Tecnológico de Monterrey

E2. Actividad Integradora 2

Análisis y diseño de algoritmos avanzados (TC2038.652)

Nombre(s)	Apellidos	Matrícula	E-Mail
Jesús Alexander	Meister Careaga	A01656699	A01656699@tec.mx
Román Mauricio	Elías Valencia	A01656603	a01656603@tec.mx
Raúl Armando	Vélez Róbles	A01782488	a01782488@tec.mx

Profesor:
Cristhian Alejandro Ávila Sánchez

Agosto – Diciembre

Fecha de entrega: 29 de noviembre de 2023

Descripción del problema.

Se nos solicitó lo siguiente:
1. Leer un archivo de entrada que contiene la información de un grafo representado en forma de una matriz de adyacencias con grafos ponderados.
El peso de cada arista es la distancia en kilómetros entre colonia y colonia, por donde es factible meter cableado. El programa debe desplegar cuál es la forma óptima de cablear con fibra óptica conectando colonias de tal forma que se pueda compartir información entre cuales quiera dos colonias.

2. Debido a que las ciudades apenas están entrando al mundo tecnológico, se requiere que alguien visite cada colonia para ir a dejar estados de cuenta físicos, publicidad, avisos y notificaciones impresos. por eso se quiere saber ¿cuál es la ruta más corta posible que visita cada colonia exactamente una vez y al finalizar regresa a la colonia origen?
El programa debe desplegar la ruta a considerar, tomando en cuenta que la primera ciudad se le llamará A, a la segunda B, y así sucesivamente

3. El programa también debe leer otra matriz cuadrada de N x N datos que representen la capacidad máxima de transmisión de datos entre la colonia i y la colonia j. Como estamos trabajando con ciudades con una gran cantidad de campos electromagnéticos, que pueden generar interferencia, ya se hicieron estimaciones que están reflejadas en esta matriz.
La empresa quiere conocer el flujo máximo de información del nodo inicial al nodo final. Esto debe desplegarse también en la salida estándar.

4. Teniendo en cuenta la ubicación geográfica de varias "centrales" a las que se pueden conectar nuevas casas, la empresa quiere contar con una forma de decidir, dada una nueva contratación del servicio, cuál es la central más cercana geográficamente a esa nueva contratación. No necesariamente hay una central por cada colonia. Se pueden tener colonias sin central, y colonias con más de una central.

Identificamos los siguientes problemas:

1. Encontrar el árbol de expansión mínima, ya que debemos conectar las colonias usando la menor cantidad de cable
2. Resolver el problema del viajero, ya que la persona debe salir de una colonia, visitar todas una vez, y regresar a la colonia de origen
3. Encontrar el flujo máximo
4. Realizar diagramas de voronoi

Problemas desarrollados

Árbol de expansión mínima
Para este problema se utilizó una implementación del algoritmo de Prim, que nos permite encontrar un árbol de expansión mínima de un grafo.

El algoritmo de prim funciona de la siguiente manera: Partiendo de un nodo inicial, se ingresan al árbol el siguiente nodo más cercano adyacente al árbol que no sea parte aun del mismo. Este proceso se repite hasta terminar de añadir todos los nodos al árbol.

Complejidad:
Temporal: O(Vˆ2)
Donde V es el número de nodos o vértices del grafo.
La razón de esta complejidad es que, en cada iteración del algoritmo, necesitamos seleccionar el vértice o nodo con la distancia mínima. Para hacer esto, necesitamos recorrer todos los vértices. Esto da lugar a una complejidad de tiempo de O(V) por cada extracción de vértice con clave mínima. Como necesitamos hacer esto para todos los vértices, la complejidad total del tiempo se convierte en O(V^2).

Espacial: O(V)
Debido a que construye un array igual de largo que la cantidad de vértices.

Problema del viajero
Para este problema utilizamos un algoritmo genético, para poder reducir la complejidad temporal a una más pequeña.

Funciona generando una población de soluciones aleatorias del tipo [0,1,2,3,0], el cual representa la lista de ciudades en orden de visita. Después, se realiza un proceso de selección de las mejores soluciones, se mezclan las supervivientes y se mutan de manera aleatoria para mantener la variedad. Esto se repite por cierto número de generaciones y al final se obtiene la mejor solución encontrada.

Complejidad:
Temporal: O(G*P*N)
Donde G es el número de generaciones, P el tamaño de la población y N el número de nodos del grafo.
Esto se debe a que cada solución generada tiene un tamaño N, se generan P soluciones por cada población, y se generan G generaciones. Si N, P y G tienen el mismo valor, la complejidad converge en O(N^3)

Espacial: O(G*P*N)
Debido a que la función genera un array de tamaño N por cada solución, hay P soluciones por población y hay G poblaciones en total.

Flujo máximo
Para este problema utilizamos el algoritmo de Edmonds-Karp, lo que se busca descubrir es la cantidad de flujo máximo que puede recibir el nodo final a partir del nodo inicial.

Este algoritmo funciona encontrado caminos de aumento de flujo (que no superen la capacidad del camino). Mientras siga encontrando un camino de aumento, suma la cantidad máxima de flujo que puede pasar por ese camino al flujo máximo total.

Complejidad:
Temporal: O(V Eˆ2)
Donde V son los vértices del grafo y E son las aristas.
Encontrar el camino de aumento como mucho tiene una complejidad temporal de O(E), pues utiliza BFS, además, el algoritmo de Edmonds-Karp realiza a lo sumo O(V E) iteraciones, ya que cada vez que se satura una arista (es decir, se le asigna el máximo flujo posible), la distancia del camino más corto entre la inicio y el nodo destino, aumenta en al menos una unidad, y esta distancia no puede ser mayor que V. Por lo tanto, el costo total del algoritmo es O(V E) x O(E) = O(V E^2).

Diagrama de Voronoi:

Los diagramas de Voronoi son representaciones gráficas que dividen un espacio en regiones basadas en la proximidad a un conjunto específico de puntos. Cada región comprende el área más cercana a un punto dado en comparación con otros puntos en el conjunto.
Con ellos, podemos cumplir lo que se nos pide en el cuarto punto de la entrega. Podemos implementarlos fácilmente usando la librería scipy de python.

Conclusiones

Finalmente, podemos concluir que esta actividad nos muestra cómo podemos implementar el uso de algoritmos avanzados para encontrar soluciones a problemas de la vida real. Además, ayudó a mejorar nuestras capacidades en la implementación y diseño de algoritmos, específicamente de algoritmos de grafos.

Referencias

GeeksforGeeks. (2023b, noviembre 24). *PRIMS Algorithm for Minimum Spanning Tree MST*. https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/

Edmonds-Karp Algorithm | Brilliant Math & Science Wiki. (s. f.). https://brilliant.org/wiki/edmonds-karp-algorithm/