Laboratorio

Link State (Algoritmo de Dijkstra)

Ciencias de la Computación VIII

Resumen

El enrutamiento Link-State utiliza el algoritmo de Dijkstra para calcular la ruta más corta en redes. Cada router conoce la topología completa al recibir actualizaciones de estado de enlace de otros routers. Con esta información, Dijkstra construye un árbol de rutas más cortas desde el router hacia todos los demás nodos en la red, considerando el costo de los enlaces. A diferencia de los protocolos de enrutamiento por distancia, los routers Link-State tienen una visión global de la red, lo que les permite adaptarse más rápido a los cambios topológicos.
Los RFC's que utilizan el algoritmo o estan realacionados con routing son:
RFC 2328, RFC 5340, RFC 8218, RFC 3626, RFC 1058

Instrucciones de Implementación

No vamos a implementar el protocolo **OSPF,** ni crearemos cliente/servidor UDP en el puerto 89, nos vamos a enfocar en el algoritmo y simular que los Request/Response de los Headers ya hubo recursión dentro de la red y ya los hemos recibido todas las metricas desde el nodo donde iniciamos.

Su implementación debe de ejecutar el algoritmo de enrutamiento Dijkstra, desde un nodo inicial seleccionado.

Debe leer un archivo TXT, el nodo inicial, los nodos, conexiones y costos.

Debe mostrar cada paso hasta llegar al calculo final de todos los Nodos.

En consola debe de mostrar de encabezado N y luego las columnas Step, D(k) y p(k) (de cada Nodo) y por ultimo N' como en los ejemplos

Política de Selección de Empate:
- En la iteración de selección del nodo de costo mínimo, si hay más de uno, se selecciona el siguiente en orden alfabético.
Ejemplo: Si los nodos candidatos son (i, n, t, e, l), se selecciona "e" por ser el primero alfabéticamente, (e, i, l, n, t).

- En el cálculo del costo mínimo de D(v) para elegir p(v), si ambos nodos tienen el mismo costo de ruta, se elige el nuevo nodo, es decir, nodo "w".
Ejemplo (tomado del Tab Ejemplo 2):
D(v) = min(D(v), D(w) + c(w,v)) # Definición
D(h) = min(D(h), D(g) + c(g,h)) # Ejemplo 2
Esto resulta en D(h) = min(6,6) y p(h) = "f" o "g". En este caso, se seleccionará el nuevo nodo, que sería "g", ya que el actual es "f"

Descripción de Funciones

c(w,v): Es el costo desde w a v, donde w y v son vecinos.

D(v): Menor costo desde el nodo de origen hasta el destino v de la iteración actual.

p(v): Nodo anterior (vecino de v) en la ruta de menor costo actual desde el origen hasta v.

N': subconjunto de nodos; v está en N' si la ruta de menor costo desde el origen hasta v es definitivamente conocida.

Algoritmo de Dijkstra

```
Inicialización:
N' = {u}
Para todos los nodos v
        Si v es un vecino de u
                Entonces D(v) = c(u,v)
                         p(v) = u
        de otro modo D(v) = infinito

Repetir:
        Encontrar w que no esté en N' tal que D(w) es un mínimo
        Añadir w a N'
        Actualizar D(v) para cada vecino v de w y no esté en N':
                D(v) = min( D(v), D(w) + c(w,v) )
                p(v) = Depende si v o w es el minimo.
Hasta N' = N
```

Ejecución de Topología Trivial donde todos son Vecinos
Inicianciando desde Nodo (a)

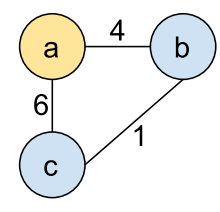
En la primera ejecución nuestro nodo u es el nodo inicial y v son los vecinos que los enlaza una conexión con costo.

N' = {a} N = {a,b,c}

Para este ejemplo, en la primera evaluación se conoce toda la red, pero no se determina la ruta de menor costo. Además, todos son vecinos; no hay ninguno que esté fuera de la primera evaluación, por lo que no se le asigna a algún nodo infinito o un número muy grande

Los calculos de los vecinos serian:

D(v) = c(u,v) **=>** D(b) = c(a,b) **=>** D(b) = 4
D(v) = c(u,v) **=>** D(c) = c(a,c) **=>** D(c) = 6



En el ciclo de Repeticiones va parar en 2 iteraciones:
Encontrar nodo w que no esté en N' tal que D(w) es un mínimo, en este caso, de los 2 calculados, el candidato entre D(b) y D(c), osea {4,6} sus costos respectivos, tomariamos al nodo b por tener 4 de costo.
Añadimos el nodo seleccionado a N' y actualizamos su D(v) de cada vecino de w y que no este en N'.
Los nodos vecinos y que no estan en N' el unico nodo es c en actualizar su costo.

