|  |  |
| --- | --- |
| **Nombre** | **A1** |
| **Nombre** | **0** |
| Resumen | - Cargar el Grafo No Dirigido (grafo más grande) de la malla vial de la ciudad completa de Bogotá creado en el taller 7. |
| Entradas |  |
| Resultado | Se mostrará en consola: Total de vértices, Total de arcos. |
| Complejidad |  |
| Estructura | Grafo No Dirigido, |
| **Nombre** | 1 |
| Resumen | Al grafo creado se agregará la información de costo |
| Entradas |  |
| Resultado | Los arcos ahora tienen costos |
| Complejidad | O(E) |
| Estructura | Grafo No Dirigido, en la que vamos a modificar los arcos y obtener todos los datos |
| **Nombre** | 2 |
| Resumen | Buscar los tiempos promedio de viaje que están en un rango y que son del primer trimestre del 2018. |
| Entradas | Limite bajo, Limite alto, Numero de viajes |
| Resultado | Se debe mostrar únicamente N viajes ordenados por zona de origen y zona de destino. Por cada viaje se debe mostrar su zona de origen, zona de destino, mes y tiempo promedio mensual del viaje. |
| Complejidad | O(n) peor caso |
| Estructura | Cola de prioridad, solo se requiere los n viajes con una propiedad y que estén ordenados por algo fijo, asique la cola de prioridad es lo mejor ya que si se organiza por zona de origen y destino, solo se debe buscar los n viajes que cumplan la propiedad. |
| **Nombre** | 3 |
| Resumen | Dada una localización geográfica con latitud y longitud, encontrar el Id del Vértice de la malla vial más cercano por distancia Haversine. |
| Entradas | latitud y longitud |
| Resultado | Se dará el ID del vértice mas cercano |
| Complejidad | O(1) caso general, en el peor caso muy particular O(V) |
| Estructura | Tabla de hash, en la que se almaceno los vértices, y además es optima ya que buscara en los vértices colisionados que son los mas cercanos |
| **Nombre** | 4 |
| 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. |
| Entradas | Longitud y latitud de origen, Longitud y latitud de destino |
| Resultado | Respuesta en consola: se imprimira en la consola de texto el camino a seguir, informando el total de vértices, sus vértices (Id, latitud, longitud), el costo mínimo (menor tiempo promedio en segundos) y la distancia estimada (sumatoria de distancias Haversine en Km).  Visualización mapa: Mostrara el camino resultante en Google Maps (incluyendo la ubicación de inicio y la ubicación de destino). |
| Complejidad | E\*log(V), ya que usualmente es O(1) el punto 3 |
| Estructuras | Se usará el algoritmo Dijkstra, dos arreglos y una lista de prioridad |

|  |  |
| --- | --- |
| **Nombre** | **B3** |
| Resumen | dasdasdasdasdasda |
| **Nombre** | 5 |
| Resumen | Determinar los n vértices con menor velocidad promedio en la ciudad de Bogotá. |
| Entradas | Numero de vértices a mirar. |
| Resultado | Respuesta en consola: Mostrar los n vértices resultantes en la consola de texto (su identificador, su ubicación (latitud, longitud), ordenados de menor a mayor por la velocidad promedio del vértice. Informar el número de componentes conectados (subgrafos) que se definen entre estos vértices en el grafo original. Por cada componente informar los identificadores de los vértices que la componen.  Visualización mapa: marque la localización de los n vértices resultantes en un mapa en Google Maps usando un color 1. Destaque la componente conectada más grande (con más vértices) usando un color 2. Para esta componente muestre sus vértices y sus arcos. |
| Complejidad | O(V\*E) |
| Estructura | Grafo No Dirigido, para saber los datos |
| **Nombre** | 6 |
| Resumen | Calcular un árbol de expansión mínima (MST) con criterio distancia, utilizando el algoritmo de Prim, aplicado al componente conectado (subgrafo) más grande de la malla vial de Bogotá. |
| Entradas |  |
| Resultado | Respuesta en consola: imprimira en la consola de texto el tiempo que toma el algoritmo en encontrar la solución (en milisegundos), y la siguiente información del árbol generado: el total de vértices en el componente, los vértices (identificadores), los arcos incluidos (Id vértice inicial e Id vértice final) y el costo total (distancia en Km) del árbol.  Visualización mapa: Se mostrara el árbol generado resultante en Google Maps: sus vértices y sus arcos. |
| Complejidad | O((E+V)+ÉLog(É)) (COMPLEJIDAD DE componentes conexos y lazy prim ) siendo É los arcos del subgrafo |
| Estructura | Como se organiza CC una lista de prioridad, y por lazy prim una Queue, un arreglo y una cola de prioridad. |
| **Nombre** | 7 |
| Resumen | Encontrar el camino de menor costo (menor distancia Haversine) para un viaje entre dos localizaciones geográficas de la ciudad |
| Entradas | Longitud y latitud de origen, Longitud y latitud de destino |
