



**Tecnológico
de Monterrey**

Instituto Tecnológico de Estudios Superiores Monterrey

CAMPUS QUERÉTARO

Análisis y diseño de algoritmos avanzados

Ramona Fuénte Valdéz

Grupo 602

AI2_Actividad Integradora 2

PRESENTAN

Rodrigo Terán Hernández

A01704108

Fecha:
18/11/2023

Índice

Índice	2
Problemática	3
Introducción	3
Recorridos a cada ciudad	3
Propuesta de solución	3
Captura de pantalla:	3
Recorriendo todas las ciudades	4
Propuesta de solución	4
Captura de pantalla:	4
Flujo máximo	5
Propuesta de solución	5
Captura de pantalla:	5
Nuevas centrales	6
Propuesta de solución	6
Captura de pantalla:	6
Ejemplo	7
Conclusiones	8
Referencias	8

Problemática

Introducción

En la situación problema se nos presenta un escenario en donde se tienen unas ciudades en las cuales se necesitan conectar por los proveedores de Internet y el problema es saber cuál es la manera más óptima de interconectar estas ciudades.

Recorridos a cada ciudad

Queremos conectar con fibra óptica todas las ciudades entre sí. Pero la fibra óptica no es barata, así que queremos hacerlo de la manera más eficiente posible.

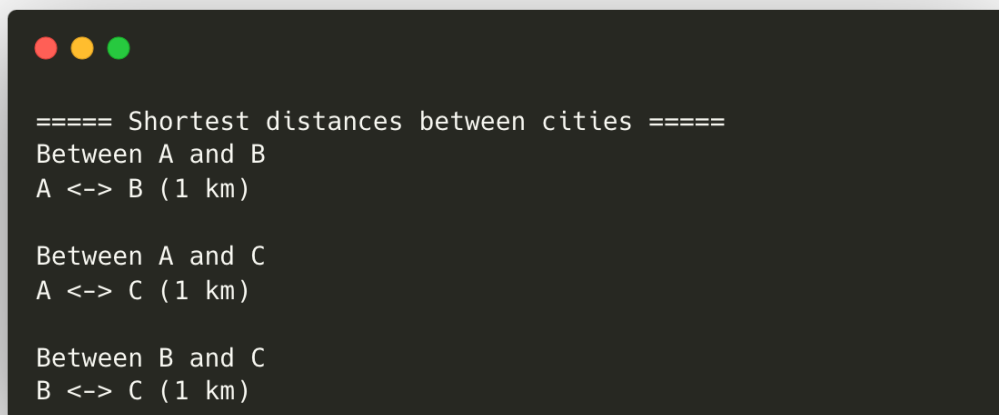
Propuesta de solución

Es por eso que se utilizó el algoritmo de Dijkstra para obtener los caminos más cortos de una ciudad a todas las demás ciudades. Este algoritmo tiene una complejidad de:

$$O(n \log n)$$

Siendo n el número de nodos.

Captura de pantalla:

A screenshot of a terminal window with a dark background and light gray text. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The text inside the terminal displays the shortest distances between three cities: A, B, and C. It shows that the distance between A and B is 1 km, between A and C is 1 km, and between B and C is 1 km.

```
==== Shortest distances between cities ====  
Between A and B  
A <-> B (1 km)  
  
Between A and C  
A <-> C (1 km)  
  
Between B and C  
B <-> C (1 km)
```

Recorriendo todas las ciudades

Ahora se nos plantea la necesidad de recorrer todas las ciudades sin pasar por la misma ciudad más de una vez ya que se necesitan entregar activos en físico. ¿Cómo podemos lograr esto de la manera más eficiente posible?

A este problema se le conoce como el problema del viajante.

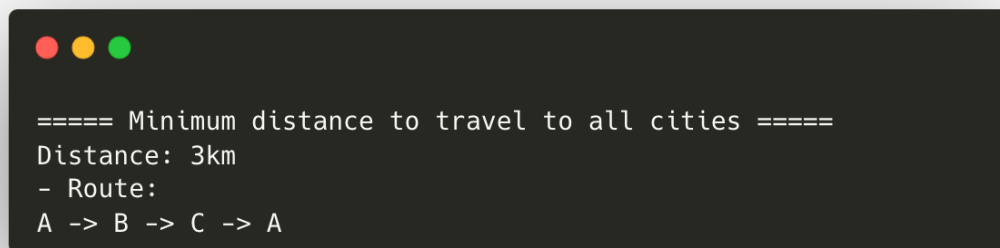
Propuesta de solución

Este problema aún no tiene una solución definitiva ya que es un problema NP. Entonces en este reto presentamos una solución usando programación dinámica. Este algoritmo tiene una complejidad de:

$$O(n^2 \times 2^n)$$

Siendo n el número de nodos.

