ANÁLISIS DEL RETO 4

Ana Fadul Ramos, 202320756, a.fadul@uniandes.edu.co

Manuela Galarza, 202320796, m.galarza@uniandes.edu.co

Nicolás Parra, 202322257, n.parraz@uniandes.edu.co

Carta de Datos

Descripción

Este requerimiento se hacer la carga de datos para poder utilizarlos en los siguientes requerimientos.

Entrada	Los cv's de vuelos y aeropuertos
Salidas	Estructura de Datos
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Creación de indices para las estructuras	O(1)
Agregar vertices y arctos	O(1)
Organizar aeropuertos por el más concurrido con MergeSort	O(n*logn)
Impresión de la información	O(1)
TOTAL	O(n*logn)

Análisis

Considerando que agregar vértices y arcos a un grafo es igual a O(1), podemos igualar esta complejidad a casi toda la carga de datos, con la excepción de la creación de listas basadas en el orden de concurrencia de cada aeropuertos, al hacerle merge sort a estas se obtiene una complejidad de O(n*logn) en la cual n es el número de vértices, y por ende es también comúnmente representado por la variable v.

Requerimiento 1

Descripción

Este requerimiento se encarga de encarga de identificar si hay una ruta entre dos destinos turísticos.

Entrada	Punto de origen (localización con latitud y longitud), punto de destino (localización con latitud y longitud)
Salidas	Tiempo de ejecución, Distancia total del camino entre el punto de origen y el de destino, El número de aeropuertos que se visitan en el camino encontrado, Información acerca del camino encontrado.
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Encontrar los aeropuertos más cercanos a las coordenadas input usando el método de Harvesine	O(A) A: número total de aeropuertos
Verificar que las distancias sean menores o iguales a 30, si no se cumple la función se acaba.	O(1)
Utilizar DFS para encontrar si existe un camino entre el nodo de origen y el nodo destino.	O(v + e) e: edges, v: vértices en vuelos_comerciales_min
Construir la tabla de escalas para tabular la información.	O(L)
TOTAL	***O(A+v + e+L) -> el número de vértices debe ser mayor que las escalas L, entre todos los aeropuertos.***

Análisis

Primero se encuentra el aeropuerto más cercano a las coordenadas ingresadas por el usuario mediante el método de Harvesine, el cual toma una complejidad de v, siendo v el número de aeropuertos de grafo ya que los debe comparar a las coordenadas originales y decidir cuál es el más cercano. Una vez obtenido el aeropuerto más cercano, se verifica que se cumpla la regla de la distancia lo cual es O(1) ya que es simplemente una comparación numérica. Después de todo esto, se realiza un recorrido DFS del grafo para hallar la existencia de un camino entre dos destinos, esto se realiza en O(e + v), siendo e el número de arcos en el grafo y v el número de vertices en el grafo, dando a entender que se recorrió todo el grafo por completo. Por último, se construye la tabla de escalas para tabular la información obtenida, lo cual toma una complejidad de O(L), siendo L la longitud de la lista a tabular.

Requerimiento 2

Descripción

Este requerimiento se encarga de identificar el itinerario con menos escalas entre dos destinos turísticos.

Entrada	Punto de origen (localización con latitud y longitud), punto de destino (localización con latitud y longitud)
Salidas	Tiempo de ejecución, Distancia total del camino entre el punto de origen y el de destino, El número de aeropuertos que se visitan en el camino encontrado, Información acerca del camino encontrado.
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Encontrar los aeropuertos más cercanos a las coordenadas input usando el método de Harvesine	O(A) A: número total de aeropuertos
Verificar que las distancias sean menores o iguales a 30, si no se cumple la función se acaba.	O(1)
Encontrar el camino con el menor # de escalas entre dos destinos turísticos mediante BFS.	O(v + e) e: edges, v: vértices en vuelos_comerciales
Construir la tabla de escalas para tabular la información.	O(L)
TOTAL	***O(A+v + e+L) -> el número de vértices debe ser mayor que las escalas L, entre todos los aeropuertos.***

