Requerimientos Funcionales Proyecto 3

* El requerimiento 1 solo explica los tres tipos de costos que existen en el proyecto para los arcos, por lo que no es necesario realizar la tabla.
* En la mayoría de los requerimientos cuando sea necesario retornar una lista de objetos, se utilizaría la estructura de datos Queue para hacer el retorno.

|  |  |
| --- | --- |
| **Nombre** | R0 – Carga del grafo no dirigido |
| **Resumen** | Cargar el grafo no dirigido de la malla vial de la ciudad completa de Bogotá creado en el taller 7. Informar el total de vértices y de arcos que definen el grafo cargado. Solo es permitido leer una vez la información de los archivos. |
| **Entradas** | |
| Archivo de datos. | |
| **Resultados** | |
| Se informa al usuario el total de vértices y arcos que definen el grafo. | |
| **Complejidad temporal** | |
| O(N) | |
| **Estructura de Datos** | Grafo no dirigido |
| **Justificación** | El proyecto se basa en el uso de un grafo no dirigido para guardar los datos que se deben utilizar. |

|  |  |
| --- | --- |
| **Nombre** | R2 – Completar grafo con los tres costos y crear archivo JSON. |
| **Resumen** | Completar el grafo con los tres costos para cada arco a partir de los datos de Uber y la malla vial de Bogotá. Crear un archivo JSON para guardarlo y un método que permita cargar el nuevo grafo JSON generado. |
| **Entradas** | |
| Archivo de datos. | |
| **Resultados** | |
| Ninguno. | |
| **Complejidad temporal** | |
| O(N) | |
| **Estructura de Datos** | Grafo no dirigido |
| **Justificación** | En el mismo grafo creado en el requerimiento 0 se deben agregar los diferentes tipos de costos. |

|  |  |
| --- | --- |
| **Nombre** | R3 – ID del vértice de la malla vial más cercano por distancia Harvesine. |
| **Resumen** | Dada una localización geográfica con latitud y longitud, encontrar el ID del vértice de la malla vial más cercano por distancia Harvesine. |
| **Entradas** | |
| Longitud | |
| Latitud | |
| **Resultados** | |
| Se debe retornar el ID del vértice de la malla vial más cercano a la localización geográfica por distancia Harvesine. | |
| **Complejidad horaria** | |
| O(E) | |
| **Estructura de Datos** | Cola de prioridad orientada a menor. |
| **Justificación** | Para la localización dada, que representa un vértice, se pueden poner los arcos con su costo harversine en la cola, y al hacer dequeue, se retornaría a la primera vez el ID más cercano. |

|  |  |
| --- | --- |
| **Nombre** | R4 – Camino de costo mínimo (menor tiempo promedio según Uber en la ruta) para un viaje entre dos localizaciones geográficas de la ciudad. |
| **Resumen** | Encontrar el camino de costo mínimo (menor tiempo promedio según Uber en la ruta) para un viaje entre dos localizaciones geográficas de la ciudad. Estas localizaciones deben aproximarse a las localizaciones más próximas en la malla vial. |
| **Entradas** | |
| Longitud (origen) | |
| Latitud (origen) | |
| Longitud (destino) | |
| Latitud (destino) | |
| **Resultados** | |
| Se retorna el camino de costo mínimo (menor tiempo promedio según Uber en la ruta) para un viaje entre dos localizaciones geográficas de la ciudad. Adicionalmente, se informa el total de vértices (con sus ID’s, latitudes y longitudes), el costo mínimo (menor tiempo promedio en segundos) y la distancia estimada (sumatoria de distancias Harvesine en km).  Se muestra el camino resultante en Google Maps (incluyendo la ubicación inicial y de destino). | |
| **Complejidad horaria** | |
| O(E\*Log(V)) | |
| **Estructura de Datos** | Un arreglo de arcos y un arreglo de double para el costo. Adicionalmente, una cola de prioridad orientada a menor. |
| **Justificación** | Este requerimiento se puede resolver utilizando el algoritmo Dijkstra que busca los caminos más cortos, el cual utiliza las estructuras mencionadas anteriormente para funcionar eficientemente. |

|  |  |
| --- | --- |
| **Nombre** | R5 – N vértices con menor velocidad promedio en la ciudad de Bogotá. |
| **Resumen** | Determinar los n vértices con menor velocidad promedio en la ciudad, siendo la velocidad promedio de un vértice el promedio de las velocidades de todos sus arcos. |
| **Entradas** | |
| N | |
| **Resultados** | |
| Se muestran los n vértices con su ID, latitud y longitud, ordenados de menor a mayor por la velocidad promedio del vértice. Se informan el número de componentes conexos que se definen en estos vértices (por cada componente informar los ID’s de los vértices que los componen).  Se marca la localización de los n vértices resultantes en un mapa en Google Maps usando un color 1 y un color 2 que destaque la componente conexa más grande (para esta componente muestre sus vértices y sus arcos). | |
| **Complejidad horaria** | |
| O(V\*E) | |
| **Estructura de Datos** | Cola de prioridad orientada a menor. |
| **Justificación** | Al realizar un dequeue en una cola de prioridad, el resultado que se retorna es el dato menor o mayor, teniendo en cuenta la forma en la que se decide orientar la cola. |

