1º Cuatrimestre 2013 **Ejercicio Nº 2 - Dijkstra**	**Taller de Programación I (75.42)** **Facultad de Ingeniería** **Universidad de Buenos Aires**

Ejercicio Nº 2 - Dijkstra

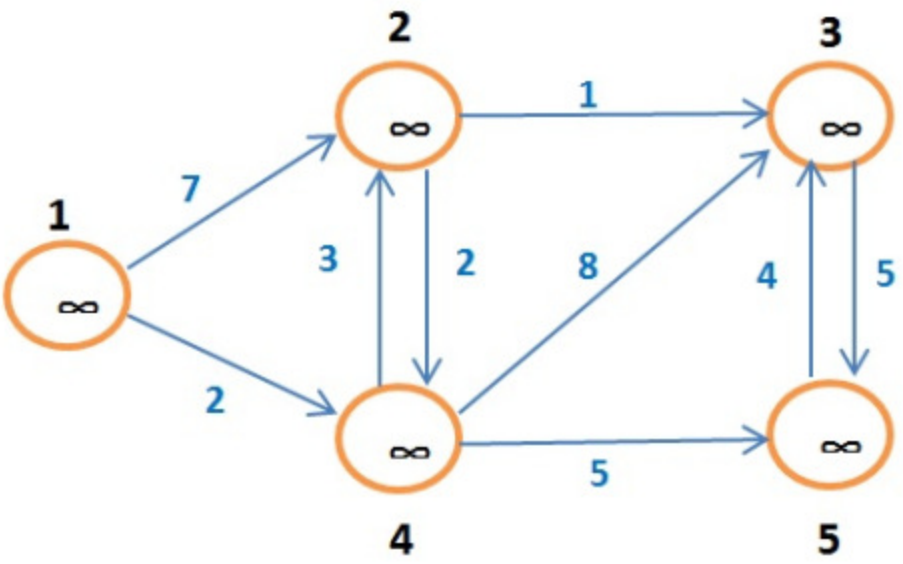
Introducción

Debido a los constantes problemas de tráfico de red que posee la empresa ITCorp S.A, se nos ha encargado el desarrollo de un módulo para el cálculo de ruteo entre PCs a través de routers contenidos en una red para de esta forma poder configurar de forma manual el ruteo entre PCs y minimizar el recorrido de los paquetes de datos.

El desarrollo del trabajo permitirá aplicar y afianzar conceptos y conocimientos del lenguaje de programación C, en particular la utilización de memoria dinámica, TDAs y estructuras agregadas.

Para resolver el problema del camino de ruteo entre dos PCs se solicita utilizar el Algoritmo de Dijkstra que tiene como objetivo calcular el camino mínimo entre dos nodos.

El algoritmo de dijkstra determina la ruta más corta desde un nodo origen hacia los demás nodos para ello es requerido como entrada un grafo cuyas aristas posean pesos.



A continuación se muestra un pseudocódigo que explica el funcionamiento del mismo

Algoritmo

Teniendo un grafo dirigido ponderado de N nodos no aislados, sea x el nodo inicial, un vector D de tamaño N guardará al final del algoritmo las distancias desde x al resto de los nodos.

1. Inicializar todas las distancias en D con un valor infinito relativo ya que son desconocidas al principio, exceptuando la de x que se debe colocar en 0 debido a que la distancia de x a x sería 0.
2. Sea a = x (tomamos a como nodo actual).
3. Recorremos todos los nodos adyacentes de *a*, excepto los nodos marcados, llamaremos a estos nodos no marcados vi.
4. Si la distancia desde x hasta vi guardada en D es mayor que la distancia desde x hasta a, sumada a la distancia desde a hasta vi; esta se sustituye con la segunda nombrada, esto es:
5. si (Di > Da + d(a, vi)) entonces Di = Da + d(a, vi)
6. Marcamos como completo el nodo a.
7. Tomamos como próximo nodo actual el de menor valor en D (puede hacerse almacenando los valores en una cola de prioridad) y volvemos al paso 3 mientras existan nodos no marcados.

Una vez terminado el algoritmo, D estará completamente lleno.