N' = {a, b} N = {a,b,c}

D(v) = min(D(v), D(w) + c(w,v)) **=>** D(c) = min(D(c), D(b) + c(b,c))
=> D(c) = min(6, 4 + 1) **=>** D(c) = min(6, 5) **=>** D(c) = 5

En el siguiente ciclo, al encontrar el nodo w que no esté en N' y que D(w) sea el mínimo, al quedar un único nodo (c) se agrega a N' y, al no tener vecinos, se termina el ciclo.

N'={a,b,c} === N={a,b,c}

D(b) = 4 p(b)=a
D(c) = 5 p(c)=b

Ejemplo del Archivo TXT para la anterior explicación:

```
a
a-b:4
a-c:6
b-c:1
```

La primera linea del documento siempre será el nodo inicial desde donde se calculará la tabla de enrutamiento.
El formato de las siguientes lineas se relacionan 2 nodos por medio de un guion (-) y se coloca dos puntos (:) seguido de su costo

Dentro del Laboratorio se entregan varios TXT ya con las topologias armadas y precalculadas para su evaluación.

Nota: Puede crear otras topologías, además de las de ejemplo o conocidas, para validar que su funcionamiento sea correcto.

Ejemplo de LOG de ejecución del Laboartorio

```
C:\user\CC8>python --version
Python 3.12.2
C:\user\CC8>python LinkState.py nodos.txt
--------+-----------+-----------+----+
| N = a,b,c                         |
--------+-----------+-----------+----+
| Paso  | D(b),p(b) | D(c),p(c) | N' |
--------+-----------+-----------+----+
|   0   |    4,a    |    6,a    | a  |
--------+-----------+-----------+----+
|   1   |    4,a    |    5,b    | b  |
--------+-----------+-----------+----+
|   2   |    4,a    |    5,b    | c  |
--------+-----------+-----------+----+
C:\user\CC8>
```

Entrega

El laboratorio debe ser entregado por medio del GES, con todos los archivos en un ZIP.

La calificación para este laboratorio será Virtual por lo que debe enviar un video explicando lo siguiente: (Tomar en cuenta que el GES tiene un limite de subida de archivos, enviar en un archivo README el link de Youtube)

- Se adjunta la hoja de topologías de la actividad que se realizó de forma presencial en clase, la cual se evaluará por separado. En caso de que no haya asistido a clase, debes presentar sus resultados y procedimientos de los cuatro ejercicios de la hoja de topologías para recibir una calificación en la asignación.

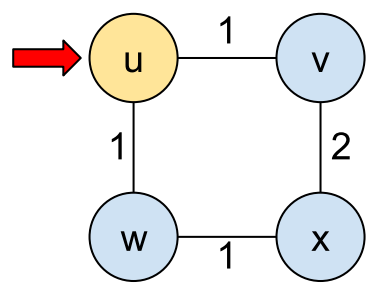
- Explicación de la lógica de programación y los cálculos, así como de cualquier modificación o simplificación en la forma de presentar la información o calcular el algoritmo.

- Mostrar la ejecución de los seis archivos de las topologías conocidas y calculadas, tres ejemplos y tres actividades, para validar que su funcionamiento sea dinámico y correcto.
topo1.txt -> Ejemplo 1
topo2.txt -> Ejemplo 2
topo3.txt -> Ejemplo 3
topo4.txt -> Ejercicio 1 Hoja de Topologias
topo5.txt -> Ejercicio 2 Hoja de Topologias
topo6.txt -> Ejercicio 3 Hoja de Topologias

- Ejecuta la topología topo7.txt
Argumenta y demuestra por qué tu respuesta es correcta.

El laboratorio puede recibir una calificación de cero si:
- No se realiza en Python 3.x.x.
- No compila o no coincide con lo mostrado en el video.
- No cumple con el funcionamiento esperado del algoritmo de Dijkstra.
- La salida en consola no muestra al menos lo mínimo del ejemplo.

EJEMPLO 1



El Primer paso es hacer la inicialización de todos los nodos.

