Estimación de complejidades

Universidad de los Andes

Universidad de los Andes

Estructuras de Datos

**PROYECTO 3 – ESTRUC TURAS DE DATOS 201810**

**Requerimiento 1:** *Mostrar en la consola de texto la información del vértice más congestionado en Chicago (aquel que contiene la mayor cantidad de servicios que salen y llegan a él): su (latitud, longitud), total servicios que salieron y total de servicios que llegaron.*

En este método se recorrerá todo el grafo para buscar aquel vértice que cuente con una mayor cantidad de servicios (que salen y llegan). Por este motivo, se estima que se debe hacer un recorrido total de este para obtener su información. Hay varios algoritmos que nos permiten realizar esta búsqueda indirectamente, como DFS y BFS. Estos algoritmos cuentan con una complejidad de O(E+V)[[1]](#footnote-1).

Además, es importante decir que para no tener que realizar verificaciones de más en la obtención de los servicios, se emplearán dos listas por vértice que almacenan los servicios que llegaron y los que salieron respectivamente. Estas listas serán manejadas y estarán contenidas en la clase Ubicación.

**Requerimiento 2:** *Calcule los componentes fuertemente conexos presentes en el grafo. Asígnele un color a los vértices que componen un componente. Retorne una lista en donde en cada nodo, se tiene la información de un componente conexo (color, número de vértices que lo componen). Muestre en la consola de texto el total de componentes identificadas y la información de la lista de componentes conexo.*

En este método debemos tener en cuenta que el algoritmo de Kosaraju es el ideal. Partiendo de esta escogencia podemos decir que la complejidad para hallar las componentes del grafo será de O(V+E).

**Requerimiento 3:** *A partir del grafo cargado al inicio y de los componentes conectados encontrados en el punto 2, genere un mapa coloreado de la red vial de Chicago utilizando Google Maps, de la siguiente forma:*

*Para cada vértice del grafo, calcule el radio del círculo de acuerdo a la "densidad" de servicios que salen y llegan a dicho vértice*

En este método implementaremos de manera activa el requerimiento anterior, en otras palabras, nos basaremos en los datos hallados para poder visualizar en Google Maps las componentes y sus respectivos vértices con colores distintivos. Respecto a complejidades podemos hacer referencia a la solicitada densidad de servicios la cual consiste en una operación matemática simple que se empleará con los tamaños de las listas de servicios almacenadas en cada vértice a partir de la clase Ubicación. Por ende, esto costaría aproximadamente O(1).

**Requerimiento 4:** *Encontrar el camino de costo mínimo (menor distancia) para un servicio que inicia en un punto (latitud, longitud) escogido aleatoriamente de la información cargada del archivo de calles (StreetLines.csv) y finaliza en un punto (latitud, longitud) escogido también de manera aleatoria del archivo de calles. Inicie el camino en el vértice del grafo más cercano a su punto de inicio y finalícela en el vértice más cercano del grafo a su punto de destino. Muestre en la consola de texto el camino a seguir, informando sus vértices, el tiempo estimado, la distancia estimada y el valor estimado a pagar.*

En este método se utilizará el algoritmo de Dijsktra ya que se requiere hallar el camino de costo mínimo en un grafo dirigido como es el que estamos manipulando. La complejidad de este algoritmo sería en su peor caso de O (E logV). Además, hay que tener en cuenta que los vértices que llegan del archivo ‘.csv’ no son siempre una localización incluida en el grafo, por este motivo tendríamos que aproximarle a uno de los vértices. Esta operación podría tomar una complejidad de O(V) en el peor de los casos, ya que buscaríamos todos los vértices que tengan una distancia harvesiana menor a 100 respecto a la dada. Por ende, la complejidad final de este método podría llegar a tomar O(V).

**Requerimiento 5:** *Dado un servicio que inicia en un punto (latitud, longitud) escogido aleatoriamente de la información cargada del archivo de calles (StreetLines.csv) y finaliza en un punto (latitud, longitud) escogido también de manera aleatoria del archivo de calles. Aproxime los puntos de inicio y fin a los vértices más cercanos en el grafo y encuentre los caminos de mayor y menor duración entre dichos puntos. Muestre en la consola de texto los dos caminos, y para cada uno de ellos: sus vértices, el tiempo estimado, la distancia estimada y el valor estimado a pagar.*

El método asociado a este requerimiento tendrá una complejidad de ya que tiene que realizar el algoritmo de Dijsktra dos veces. En la primera ocasión para hallar el menor costo y en la segunda para hallar el mayor. Vale aclarar que se requiere realizar una modificación a Dijsktra para así poder obtener también mayores costos. Este proceso lo llamaremos Dijsktra invertido.). Además, hay que tener en cuenta que los vértices que llegan del archivo ‘.csv’ no son siempre una localización incluida en el grafo, por este motivo tendríamos que aproximarle a uno de los vértices. Esta operación podría tomar una complejidad de O(V) en el peor de los casos, ya que buscaríamos todos los vértices que tengan una distancia harvesiana menor a 100 respecto a la dada. Por ende, la complejidad final de este método podría llegar a tomar O(V).

**Requerimiento 6:** *Dado un servicio que inicia en un punto (latitud, longitud) escogido aleatoriamente de la información cargada del archivo de calles (StreetLines.csv) y finaliza en un punto (latitud, longitud) escogido también de manera aleatoria del archivo de calles. Indique si existen caminos entre ambos puntos, en los que no deba pagar peaje. Si dichos caminos existen, retorne una lista con todos ellos, ordenadas de menor a mayor por tiempo y de mayor a menor por costo.*

En este último método se utilizará el algoritmo de Dijsktra ya que se requiere hallar el camino de menos peajes en un grafo dirigido como es el que estamos manipulando. La complejidad de este algoritmo sería en su peor caso de O (E logV). Por otra parte, hay que tener en cuenta que los vértices que llegan del archivo ‘.csv’ no son siempre una localización incluida en el grafo, por este motivo tendríamos que aproximarle a uno de los vértices. Esta operación podría tomar una complejidad de O(V) en el peor de los casos, ya que buscaríamos todos los vértices que tengan una distancia harvesiana menor a 100 respecto a la dada. Por ende, la complejidad final de este método podría llegar a tomar O(V).

1. La letra E corresponde al número de arcos y la letra V al número de vértices. [↑](#footnote-ref-1)