|  |  |
| --- | --- |
| **Nombre** | R6 – MST con criterio distancia aplicado al componente conexo más grande de la malla vial de Bogotá. |
| **Resumen** | Calcular un árbol de expansión mínima (MST) con criterio distancia, utilizando el algoritmo de Prim aplicado al componente conexo más grande la malla vial de Bogotá. |
| **Entradas** | |
| Ninguna. | |
| **Resultados** | |
| Se muestra el tiempo que toma el algortimo en encontrar la solución (en ms), además del total de vértices en el componente, con sus ID’s, los arcos incluidos (ID vértice inicial e ID vértice final) y el costo total (en km) del árbol.  Se muestra el árbol resultante en Google Maps, sus vértices y sus arcos. | |
| **Complejidad horaria** | |
| O(E\*Log(V)) | |
| **Estructura de Datos** | Un arreglo de arcos, un arreglo de double para el costo y uno de boolean para diferenciar los arcos marcados. Adicionalmente, una cola de prioridad orientada a menor. |
| **Justificación** | Este requerimiento se puede resolver utilizando el algoritmo de Prim, el cual utiliza estas estructuras para funcionar eficientemente. |

|  |  |
| --- | --- |
| **Nombre** | R7 – Camino de menor costo (menor distancia Haversine) para un viaje entre dos localizaciones geográficas de la ciudad. |
| **Resumen** | Encontrar el camino de menor costo (menor distancia Haversine) para un viaje entre dos localizaciones geográficas de la ciudad. Estas localizaciones deben aproximarse a las localizaciones más próximas en la malla vial. |
| **Entradas** | |
| Longitud (origen) | |
| Latitud (origen) | |
| Longitud (destino) | |
| Latitud (destino) | |
| **Resultados** | |
| Se muestra el camino a seguir, informando el total de vértices con sus ID’s, latitudes y longitudes, el tiempo estimado (la sumatoria de los tiempos de sus arcos) y la distancia Haversine estimada (sumatoria de distancias Haversine en km).  Se muestra el camino resultante en Google Maps, con su ubicación de inicio y de destino. | |
| **Complejidad horaria** | |
| O(E\*Log(V)) | |
| **Estructura de Datos** | Un arreglo de arcos y un arreglo de double para el costo. Adicionalmente, una cola de prioridad orientada a menor. |
| **Justificación** | Este requerimiento se puede resolver utilizando el algoritmo Dijkstra que busca los caminos más cortos, el cual utiliza las estructuras mencionadas anteriormente para funcionar eficientemente. |

|  |  |
| --- | --- |
| **Nombre** | R8 – Que vértices son alcanzables para un tiempo T. |
| **Resumen** | A partir de las coordenadas de una localización geográfica de la ciudad de origen, indique cuáles vértices son alcanzables para un tiempo T (en s). La localización de origen debe aproximarse a la localización más próxima en la malla vial. |
| **Entradas** | |
| Longitud (origen) | |
| Latitud (origen) | |
| T (en segundos) | |
| **Resultados** | |
| Se muestran los identificadores y la ubicación de los vértices alcanzables en un tiempo T a partir de la localización de origen.  Se marca la localización de origen en un color 1 y las localización de los vértices alcanzables en un color 2 en Google Maps. | |
| **Complejidad horaria** | |
| O(E) | |
| **Estructura de Datos** | Cola de prioridad orientada a menor. |
| **Justificación** | A medida que se le hace dequeue a la cola, se va comparando el costo en tiempo a cada arco para asegurarse que se retornen solo los vértices alcanzables en el tiempo T. |

|  |  |
| --- | --- |
| **Nombre** | R9 – MST con criterio distancia aplicado al componente conexo más grande la malla vial de Bogotá. |
| **Resumen** | Calcular un árbol de expansión mínima (MST) con criterio distancia, utilizando el algoritmo de Kruskal aplicado al componente conexo más grande la malla vial de Bogotá. |
| **Entradas** | |
| Ninguna | |
| **Resultados** | |
| Se muestra el tiempo que toma el algortimo en encontrar la solución (en ms), además del total de vértices en el componente, con sus ID’s, los arcos incluidos (ID vértice inicial e ID vértice final) y el costo total (en km) del árbol.  Se muestra el árbol resultante en Google Maps, sus vértices y sus arcos. | |
| **Complejidad horaria** | |
| O(E\*Log(V)) | |
| **Estructura de Datos** | Un arreglo de arcos, un arreglo de double para el costo y uno de boolean para diferenciar los arcos marcados. Adicionalmente, una cola de prioridad orientada a menor. |
| **Justificación** | Este requerimiento se puede resolver utilizando alguno de los algoritmos de un MST, los cuales suelen tener estas estructuras para funcionar eficientemente. |