Descripción

Se deberá implementar un programa de consola que reciba la configuración de la red, con los hosts (PCs) conectados, los dispositivos de ruteo y el ruteo entre ellos. Una vez especificada la configuración se utilizará el algoritmo de Dijkstra para encontrar el camino mínimo para la comunicación entre los dispositivos que se encuentran conectados. Se escribirá por salida estándar el camino que debe recorrer un paquete de datos para llegar desde el host que se encuentra conectado en el dispositivo origen hacia los demás hosts que se encuentran conectados en la red.
Ejemplo: si se definen 3 hosts (A, B y C) se escribirá por salida estándar el camino desde A (suponiendo que era el que estaba conectado en el origen) hacia B y el de A hacia C). El orden en el que se definen las conexiones es el orden en el que se encuentran definidos en el archivo de configuración.

Formato de línea de comandos

La aplicación se ejecutará como ./tp [archivo_ruteo] donde [archivo_ruteo] es un parámetro opcional que indica el archivo que contiene la especificación de ruteo que se encuentra configurada en la red y los host que se encuentran conectados en la misma. De no especificarse, se recibirán la especificación por entrada estándar.
El resultado de la ejecución será el camino que se deberán recorrer los paquetes de datos

desde el host origen, a través de los dispositivos de ruteo, hacia los demás hosts conectados en la red.

Formato de la entrada

El archivo de especificación de ruteo es un archivo de texto plano que contiene secciones (que comienzan con [nombre sección]). Luego del comienzo de la sección se especifica su contenido (1 especificación por línea).
Las secciones que se encuentran en el archivo son:

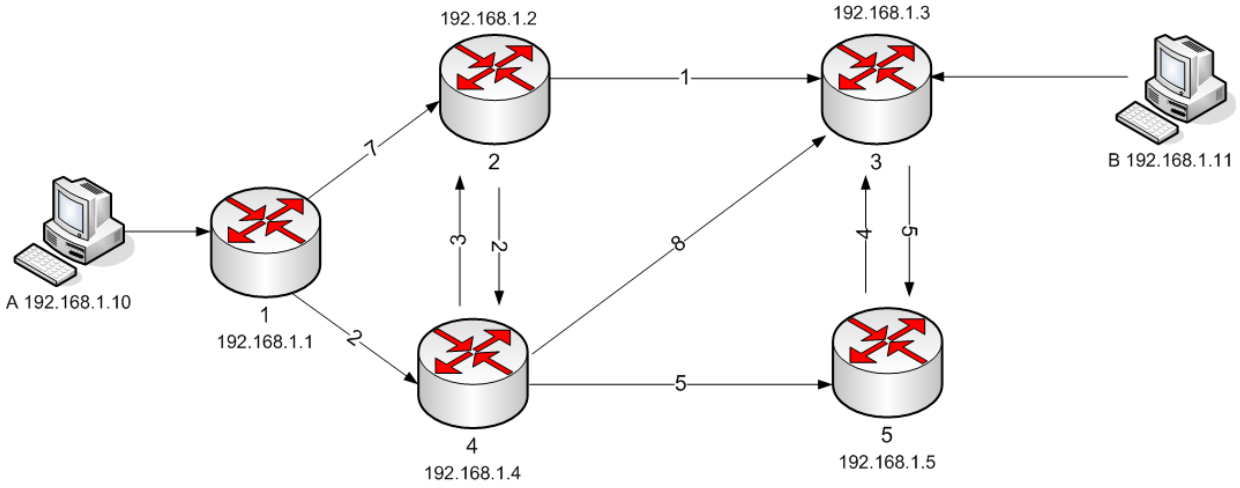
- [host]: sección que indica los host conectados la red. Dentro de esta sección se especifica los hosts que se encuentran conectados a la red.
    - Formato de especificación de host es nombre,IP,nombre_router (al cual está conectado)
    - Ejemplo: A,192.168.10.10,1 indica que el nombre del host es A, la IP del mismo es 192.168.10.10 y se encuentra conectado al dispositivo 1
- [device]: sección que contiene la definición de los dispositivos de ruteos que se encuentran configurados en la red. Dentro de esta sección vienen especificados los routers.
    - Formato de especificación de un router es nombre_roter,IP
    - Ejemplo: 1,192.168.10.1 indica que es un dispotivo con nombre “1” e IP 192.168.10.1
- [route]: sección que define la configuración de ruteo entre los dispotivos configurados dentro de la red. Dentro de esta sección se define el ruteo entre los dispositivos.
    - Formato de especificación de ruteo es: dispositivo_1->dispositivo_2,peso
    - Ejemplo: 1->2,7 indica que el dipositivo de nombre “1” se encuentra conectado al dispositivo de nombre “2” con un peso de conexión de 7.