N	u,v,w,x			
Paso	**D(v),p(v)**	**D(w),p(w)**	**D(x),p(x)**	**N'**
0	1,u	1,u	∞	u

Luego ya se comienza a recorrer los nodos actualizando su D(v), p(v) y N'

El nodo con costo minimo en N que no este en N' son (v) y (w), se escoge arbitrariamente, tomemos (v)

Los vecinos de (v) que no estan en N' es: (x)

Se calcula el costo de(x) desde (v) y encontrar el minimo
D(x) = min(D(x), D(v) + c(v,x))
D(x) = min(∞ , 1 + 2)
D(x) = 3
p(x) = v

N	u,v,w,x			
Paso	**D(v),p(v)**	**D(w),p(w)**	**D(x),p(x)**	**N'**
0	1,u	1,u	∞	u
1	1,u	1,u	3,v	v

El nodo con costo minimo en N que no este en N' es (w)
Los vecinos de (w) que no estan en N' es: (x)

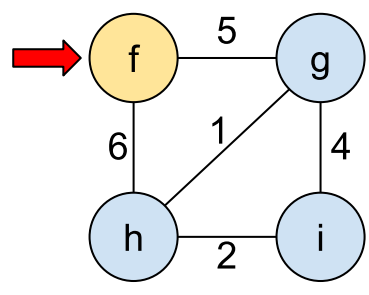
Se calcula el costo de(x) desde (w) y encontrar el minimo
D(x) = min(D(x), D(w) + c(w,x))
D(x) = min(3 , 1 + 1)
D(x) = 2
p(x) = w

N	u,v,w,x			
Paso	**D(v),p(v)**	**D(w),p(w)**	**D(x),p(x)**	**N'**
0	1,u	1,u	∞	u
1	1,u	1,u	3,v	v
2	1,u	1,u	2,w	w

El nodo con costo minimo en N que no este en N' es (x)
Los vecinos de (x) que no estan en N' no hay.
Este punto se termina la ejecución además de que N == N'

N	u,v,w,x			
Paso	**D(v),p(v)**	**D(w),p(w)**	**D(x),p(x)**	**N'**
0	1,u	1,u	∞	u
1	1,u	1,u	3,v	v
2	1,u	1,u	2,w	w
3	1,u	1,u	2,w	x

EJEMPLO 2



El Primer paso es hacer la inicialización de todos los nodos. En este ejemplo el nodo (i) no es vecino de (f) por lo que se colocó D(i)=999 como infinito y p(i)= f, esto para darle un sentido en su implementación

N	f,g,h,i			
Paso	**D(g),p(g)**	**D(h),p(h)**	**D(i),p(i)**	**N'**
0	5,f	6,f	999,f	f

Luego ya se comienza a recorrer los nodos actualizando su D(v), p(v) y N'

En la primera iteración hay un caso interesante, cuando se calcula D(h) = min(D(h), D(g) + c(g,h)) este queda como D(h) = min (6, 6) p(h) = f o g

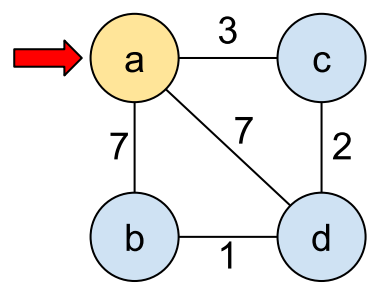
Esto dependerá de la política de actualización. Usualmente, no se cambia, puede se puede optar que se tenga la menor cantidad de saltos o se puede implementar de manera que se tengan en cuenta ambos nodos previos, en dado caso se caíga la conexión y no se tenga que buscar otra ruta o avisar a la red de la actualización de tabla de ruteo inmediatamente. Para nuestro ejemplo, vamos a cambiar al nuevo nodo, siento (f) el previo y pasamos a (g).

N	f,g,h,i			
Paso	**D(g),p(g)**	**D(h),p(h)**	**D(i),p(i)**	**N'**
0	5,f	6,f	999,f	f
1	5,f	6,g	9,g	g

Seguimos con los calculos

N	f,g,h,i			
Paso	**D(g),p(g)**	**D(h),p(h)**	**D(i),p(i)**	**N'**
0	5,f	6,f	999,f	f
1	5,f	6,g	9,g	g
2	5,f	6,g	8,h	h
3	5,f	6,g	8,h	i

EJEMPLO 3



El Primer paso es hacer la inicialización de todos los nodos.

N	a,b,c,d			
Paso	**D(b),p(b)**	**D(c),p(c)**	**D(d),p(d)**	**N'**
0	7,a	3,a	7,a	a

Luego se recorre todos los nodos actualizando su D(v), p(v) y N' como indica el algoritmo

N	a,b,c,d			
Paso	**D(b),p(b)**	**D(c),p(c)**	**D(d),p(d)**	**N'**
0	7,a	3,a	7,a	a
1	7,a	3,a	5,c	c
2	6,d	3,a	5,c	d
3	6,d	3,a	5,c	b