|  |  |
| --- | --- |
| **Nombre** | R10 – Nuevo grafo simplificado no dirigido de las zonas Uber. |
| **Resumen** | Construir un nuevo grafo simplificado no digirido de las zonas Uber, donda cada zona es representada con un único vértice y los enlaces entre ellas representan su vecindad dentro de la malla vial (No se consideran arcos que definan auto-ciclos entre una misma zona). Para cada zona seleccione una única localización referencia de manera que pueda luego visualizarse en el mapa. Adicionalmente, el costo del arco entre dos zonas vecinas corresponde al tiempo de viaje promedio reportado por Uber entre dichas zonas, teniendo en cuenta los tiempos promedio para los días en que Uber haya reportado viajes entre dichas zonas. En caso de NO existir ningún tiempo de viajes en los datos Uber entre las dos zonas, asumir un valor predefinido de 200 segundos. Entre dos vértices adyacentes solo debe crearse un arco, sin importar que pueda haber muchas intersecciones en común entre dichas zonas. |
| **Entradas** | |
| Ninguna | |
| **Resultados** | |
| Se reporta la cantidad de vértices y arcos.  Se muestra el grafo resultante en Google Maps con sus vértices (usando su localización de referencia) y sus arcos. Los arcos entre zonas conectan sus localizaciones de referencia. | |
| **Complejidad horaria** | |
| O(N) | |
| **Estructura de Datos** | Grafo no dirigido |
| **Justificación** | Se debe crear un nuevo grafo no dirigido. |

|  |  |
| --- | --- |
| **Nombre** | R11 – Camino de costo mínimo basado en el tiempo promedio entre una zona de origen y una zona de destino sobre el grafo de zonas. |
| **Resumen** | Calcular el camino de costo mínimo (algoritmo de Dijkstra) basado en el tiempo promedio entre una zona de origen y una zona de destino sobre el grafo de zonas. |
| **Entradas** | |
| Ninguna | |
| **Resultados** | |
| Se muestra el tiempo que toma el algoritmo en encontrar la solución (en ms) y el camino resultante con su secuencia de vértices/zonas y su costo total (sumatoria de tiempos de los arcos en s). Adicionalmente, se muestra el tiempo promedio desde la zona origen hasta la zona destino reportado por el archivo de resolución semanal de Uber.  Se muestra el camino de costo mínimo en Google Maps con sus vértices (usando su localización de referencia) y sus arcos. Los arcos entre zonas conectan sus localizaciones de referencia. | |
| **Complejidad horaria** | |
| O(E\*Log(V)) | |
| **Estructura de Datos** | Un arreglo de arcos y un arreglo de double para el costo. Adicionalmente, una cola de prioridad orientada a menor. |
| **Justificación** | Este requerimiento se puede resolver utilizando el algoritmo Dijkstra que busca los caminos más cortos, el cual utiliza las estructuras mencionadas anteriormente para funcionar eficientemente. |

|  |  |
| --- | --- |
| **Nombre** | R12 – A partir de una zona de origen, dar sus caminos de menor longitud a todas sus zonas alcanzables. |
| **Resumen** | A partir de una zona origen, calcular los caminos de menor longitud (cantidad de arcos) a todas sus zonas alcanzables. De estos caminos, seleccionar el camino más largo. |
| **Entradas** | |
| Ninguna | |
| **Resultados** | |
| Se muestra el tiempo que toma el algoritmo en encontrar la solución (en ms) y el camino resultante más largo con su secuencia de vértices/zonas y su número total de arcos. Si hay múltiples caminos con la mayor longitud, mostrar entre estos aquel que llegue al vértice destino con menor MOVEMENT\_ID.  Se muestra el camino más distante desde la zona de origen en Google Maps con sus vértices (usando su localización de referencia) y sus arcos. Los arcos entre zonas conectan sus localizaciones de referencia. | |
| **Complejidad horaria** | |
| O(E\*Log(V)) | |
| **Estructura de Datos** | Un arreglo de arcos y un arreglo de double para el costo. Adicionalmente, una cola de prioridad orientada a menor. |
| **Justificación** | Este requerimiento se puede resolver utilizando el algoritmo Dijkstra que busca los caminos más cortos, el cual utiliza las estructuras mencionadas anteriormente para funcionar eficientemente. |