Ejemplo de archivo de entrada

```
[host]
A,192.168.10.10,1
B,192.168.10.11,3
[device]
1,192.168.10.1
2,192.168.10.2
3,192.168.10.3
4,192.168.10.4
5,192.168.10.5
[route]
1->2,7
1->4,2
2->3,1
2->4,2
```

3->5,5
4->2,3
4->3,8
5->3,4

Representación gráfica del archivo de definición de ruteo:



Salida Estándar

Por salida estándar se imprimirá el resultado. El formato del resultado es similar al de entrada, en donde se definen secciones. Cada sección contendrá la descripción del camino a recorrer entre el host que se encuentra conectado al origen hacia los demás hosts conectados en la red.

Siempre se tomará como dispositivo origen (siempre debe existir un origen para el cálculo del camino por dijkstra) el primer dispotivo que viene especificado en la sección [device].

Siempre se toma como host origen al primer host definido en la sección [host]

Formato de las secciones:

- [route_path:origen->destino] : sección que describe el recorrido entre el host conectado al dispositivo origen y el host destino. Dentro de esta sección se detalla el recorrido.
    - Formato del recorrido: #:IP donde # es el número del paso del recorrido e IP es la IP en donde se encuentra el paso. Ejemplo: 1:192.168.10.10 indica que es el paso número 1 y su IP es 192.168.10.10

Ejemplo:
[route_path:A->B]
1:192.168.10.10
2:192.168.10.1
3:192.168.10.2
4:192.168.10.3

5:192.168.10.11

En caso de obtener más de un camino mínimo se debe elegir el camino en orden alfanumérico de acuerdo a los nombres de los routers.

Ejemplo: A partir de la configuración de red que se indica a continuaciòn, se obtienen 2 caminos mínimos.

```
[host]
A,192.168.10.10,1
B,192.168.10.11,4
[device]
1,192.168.10.1
2,192.168.10.2
3,192.168.10.3
4,192.168.10.4
[route]
1->2,1
1->3,1
2->4,1
3->4,1
```

Camino 1	**Camino 2**
1:192.168.10.10	1:192.168.10.10
2:192.168.10.1	2:192.168.10.1
3:192.168.10.2	3:192.168.10.3
4:192.168.10.4	4:192.168.10.4
5:192.168.10.11	4:192.168.10.11

Comparando caminos:

- Paso 1: mismo paso para los dos caminos.
- Paso 2: mismo paso para los dos caminos.
- Paso 3: paso 3 del camino 1 tienen como nombre “3”. En cambio en el otro camino el nombre es “4” → “3” < “4”

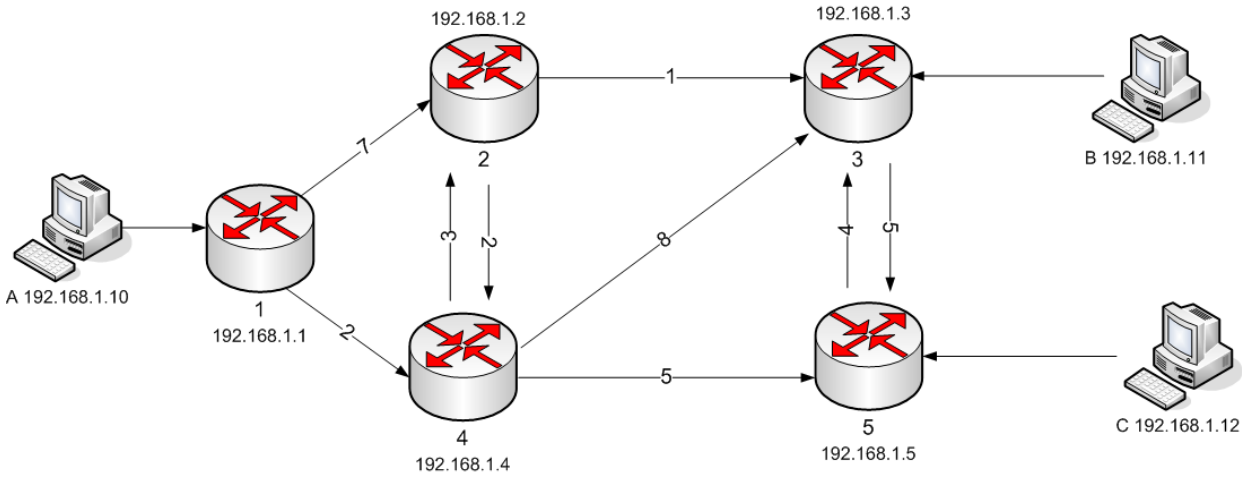
Luego de la comparación se obtiene que el camino seleccionado es el camino 1. Con lo que la salida de la ejecución del programa sería:

```
[route_path:A->B]
1:192.168.10.10
2:192.168.10.1
3:192.168.10.2
4:192.168.10.4
5:192.168.10.11
```

Códigos de retorno

El programa retorna 0 en todos los casos.

