Optimización de la Red de Centros de Atención de Emergencias en Costa Rica: Modelado y Análisis mediante Algoritmos de Grafos

Josué Torres Sibaja, C37853, josue.torressibaja@ucr.ac.cr

Breve planteamiento del problema a resolver

La Comisión Nacional de Emergencias de Costa Rica (CNE) enfrenta el reto de optimizar la gestión de sus centros de atención de emergencias. Estos centros, distribuidos en diversas ciudades del país, cumplen funciones críticas como planificación, almacenamiento de suministros y atención de emergencias. Dado que algunas ciudades están conectadas por rutas directas con tiempos de viaje conocidos, mientras que otras requieren pasar por ciudades intermedias, surge la necesidad de gestionar eficientemente esta red de transporte.

Para ello, se requiere modelar esta red como un grafo no dirigido, donde los vértices representan las ciudades con centros de atención de emergencias, y las aristas representan las conexiones directas entre ciudades, con un peso asociado que indica el tiempo promedio de viaje. A partir de este grafo, es necesario desarrollar un sistema que permita responder preguntas estratégicas, como:

¿Cuál es la mejor ciudad para centralizar equipos de manera que el tiempo total de viaje hacia otras ciudades sea mínimo?

¿Desde qué ciudad es más eficiente despachar equipos a una ciudad específica para reducir el tiempo de llegada?

¿Qué ciudades están más y menos distantes entre sí?

¿Cómo se ordenan las ciudades según el tiempo promedio de viaje hacia todas las demás?

El objetivo es proporcionar a la CNE una herramienta interactiva que facilite la toma de decisiones basada en datos, apoyándose en algoritmos de grafos para calcular rutas óptimas y distancias. Esto mejorará la capacidad de respuesta ante emergencias y la gestión de recursos.

Descripción de la solución

La solución implementada modela la red de centros de atención de emergencias como un grafo no dirigido ponderado. En este grafo, los nodos representan las ciudades y las aristas representan las rutas directas entre ellas, con un peso asociado que indica el tiempo promedio de viaje. Para resolver los problemas planteados, se utiliza el algoritmo de Floyd-Warshall para calcular todas las distancias mínimas entre pares de ciudades, complementado con funciones específicas que procesan la información del grafo según las necesidades del usuario (las preguntas que se quieran responder).

Justificación de la solución

Para la solución del problema se utilizó el algoritmo de Floyd-Warshall, el cual calcula las distancias mínimas entre todos los pares de nodos en el grafo (Cormen et al., 2022). Este algoritmo ofrece una gran facilidad de implementación, ya que se implementa de forma directa mediante

programación matricial, lo que lo hace una opción práctica para obtener una matriz de distancias mínimas que puede ser reutilizada para responder múltiples consultas sin recalcular rutas.

Así mismo, la modularidad de la solución permite separar el procesamiento del grafo (cálculo de distancias mínimas) de las funciones que responden y procesan las consultas específicas. Esto permite reutilizar la misma matriz de distancias mínimas para responder todas las preguntas.

La eficiencia del algoritmo de Floyd-Warshall lo hace de gran utilidad, debido a que es un algoritmo estándar y eficiente para resolver el problema de las distancias mínimas en grafos densos o completamente conectados, como es el caso de una red de transporte entre ciudades. Permite calcular todas las distancias mínimas entre pares de nodos, teniendo una complejidad temporal de $\Theta(V^3)$, donde V es el número de ciudades (Cormen et al., 2022).

Resultados obtenidos

Los archivos con las respuestas para cada grafo se encuentran en el mismo directorio que este reporte (carpeta *doc*), dentro de la carpeta *answers*. Además, se proveen las imágenes generadas por el programa para cada uno de los archivos de entrada, los cuales forman parte de las respuestas de cada archivo, pero no se agregaron a sus respectivos documentos de respuesta ya que, debido a su tamaño y complejidad, solamente son legibles cuando se hace zum sobre ellas.

Compilar, ejecutar e interactuar con el programa

Como se indicó en las instrucciones, el programa puede ser compilado con g++ de forma sencilla, y se puede ejecutar utilizando los siguientes comandos (se asume que el compilador ya está instalado, que se compilará el programa desde el directorio *source* y que se está utilizando una terminal de Linux):

```
g++ -o programa_grafos main.cpp createGraph.cpp graphInfo.cpp
./programa_grafos
```

Al iniciar el programa, se solicitará ingresar la ruta y el nombre del archivo .csv que se desea utilizar. Para esto, existe un directorio llamado *tests*, en el cual están los tres archivos de prueba, por lo que la ruta se debe escribir de la siguiente forma:

```
../tests/<nombre del archivo>.csv
```

La interacción general con el programa consiste en un menú de opciones. Este brinda siete opciones diferentes, las cuales son:

1. Ciudad donde es más efectivo colocar mayor capacidad de equipo.

2. Mejor ciudad para apoyar a una ciudad dada.

3. Ciudades más distantes.

4. Ciudades más cercanas.

5. Orden de ciudades por tiempo promedio.

6. Exportar grafo (CSV, DOT, PNG).

7. Salir.

Las primeras cinco opciones permiten responder las preguntas especificadas en las instrucciones con los datos del grafo que se introdujo. Para la pregunta 2 se debe escribir el nombre de una ciudad existente en el grafo, de lo contrario la respuesta no podrá ser procesada. La opción 6 permite crear una lista de adyacencia simplificada del grafo en formato CSV o DOT, y generarla en la carpeta *output*. A partir del archivo DOT se puede crear una imagen del grafo en formato PNG. Finalmente, la opción 7 permite finalizar el programa.

El código incluye las bibliotecas necesarias para el correcto funcionamiento del programa principal. Sin embargo, si se desea generar la imagen del grafo en formato PNG, utilizando *Graphviz*, se deben seguir estos pasos (en caso de que *Graphviz* no se haya instalado previamente):

1. Abrir una terminal.

2. Escribir los siguientes comandos:

```
sudo apt-get update
sudo apt-get install graphviz
dot -V
```

Esto debería mostrar la versión de *Graphviz* instalada, verificando que se instaló correctamente. De esta forma, el programa podrá convertir un archivo DOT a PNG, generando la imagen en la carpeta de salida (*output*).

Notas adicionales:
La documentación del código en formato Doxygen, al igual que los nombres de las variables y funciones utilizadas, están escritas en inglés. Sin embargo, las preguntas e impresiones adicionales en pantalla se hacen en español, esto para simular (y facilitar, si fuese el caso) una interacción real entre el personal de la CNE y el programa. Así mismo, se omite el uso de tildes y caracteres propios del español para evitar errores en la impresión de los mensajes.

Referencias

Cormen, T. et al. (2022). Introduction to algorithms (4th ed.). MIT Press.