| Resultado | Respuesta en consola: imprimir en la consola de texto el camino a seguir, informando el total de vértices, sus vértices (Id, latitud, longitud), el tiempo estimado (la sumatoria de los tiempos de sus arcos) y la distancia Haversine estimada (sumatoria de distancias Haversine en Km).  Visualización mapa: Mostrar el camino resultante en Google Maps (incluyendo la ubicación de inicio y la ubicación de destino). |
| Complejidad | E\*log(V) |
| Estructura | Se usará el algoritmo Dijkstra, dos arreglos y una lista de prioridad |
| **Nombre** | 8 |
| 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 segundos) dado por el usuario |
| Entradas | Longitud y latitud de origen |
| Resultado | Respuesta en consola: Muestre en la consola los identificadores y la ubicación (lat, lon) de los vértices alcanzables en un tiempo T a partir de la localización de origen.  Visualización mapa: Marque la localización de origen en un color 1 y las localizaciones de los vértices alcanzables en un color 2 en Google Maps ). |
| Complejidad | O(V+E) peor caso |
| Estructura | Dos Queue para saber en que vértices y arcos he estado |
| **Nombre** | 9 |
| Resumen | Calcular un árbol de expansión mínima (MST) con criterio distancia, utilizando el algoritmo de Kruskal, aplicado al componente conectado (subgrafo) más grande de la malla vial de Bogotá. |
| Entradas |  |
| Resultado | Respuesta en consola: Muestre en la consola de texto el tiempo que toma el algoritmo en encontrar la solución (en milisegundos), y la siguiente información del árbol generado: el total de vértices en la componente, los vértices (identificadores), los arcos incluidos (Id vértice inicial e Id vértice final) y el costo total (distancia en Km) del árbol.  Visualización mapa: Muestre el árbol generado resultante en Google Maps: sus vértices y sus arcos. |
| Complejidad | O(E+V +ÉLog(’V)) siendo É y’V los arcos y vértices respectivamente del subgrafo |
| Estructura | Como se organiza CC una lista de prioridad, y por el uso de Kruskal una Queue y una cola de prioridad |
| **Nombre** | 10 |
| Resumen | Construir un nuevo grafo simplificado No dirigido de las zonas Uber, donde cada zona (MOVEMENT\_ID) es representada con un único vértice y los enlaces entre ellas representan su vecindad dentro de la malla vial. |
| Entradas |  |
| Resultado | Respuesta en consola: Al final de la construcción del grafo de zonas, reportar la cantidad de vértices y arcos (cada arco debe contarse una única vez).  Visualización mapa: Muestre el grafo resultante en Google Maps: sus vértices (usando su localización de referencia) y sus arcos. Los arcos entre zonas conectan sus localizaciones de referencia. |
| Complejidad | O(VlogE) |
| Estructura | Grafo no dirigido para retornar, y 3 colas para las zonas ya completadas, para los arcos de cada zona y para los vértices de cada zona para agregar los arcos |
| **Nombre** | 11 |
| 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 | Longitud y latitud de origen, Longitud y latitud de destino |
| Resultado | Respuesta en consola: Muestre en la consola de texto el tiempo que toma el algoritmo en encontrar la solución (en milisegundos) y del camino resultante: su secuencia de vértices/zonas y su costo total . Adicionalmente, mostrar el tiempo promedio desde la zona origen hasta la zona destino reportado por el archivo de resolución semanal de Uber. Esto con el propósito que el usuario pueda comparar el tiempo estimado en el grafo de zonas y el tiempo estimado con los datos de Uber entre la zona origen y la zona destino.  Visualización mapa: Muestre el camino de costo mínimo en Google Maps: sus vértices (usando su localización de referencia) y sus arcos. Los arcos entre zonas conectan sus localizaciones de referencia. |
| Complejidad | E\*log(V) |
| Estructura | Dos arreglos y una lista de prioridad por usar dijktra |
| **Nombre** | 12 |
| 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 (mayor cantidad de arcos); este será el camino desde la zona origen a su zona más distante (teniendo en cuenta el número de arcos) en el grafo de zonas. |
| Entradas | Una zona de origen |
| Resultado | Respuesta en consola: Muestre en la consola de texto el tiempo que toma el algoritmo en encontrar la solución (en milisegundos) y del camino resultante más largo: su secuencia de vértices/zonas (MOVEMENT\_ID) y su número total arcos. Si hay múltiples caminos con la mayor longitud (número de arcos), mostrar entre estos caminos aquel que llegue al vértice destino con menor MOVEMENT\_ID.  Visualización mapa: Muestre el camino más distante desde la zona de origen en Google Maps: sus vértices (usando su localización de referencia) y sus arcos. Los arcos entre zonas conectan sus localizaciones de referencia. |
| Complejidad | No es estimable de manera correcta ya que es greedy |
| Estructura | Dos Queue uno para los vértices y el otro para los arcos |