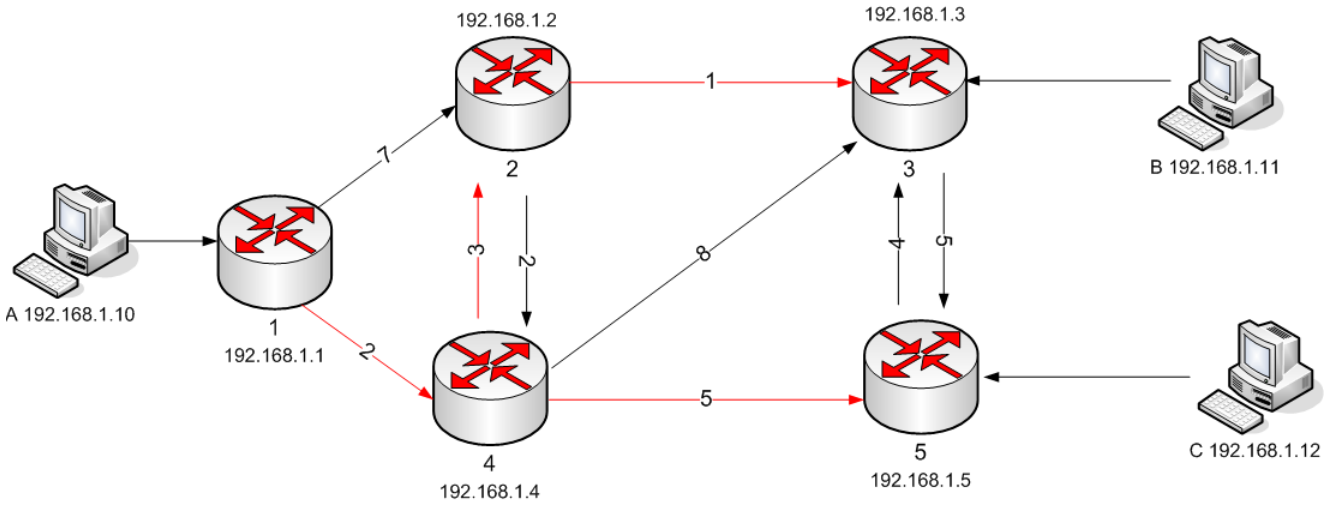
Ejemplos de ejecución



Entrada

[host]
A,192.168.10.10,1
B,192.168.10.11,3
C,192.168.10.12,5
[device]
1,192.168.10.1
2,192.168.10.2
3,192.168.10.3
4,192.168.10.4
5,192.168.10.5
[route]
1->2,7
1->4,2
2->3,1
2->4,2
3->5,5
4->2,3
4->3,8
5->3,4

Resolución de camino más corto por Dijkstra (camino solución en color rojo)



Salida

[route_path:A->B]
1:192.168.10.10
2:192.168.10.1
3:192.168.10.2
4:192.168.10.3
5:192.168.10.11
[route_path:A->C]
1:192.168.10.10
2:192.168.10.1
3:192.168.10.4
4:192.168.10.5
5:192.168.10.12

Restricciones

La siguiente es una lista de restricciones técnicas exigidas por el cliente:

1. El sistema debe desarrollarse en ISO C (C99).
2. Se puede asumir que el formato del archivo de entrada es siempre correcto.
3. Está prohibido el uso de variables globales
4. La resolución del trabajo debe estar orientada al uso de TDAs (tipos de datos abstractos), generando estructuras tales como listas, colas, etc. que provean una interfaz que oculte la implementación; está terminantemente prohibido violar el concepto de abstracción.
5. Debe utilizarse la cantidad necesaria de memoria, apelando a la reserva en forma dinámica cuando sea necesaria; no se considerará correcto utilizar memoria en exceso.
6. La implementación del algoritmo para la resolución del problema tiene que ser de autoría

del alumno que resuelve el ejercicio.

Referencias

- http://jariasf.wordpress.com/2012/03/19/camino-mas-corto-algoritmo-de-dijkstra/
- http://es.wikipedia.org/wiki/Tipo_de_dato_abstracto