Universidad de los Andes

FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
DISEÑO Y ANÁLISIS DE ALGORITMOS

TAREA 5 - PARTE 4

Sergio Pardo Gutierrez
Juan Diego Calixto
Juan Diego Yepes

19 de Abril de 2022
Bogotá D.C.

Tabla de contenidos

1 Enunciado

Una ciudad se diseño de tal modo que todas sus calles fueran de una sola vía. Con el paso del tiempo la cantidad de habitantes de la ciudad creció y esto produjo grandes trancones en algunas de las vias debido a algunos desvíos innecesarios que tienen que tomar los habitantes de la ciudad para poder llegar a sus trabajos. Por lo tanto, el alcalde tomó la decisión de ampliar algunas vias para que puedan convertirse en doble via. Dado el mapa de la ciudad y el costo de convertir cada via actual en doble via, determinar qué vias se deben convertir, de modo que se pueda transitar de cualquier punto a cualquier punto de la ciudad por dobles vias y que el costo de la conversión sea el mínimo posible.

2 Solución

Para resolver este problema primero se enlistan las entradas y salidas:

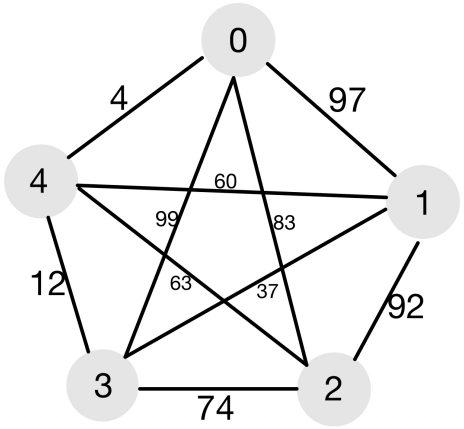
E/S	Nombre	Tipo	Significado
E	matrix	`list`	Matriz con los pesos de construir cada via. v
S	vias a contruir	`string`	e Vías a construir con su información

La entrada será una matriz de adyacencias similar a la recibida en los puntos anteriores. La única diferencia es que esta matriz debe ser simétrica para dar un resultado consistente puesto que el algoritmo solo lee la diagonal superior.

Para ilustrar mejor la entrada se toma como ejemplo la siguiente matriz:

	0	1	2	3	4
0	0	97	83	99	4
1	97	0	92	37	60
2	83	92	0	74	63
3	99	37	74	0	12
4	4	60	63	12	0

La cual daría lugar al siguiente grafo:



Por otro lado, la salida es una cadena de caracteres con el siguiente formato: "Se necesita la autopista de costo x que va desde el nodo n hasta el nodo m".

2.1 Explicacion de entradas y salidas

Podemos asumir que todo el grafo es no dirigido. Si bien todas las autopistas existentes en el problema son unidireccionales, se quiere ver cuales deberían ampliarse para ser bidireccionales. Por este motivo, puede asumirse que todas las autopistas son bidireccionales y seleccionar las autopistas de menor peso tal que todos los nodos queden conectados. De esta manera, todas aquellas que no sean parte de la respuesta simplemente se dejarán como autopistas unidireccionales.

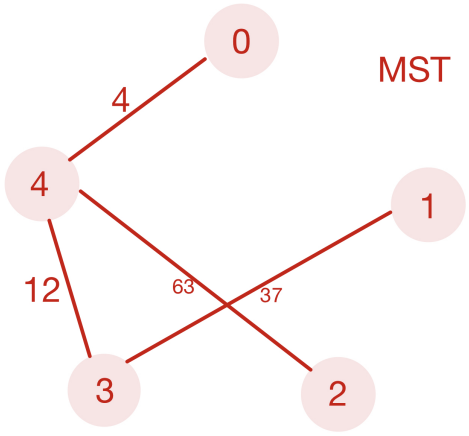
Con base en esta aproximación al problema, seleccionamos el algoritmo de Kruskal que genera un árbol de recubrimiento mínimo sobre el grafo. Este retorna una lista con los ejes de menor costo tal que formen una estructura que conecte todos los nodos. En este caso los ejes son autopistas y los nodos los lugares que conectan.

Al implementar el algoritmo desarrollado con la matriz de ejemplo el programa genera el siguiente resultado:

Se necesita la autopista de costo 4 que va desde el nodo 0 hasta el nodo 4

Se necesita la autopista de costo 12 que va desde el nodo 3 hasta el nodo 4
Se necesita la autopista de costo 37 que va desde el nodo 1 hasta el nodo 3
Se necesita la autopista de costo 63 que va desde el nodo 2 hasta el nodo 4

Lo cual se ve así:



2.2 Explicacion de la solución

El algoritmo funciona creando un diccionario que guarda los ejes con un como llave id y guardando como valor el costo de la autopista, su lugar de origen y destino. Además se crea una lista que guarda todos los id's de los ejes. Después, se ordenan los id's de los ejes con base en el peso del eje que representan. Posteriormente se itera sobre cada eje en esta lista y se verifica si crea un ciclo o no en el grafo. Si no lo crea, el eje es agregado a la lista que guarda el árbol de recubrimiento mínimo. Si crea un ciclo simplemente continúa el proceso con el siguiente eje.

Para detectar un ciclo se utiliza la estrategia de los representantes de cada partición y se aplana el árbol a través de esta estructura para reducir la complejidad temporal.

2.3 Casos de prueba

Para probar el algoritmo de Kruskal hay 3 archivos de prueba identificados con el sufijo kruskal. Hay un caso de un grafo de 5 vértices, uno de 100 y uno de 1000. Todos estos completamente conectados como el mostrado en el ejemplo. Sin embargo, el algoritmo puede probarse con cualquier otro archivo de texto y dará una respuesta con sentido, pero semánticamente incorrecta puesto que estos archivos están diseñados para grafos dirigidos y el algoritmo ignorará la mitad de la matriz. Si desea generar un caso de prueba nuevo, puede ir ejecutar el siguiente comando en la terminal `"python graphKruskal.py > distances100\textunderscore kruskal.txt"` cambiando el numero del archivo de texto (este es solo indicativo). Para que se genere el grafo con el tamaño deseado, solo debe ir a la linea 32 del archivo graphKruskal.py y cambiar el 0 por el valor deseado. Para correr el programa en terminal utilice el comando "python kruskal.py < distances5.txt" y cambie el nombre del archivo por el que desea ejecutar. Puede agregar también un archivo de salida para guardar el resultado: "python kruskal.py < distances5.py > respuesta.txt"