|  |  |
| --- | --- |
| Proyecto  Cupi2 | ISIS-1206 Estructuras de datos  **Requerimientos funcionales** |
| Ejercicio | Proyecto 3 |
| Autor: | Andrés Delgado y Juan Villamarín |
| Semestre: | 2019-2 |

|  |  |
| --- | --- |
| **Nombre** | R4A. -Encontrar el camino de costo mínimo (menor tiempo promedio según Uber en la ruta) para un viaje entre dos localizaciones |
| **Resumen** | Muestra 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). |
| **Entradas** | |
| Latitud y longitud ubicación origen | |
| Latitud y longitud ubicación origen | |
| **Resultados** | |
| Se muestra el camino de arcos con su información respectiva en la consola y en el mapa visualizados | |
| **Complejidad temporal** | |
| **Se usará el algoritmo de caminos más cortos DJIKSTRA con complejidad temporal de Elog(v) E(cantidad de arcos) y V(cantidad de vértices)** | |

|  |  |
| --- | --- |
| **Nombre** | R5A. Determinar los n vértices con menor velocidad promedio en la ciudad de Bogotá. |
| **Resumen** | Determinar los n vértices con menor velocidad promedio en la ciudad de Bogotá. Siendo la velocidad promedio de un vértice v, el promedio de las velocidades de todos sus arcos. El parámetro n es un dato de entrada dado por el usuario. |
| **Entradas** | |
| N vértices sobres los que se desea conocer su velocidad | |
| **Resultados** | |
| 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. | |
| **Complejidad temporal** | |
| **Primeramente se sacaran los vértices de la hash table y se ordenaran con un merge sort O(Nlog(N)) según su velocidad promedio. Se sacan los n vértices solicitados por el usuario y usar el algoritmo KUSARAJU-SHARIR para encontrar las componentes conectadas entre estos vértices complejidad del algoritmo (2(V+E))** | |

|  |  |
| --- | --- |
| **Nombre** | R6A. - 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. |
| **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** | |
| Ninguna | |
| **Resultados** | |
| Se imprimen en consola los vértices y arcos que hacen parte del MST,se visualiza en el mapa | |
| **Complejidad temporal** | |
| **Algoritmo a usar Prim; complejidad (E log V)** | |

|  |  |
| --- | --- |
| **Nombre** | R7B. Encontrar el camino de menor costo (menor distancia Haversine) para un viaje entre dos localizaciones geográficas de la ciudad |
| **Resumen** | Muestre 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 |
| **Entradas** | |
| Latitud y longitud de la ubicación origen | |
| Latitud y longitud de la ubicación destino | |
| **Resultados** | |
| Muestra en la consola de texto el camino a seguir, informando el total de vértices, sus vértices la distancia y el tiempo.Se visualiza el camino en el mapa | |
| **Complejidad temporal** | |
| **Se usará el algoritmo de caminos más cortos DJIKSTRA con complejidad temporal de Elog(v) E(cantidad de arcos) y V(cantidad de vértices)** | |

|  |  |
| --- | --- |
| **Nombre** | R8B. A partir de las coordenadas de una localización geográfica de la ciudad (lat, lon) de origen, indique cuáles vértices son alcanzables para un tiempo T (en segundos) |
| **Resumen** | A partir de las coordenadas de una localización geográfica de la ciudad (lat, lon) de origen, indique cuáles vértices son alcanzables para un tiempo T (en segundos) dado por el usuario. La localización de origen debe aproximarse a la localización más próxima en la malla vial. |
| **Entradas** | |
| Longitud y latitud | |
| T tiempo en segundos | |
| **Resultados** | |
| 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. | |
| **Complejidad temporal** | |
| **N,siendo N la cantidad de ertices adyacentes al dado por parámetro.Se verifica que sea alcanzable con el tiempo dado por parametro** | |

|  |  |
| --- | --- |
| **Nombre** | R9B. Calcular un árbol de expansión mínima (MST) con criterio distancia, utilizando el algoritmo de Kruskal . |
| **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** | |
| Ninguna | |
| **Resultados** | |
| Muestra en la consola de texto el tiempo que toma el algoritmo en encontrar la solución (en milisegundos),arcos vértices,costo y su visualización en el mapa | |
| **Complejidad temporal** | |
| **Algoritmo a utilizar KRUSKAL; complejidad ElogE** | |

|  |  |
| --- | --- |
| **Nombre** | R10C.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. |
| **Resumen** |  |
| **Entradas** | |
| zona de salida | |
| **Resultados** | |
| Al final de la construcción del grafo de zonas, reportar la cantidad de vértices y arcos (cada arco debe contarse una única vez). | |
| **Complejidad temporal** | |
| **Complejidad O(2) al estar en una hash table se puede llegar a los datos con facilidad ya que la llave se define por zona origen y trimestre por lo que al cargar los datos de un semestre se debe acceder primero al primer trimestre del semestre y luego al segundo trimestre del semestre** | |

|  |  |
| --- | --- |
| **Nombre** | R11C. 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. |
| **Resumen** | Dadas una zona origen y una zona destino se debe encontrar el camino de menor costos se debe mostrar les vertices de este camino y su costo |
| **Entradas** | |
| Zona origen dada | |
| Zona destino dada | |
| **Resultados** | |
| Se debe mostrar el tiempo que toma el algoritmo en encontrar la solución (en milisegundos) y del camino resultante | |
| **Complejidad temporal** | |
| **Complejidad E log V ya que se usará una implementación de dijkstra sobre una estructura binary heap y esta es la complejidad total** | |

|  |  |
| --- | --- |
| **Nombre** | R12C. – 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) |
| **Resumen** | A partir de la zona origen dada por parametro se busca el camino más largo dado por la cantidad de arcos |
| **Entradas** | |
| Zona origen | |
| **Resultados** | |
| Se debe mostrar 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 | |
| **Complejidad temporal** | |
| **Complejidad E log V ya que se usará una implementación de dijkstra sobre una estructura binary heap y esta es la complejidad total** | |