E.T.S.I.I.T - Grado en Ingeniería de Tecnologías de Telecomunicación
Redes de Comunicaciones - Curso 2023/2024

Práctica 1 - Implementación del algoritmo de *Dijkstra*

1. Objetivos de la práctica

Como complemento a la presentación que se hace en teoría de algunos de los algoritmos de encaminamiento más habituales, se plantea en la práctica el desarrollo de uno de ellos, en concreto el Algoritmo de *Dijkstra*, utilizando el lenguaje de programación C.

El resultado final debe ser un fichero ejecutable, al que se le pasen dos argumentos: (1) el nombre del fichero en el que está definida la red, según el formato descrito más adelante, y (2) el nodo origen $\mathcal{S}$. El resultado será las rutas de menor coste entre $\mathcal{S}$ y el resto de nodos de la red, si existen, y el coste total de cada una.

Los objetivos concretos que se plantean en la práctica son, por tanto:

- Utilización del lenguaje de programación C para implementar un algoritmo de encaminamiento.
- Asimilar la utilización de la matriz de adyacencia como representación de un grafo.
- Desarrollar la funcionalidad que permita obtener las rutas completas interpretando la información correspondiente a los predecesores.
- Proteger el código frente a la posibilidad de que no existiera ruta de $\mathcal{S}$ a algún nodo de la red.

2. Introducción

El algoritmo de *Dijkstra* es uno de los más importantes para resolver el problema de encontrar los caminos de coste mínimo en un grafo dirigido, en el que todos los enlaces tienen que tener un coste positivo. Desde que fue propuesto por el matemático e informático Edsger W. Dijkstra en 1956, se han propuesto varias modificaciones y extensiones, y se ha utilizado para resolver diferentes problemas.

Existen numerosos textos en los que se puede encontrar información acerca de este algoritmo. En los apuntes del Tema 2 de la asignatura [1] se hace un breve resumen del mismo. En [2] se presentan algunas de las aplicaciones adicionales que puede tener el algoritmo, y se realiza un análisis de su complejidad computacional. Para una descripción más detallada se puede acudir, por ejemplo, al texto de Cormen *et al.* [3, §24.3].

Tal y como se puede ver en el pseudocódigo que se muestra en la Figura 1, el Algoritmo de Dijkstra se basa en la utilización de un conjunto $\mathcal{Q}$, del que se van eliminando los nodos a medida que se establezca la ruta de coste mínimo desde una fuente, $\mathcal{S}$ y ellos. Cuando finaliza su ejecución, se pueden establecer, por tanto, el camino de coste mínimo desde $\mathcal{S}$ al resto de nodos de la red.

```
INITIALIZATION
1. d(S) = 0
2. for all v in N but S
3.      d(v) = ∞
4. Q = N
MAIN LOOP
5. while Q ≠ ∅
6.      u vertex in Q with min{d(v)}
7.      delete u from Q
8.      for all v in Q adjacent to u
9.        if d(v) > d(u) + c(u,v)
10.             d(v) = d(u) + c(u,v)
11.             prev(v) = u
```

Figura 1: Pseudocódigo del Algoritmo de Dijkstra

3. Desarrollo

3.1. Lectura fichero entrada

Se facilita el código que procesa un fichero en el que se representa un grafo dirigido (con el formato que se especifica en el Anexo A), y genera su correspondiente matriz de adyacencia, que es la representación que se utilizará a la hora de ejecutar el Algoritmo de *Dijkstra*. La utilización de este código es completamente opcional.

3.2. Funcionamiento del programa

Se ejecutará el programa con dos argumentos: (1) fichero en el que se especifica el grafo; y (2) identificador del nodo fuente. Al ser ejecutado, se deberá mostrar por pantalla todos los pasos intermedios del algoritmo, indicando las distancias, predecesores y el conjunto $\mathcal{Q}$ en cada uno de ellos.

Además, una vez presentados todos los pasos intermedios, se indicarán las rutas **completas** desde el nodo fuente, $\mathcal{S}$ (argumento de entrada al ejecutar el programa), al resto de nodos de la red, indicando el número de saltos y su coste total.

En el Anexo B se muestra un ejemplo ilustrativo de cómo podría ser el resultado de la ejecución del programa.

Será necesario, además, que el programa funcione correctamente ante la situación en la que no exista una ruta de $\mathcal{S}$ a algún nodo de la red, indicándolo así al ejecutarse.

3.3. Entorno de desarrollo

Cada estudiante puede utilizar (en sus equipos) el entorno de desarrollo que elijan, siempre que el código se pueda compilar y ejecutar, según los requisitos que se han establecido en el entorno que se habilita en los PCs del laboratorio, y que se describe a continuación.

Se hará uso de una máquina virtual *Ubuntu* bajo el sistema operativo *Windows*. Se utilizará el usuario `alumnos`, con contraseña `telematica`.
Atención: Debido a la configuración del sistema *host* y de la máquina virtual, los alumnos deberán guardar los archivos en los que hayan trabajado para no perder los cambios, y poder empezar con ellos durante la sesión siguiente.

