el algoritmo y a la información de la red propuesta. En el mapa visualiza la red de comunicaciones propuesta. |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | GrafoNoDirigido<K, V> con K igual a Integer y V igual a String y.  LinearProbing<K, Vertice<K, V> > con K igual a Integer y V igual a String  MaxHeapCP<Item> donde item es un comparendo |
| Complejidad Temporal y justificación: | Se utilizará un algoritmo de árbol de recubrimiento mínimo más concretamente el algoritmo Prim eager O(E log V) |

|  |  |
| --- | --- |
| Nombre: | **Requerimiento 1 parte C** |
| Descripción: | Obtener los caminos más cortos para que los policías puedan atender los M comparendos más graves. |
| Datos de Entrada: | El número M de comparendos que se requieren |
| Datos de Salida: | En consola se muestra el tiempo que toma el algoritmo y los caminos resultantes para cada uno de los m comparendos.  En el mapa se encuentran los caminos resultantes |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | GrafoNoDirigido<K, V> con K igual a Integer y V igual a String y.  LinearProbing<K, Vertice<K, V> > con K igual a Integer y V igual a String  MaxHeapCP<Item> donde item es un comparendo |
| Complejidad Temporal y justificación: | Se utilizará el algoritmo de Dijkstra con cada comparendo hacia las estaciones de policía y se elegirá el camino mas corto entre los caminos cortos O(E log V) |

|  |  |
| --- | --- |
| Nombre: | **Requerimiento 2 parte C** |
| Descripción: | Identificar las zonas de impacto de las estaciones de policía |
| Datos de Entrada: | Ninguno |
| Datos de Salida: | Se muestra en consola el número de vértices y arcos del grafo nuevo creado, el tiempo que tardo el algoritmo en encontrar la solución, la cantidad de comparendos que atiende cada estación de policía y para cada componente mostrar el color y numero de vértices incluidos.  En el mapa se muestra el nuevo grafo con los vértices y arcos pintados. |
| Parametrización de Estructura(s) Genérica(s) de Datos a utilizar: | GrafoNoDirigido<K, V> con K igual a Integer y V igual a String y.  LinearProbing<K, Vertice<K, V> > con K igual a Integer y V igual a String  Arco<K, V> con K igual a Integer y V igual a String |
| Complejidad Temporal y justificación: | Se hará un árbol de recubrimiento mínimo respecto a cada estación de policía, y se compararan las distancias de cada vértice con Dijkstra si varias estaciones coinciden con un mismo vértice en su mst, la que tenga un menor costo en distancia será la que lo agregue en su zona de impacto. O(E log V) |