Análisis

Primero se encuentra el aeropuerto más cercano a las coordenadas ingresadas por el usuario mediante el método de Harvesine, el cual toma una complejidad de v, siendo v el número de aeropuertos de grafo ya que los debe comparar a las coordenadas originales y decidir cuál es el más cercano. Una vez obtenido el aeropuerto más cercano, se verifica que se cumpla la regla de la distancia lo cual es O(1) ya que es simplemente una comparación numérica. Después de todo esto, se realiza un recorrido BFS del grafo para hallar la existencia de un camino entre dos destinos, esto se realiza en O(e + v), siendo e el número de arcos en el grafo y v el número de vertices en el grafo, dando a entender que se recorrió todo el grafo por completo. Por último, se construye la tabla de escalas para tabular la información obtenida, lo cual toma una complejidad de O(L), siendo L la longitud de la lista a tabular.

Requerimiento 3

Descripción

Este requerimiento se encarga de determinar la red de trayectos comerciales de cobertura máxima desde el aeropuerto con mayor concurrencia.

Entrada	Carga de Datos

Salidas	Aeropuerto más importante según la concurrencia comercial, número de trayectos desde este, la suma de la distancia de los trayectos, y la secuencia de trayectos encontrados.
Implementado (Sí/No)	Sí, por Ana Fadul

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Se encuentra el aeropuerto de mayor concurrencia	O(V) -> V de vértices para vuelos comerciales
Se convierte un grafo de directo a indirecto	O(V+E)
Comandos de array lists	O(1)
Algoritmo de Prim	O(E*logV)
Se encuentran las vertices dentro del prim	O(V)
Se recorren todos los vértices conectados en el prim	O(V)
Se encuentra el recorrido del prim, el peso del recorrido, y agrega a listas	O(1)
TOTAL	O(E*logV)

Análisis

El requerimiento tiene una complejidad de O(E*logV), al considerar E como el número total de arcos, y a V como el número total de verticles. Al encontrar el aeropuerto con más frecuencia es O(V) debido a que se debe comparar cada vértices, y el convertir el grafo a indirecto es O(V+E), ya que se tienen que recorrer todas las vértices y ejes conectados al grafo. La mayor complejidad teniéndola el algoritmo de PRIM al crear el MST con O(E*logV). Después de crear el MST, se encuentran sus vertices y se recorren teniendo una complejidad de O(V). Finalizando, podemos asumir que la reconstrucción del algoritmo de prim, y la reconstrucción de los pesos en un camino son O(1) debido a la implementación con hash table de los grafos. Debido a esto el req tiene una complejidad de O(E*logV)

Requerimiento 4

Descripción

Este requerimiento se encarga de

Entrada	Carga de Datos
Salidas	Aeropuerto más importante según la concurrencia, número de trayectos desde este, la suma de la distancia de los trayectos, y la secuencia de trayectos encontrados.
Implementado (Sí/No)	Si, por Nicolás Parra

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
TOTAL	

Análisis

Requerimiento 5

Descripción

Este requerimiento se encarga de

Entrada	Data_structures
Salidas	Info el aeropuerto más concurrido y su concurrencia, la distancia total del trayecto, el número de trayectos, y el tabulate de todos los posibles trayectos.
Implementado (Sí/No)	Si, por Manuela

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Se calcula el aeropuerto de mayor concurrencia y se vuelve el grafo no dirigido	O(V+E) e: edges, v: vértices en vuelos_militares_min
Convertimos vuelos_militares_km a no dirigido para hacer el prim.	O(V+E)
Se hace el prim para obtener el mst en kilometros	O(E log V)
Se calcula el numero de trayectos	O(1)
Se obtienen los vertices adyacentes del prim	O(V)
Hacer una tabla con todos los vertices, y cada que se agrega a la tabla se hace un recorrido MST	O(V ^ 2) por que recorrido mst es O(V) y se hace V veces
TOTAL	***O(V ^2)***

Análisis

Primero calculamos el aeropuerto mas concurrido, para hacer esto se convierte en no dirigido y se cuentan sus vecinos. Luego creamos un mst desde el aeropuerto mas concurrido con un prim, pero para hacer esto primero toca volver el grafo no dirigido. Con el mst hecho calculamos el posible numero de trayectos buscando cuantos arcos tiene el grafo. Obtenemos la informacion relevante del origen y sacamso la lista de todos los vertices del grafo para obetener la informacion de estos y usando un recorridoMST obtenemos el trayecto. Toda esta ultimo informacion se devuelve en un tabulate.