Para editar el código del programa se puede emplear cualquier aplicación de edición de texto o código. En el entorno de desarrollo del laboratorio están disponibles, al menos, `gedit` y `sublime`. Por otro lado, para compilar un fichero en el Sistema Operativo *Linux* se tiene que emplear el siguiente comando (en un `terminal`), dentro de la carpeta en la que se encuentre el código fuente (por ejemplo, `dijkstra.c`): `gcc dijkstra.c -o dijkstra`. Se generará un fichero `dijkstra`, que será el comando que se tendrá que utilizar para ejecutar el programa (`./dijkstra arg1 arg2`).

4. Finalización de la práctica

Al finalizar la práctica y comprobar que el comportamiento del código se ajusta a los requisitos presentados anteriormente, se deberá mostrar su correcto funcionamiento al profesor responsable del grupo. Además, se preparará un informe según lo que recoge en el Anexo C.

La fecha límite para presentar el correcto funcionamiento del programa, y para subir en la tarea correspondiente en el Aula Virtual (*Moodle*) tanto el informe como el código (fichero .c) es el **12 de abril**.

5. Referencias

[1] Ramón Agüero. *Tema 2. Algoritmos de encaminamiento. Redes de Comunicaciones. Open Course Ware de la Universidad de Cantabria. Enseñanzas Técnicas.* URL: `http://ocw.unican.es` (visitado 10-02-2024).

[2] Ramón Agüero. *Tema 2. Algoritmos sobre grafos. Dimensionado y Planificación de Redes. Open Course Ware de la Universidad de Cantabria. Enseñanzas Técnicas.* URL: `http://ocw.unican.es` (visitado 10-02-2024).

[3] Thomas H. Cormen et al. *Introduction to Algorithms.* 3.ª ed. The MIT Press, 2009. ISBN: 978-0262033848.

Anexo A. Formato del fichero de entrada

La primera línea del fichero de entrada contiene el número de nodos de la red y, a partir de la segunda línea, se representa el grafo utilizando una línea por enlace, con el formato `origen,destino,coste`, como se muestra en la Figura 2.

```
4
1,2,8
1,4,1
2,3,1
3,1,4
4,2,2
4,3,9
```

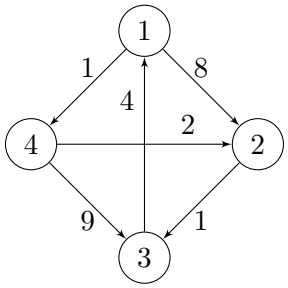

Figura 2: Formato del fichero de entrada para un grafo ilustrativo

Anexo B. Ejemplo de ejecución

A continuación se muestra una posible representación de la salida obtenida al ejecutar el programa con el grafo de la Figura 2, utilizando como nodo fuente el 1.

```
Number of nodes is 4
Adding a link from [  1] to [  2], with a cost of 8
Adding a link from [  1] to [  4], with a cost of 1
Adding a link from [  2] to [  3], with a cost of 1
Adding a link from [  3] to [  1], with a cost of 4
Adding a link from [  4] to [  2], with a cost of 2
Adding a link from [  4] to [  3], with a cost of 9

Source node is  1

Initialization:
Distances: 0 999 999 999
Predecessors: 1 0 0 0
Q: 1 2 3 4

Iteration  1:
Deleting [ 1] from Q
Distances: 0 8 999 1
Predecessors: 1 1 0 1
Q: 2 3 4

Iteration  2:
Deleting [ 4] from Q
Distances: 0 3 10 1
Predecessors: 1 4 4 1
Q: 2 3

Iteration  3:
Deleting [ 2] from Q
Distances: 0 3 4 1
Predecessors: 1 4 2 1
Q: 3

Iteration  4:
Deleting [ 3] from Q
Distances: 0 3 4 1
Predecessors: 1 4 2 1
Q:

Dijkstra output: routes from 1 to the other nodes
The route between 1 and 2 has 2 links and a cost of 3
1 -> 4 -> 2

The route between 1 and 3 has 3 links and a cost of 4
1 -> 4 -> 2 -> 3

The route between 1 and 4 has 1 links and a cost of 1
1 -> 4
```

Anexo C. Informe

Además del código desarrollado, cada estudiante deberá entregar un documento (formato `pdf`) con el contenido descrito seguidamente.

- Breve resumen del código desarrollado, describiendo sus funcionalidades más importantes. Se podría hacer referencia a las líneas correspondientes del pseudocódigo que se muestra en la Figura 1.
- Se deberá utilizar un grafo y describir el comportamiento del código al ejecutarlo sobre el mismo, comparándolo con el procedimiento teórico del algoritmo.
- Independientemente de que tenga que subirse a la tarea en *Moodle*, el informe incluirá, como anexo, una copia del código fuente desarrollado.

Se valorará positivamente estructurar adecuadamente el código (mediante el uso de funciones, cuando sea procedente), así como la utilización de comentarios que permitan interpretarlo de manera más sencilla. Además, se tendrá en cuenta la correcta estructura del documento, su formato, así como la claridad de las explicaciones.