Captura de pantalla:

A screenshot of a terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal displays the following text:

```
===== Minimum distance to travel to all cities =====  
Distance: 3km  
- Route:  
A -> B -> C -> A
```

Flujo máximo

Se nos presenta el problema de saber cuál es la mayor cantidad de flujo que se puede mandar en la transmisión de datos.

Propuesta de solución

Para este problema se implementó el algoritmo de grafo de flujo máximo que estudiamos en clase para obtener cuál es el flujo máximo que puede entrar por todo el grafo. Este algoritmo tiene una complejidad de:

$$O(n \times m)$$

Siendo n el número de conexiones y m la capacidad máxima del grafo.

Captura de pantalla:



Nuevas centrales

Por último se presenta la oportunidad de instalar una nueva central y se requiere saber a cuál conectarse. Por eso necesitamos saber cuál es la central que está más cercana a nuestro punto de coordenadas.

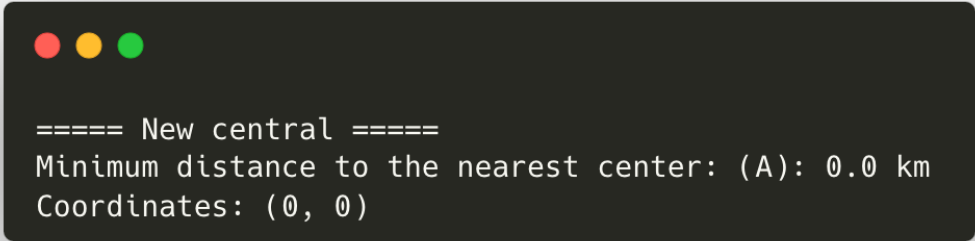
Propuesta de solución

Para este problema se planteó obtener la distancia de nuestro punto de coordenadas a cada central y obtener la más cercana. Este algoritmo tiene una complejidad de:

$$O(n)$$

Siendo n el número de ciudades.

Captura de pantalla:



```
==== New central ====  
Minimum distance to the nearest center: (A): 0.0 km  
Coordinates: (0, 0)
```

Ejemplo

Este es un ejemplo de la salida de el programa implementado:

```
Input: ./inputs/Individual_Entrada_1.txt
===== Shortest distances between cities =====
Between A and B
A <-> B (16 km)

Between A and C
A <-> B <-> C (34 km)

Between A and D
A <-> D (32 km)

Between B and C
B <-> C (18 km)

Between B and D
B <-> D (21 km)

Between C and D
C <-> D (7 km)

===== Minimum distance to travel to all cities =====
Distance: 73km
- Route:
A -> B -> C -> D -> A

===== Flow =====
Max flow: 78

===== New central =====
Minimum distance to the nearest center: (B): 103.07764064044152 km
Coordinates: (300, 100)
```

Conclusiones

El uso de estos algoritmos fue fundamental ya que el manejo de grafos se plantea muy fácil pero cuando se necesita obtener información valiosa no siempre la manera en la que se implementan las soluciones eficientes suelen ser tan triviales.

La naturaleza de la interconexión de dispositivos juega un papel muy importante en el ahorro de recursos y de una manera muy sencilla se pueden plantear grafos para su implementación. De hecho, el direccionamiento IP funciona con algoritmos de grafos como Dijkstra para mandar listas de ruteo entre ruteadores con el sistema de OSPF.

Así que en la vida real el conocer cual es la manera más eficiente de llegar de un punto A a un punto B muchas veces no es tan fácil como correr un programa ya que necesitaremos los datos como los pesos del grafo, pero una vez tengamos esa información podremos resolver cualquier problema.

Referencias

Python Program for Dijkstra's shortest path algorithm. Geeks for Geeks. Recuperado de:

<https://www.geeksforgeeks.org/python-program-for-dijkstras-shortest-path-algorithm-greedy-algo-7/>

Travelling Salesman Problem using Dynamic Programming. Recuperado de:

<https://www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/>

Algoritmos de flujo máximo y mínima de costo. Recuperado de:

<https://www.pypro.mx/app/curso/analisis-de-grafos-con-python-y-networkx/algoritmos-de-flujo-maximo-y-minima-de-costo>