Requerimiento 6

Descripción

Este requerimiento se encarga de obtener los caminos más cortos para la cobertura de los N aeropuertos más importantes del país.

Entrada	Carga de datos, y el número de aeropuertos a visitar
Salidas	Timpo del algoritmo, aeropuerto con mayor concurrencia comercial, y los caminos hacía el N número de aeropuertos.
Implementado (Sí/No)	Sí

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Encontrar valor en hash tables, y crear listas base.	O(1)
Dijkstra para todo el grafo	O(E*log(V))
Recorrer la lista de aeropuertos según la concurrencia comercial	O(V)
If statements, creación de listas, y pathTo djkstra (por implementación)	O(1)
Recorrido a través del camino entre aeropuertos	O(1)
Agregar valor a listas	O(1)
TOTAL	***O(E*log(V))***

Análisis

El requerimiento tiene una complejidad de O(E*log(V)), al tomar E como el número total de arcos y V como el número total de vértices. Esto considerando que al crear listas y hashes se tiene una complejidad de O(1). El algoritmo de dijstra tiene una complejidad de O(E*log(V)). Después, se recorre la lista de aeropuertos por concurrencia comercial, teniendo una complejidad de O(V). Por la implementación de hashes de los grafos la recreación de los caminos es de O(1), y al ya tener el algoritmo de Dijkstra el path to también es O(1). Los for loops dentro del código, al ser de los recorridos siempre tienen una complejidad de O(3) o O(2), por ende, se puede asumir su complejidad de O(1).

Requerimiento 7

Descripción

Este requerimiento se encarga de obtener el camino más corto en tiempo para llegar entre dos lugares turísticos.

Entrada	Coordenadas de origen y destino, datastructures
Salidas	Tiempo de ejecucio, el tiempo de viaje total, la distancia total, el total de aeropuertos, la distancia, y un tabulate del trayecto
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Primero se encuentran los aeropuertos más cercanos a las coordenadas ingresadas, usando Haversine y recorriendo una lista de aeropuertos.	O(A), donde A es el número de aeropuertos
Se verifican que las distancias sean menores o iguales de 30, si no lo son se obtienen los hash de los aeropuertos y se acaba la función.	O(1)
Se usa Dijkstra para encontrar los caminos más cortos desde el aeropuerto de origen	O(E log V) e: edges, v: vértices en vuelos_comerciales_min
Del Dijkstra se obtiene el camino hasta el aeropuerto destino	O(1)
Se calcula la distancia del trayecto y se le suma las distancias del origen y destino hasta sus respectivos aeropuertos	O(L) donde L es la longitud del camino
Se construye la lista de paradas para tabular	O(L)
TOTAL	***O(A + E log V + L)***

Análisis

La función utiliza Haversine para calcular las distancias desde las coordenadas hasta los aeropuertos más cercanos. Si alguna de estas distancias es mayor a 30 Km no se realiza la búsqueda del trayecto más corte si no que se retorna la información de los aeropuertos más cercanos y la distancia de la que están de las coordenadas iniciales. Si no se usan estas coordenadas para buscar la ruta más corta entre ellas en minutos, o sea el trayecto más rápido, mas no el más corto. Con este camino se recorre el grafo de vuelos comerciales con pesos en km y se recrea el trayecto para saber cuál es la distancia desde el aeropuerto de origen hasta el aeropuerto destino. Y todo este trayecto se muestra en un tabulate.