Dibujo en blanco y negro

Descripción generada automáticamente con confianza baja**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**--------------------REDES DE COMPUTADORAS------------------**

**ACTIVIDAD:**

Algoritmos Dikstra y Bellman Ford

**ALUMNO:**

Meza Vargas Brandon David – 2020630288

**GRUPO:**

2CV15

**PROFESOR:**

Gaspar Medina Fabian

**Algoritmo DIjKSTRA**

El algoritmo de Dijkstra nos permite calcular el camino o ruta más corta entre un nodo (nosotros elegimos el nodo) y todos los demás nodos en el grafo. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959.

El algoritmo es el siguiente:

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. Para el nodo actual, calculamos la distancia tentativa desde dicho nodo a sus vecinos con la siguiente formula: dt(vi) = Da + d(a.vi. Es decir, la distancia tentativa del nodo vi es la distancia que actualmente tiene le nodo en el vector D más la distancia desde dicho nodo el nodo a (el actual) al nodo vi. Si la distancia tentativa es menor que la distancia almacenada en el vector, actualizamos el vector con esta distancia tentativa.
5. Marcamos como completo el nodo a
6. Tomamos como próximo nodo actual al de menor valor en D (puede hacerse almacenando valores en una cola de prioridad) y se vuelve al paso 3 mientras existan nodos no marcados.

Veamos lo anterior con un ejemplo.

Se tiene la siguiente red, y el origen es el nodo 1.

Diagrama

Descripción generada automáticamente

Se procede con las iteraciones, en la iteración 0 asignamos la etiqueta para el nodo origen: iteración 0.

En este paso asignamos una etiqueta permanente al nodo origen.

***Etiqueta nodo 1 = [valor acumulado, procedencia] iteración***

***Etiqueta nodo 1 = [0,-] 0***

**Iteración 1:** En esta iteración evaluamos a cuáles nodos se puede llegar desde el nodo 1. En este caso se puede llegar a los nodos 2 y 3. De manera que debemos asignar las etiquetas para cada nodo.

***Etiqueta nodo 2 = [0+100,1] 1***

0 es el valor acumulado en la etiqueta del nodo de procedencia, en este caso el valor acumulado para el nodo 1, 100 es el valor del arco que une el nodo de procedencia y el nodo destino, 1 es la iteración.

***Etiqueta nodo 3 = [0+30,1] 1***

La etiqueta del nodo 3 será permanente ya que 30 es la mínima distancia posible.

Tabla

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

**Iteración 2:** Se evalúan las posibles salidas desde el nodo 3, es decir, los nodos 4 y 5, así se asignas las etiquetas.

***Etiqueta nodo 4 = [30+10,3] 2***

***Etiqueta nodo 5 = [30+60, 3] 2***

Así tenemos lo siguiente:

Tabla

Descripción generada automáticamente

Diagrama, Esquemático

Descripción generada automáticamente

**Iteración 3:** aquí evaluamos las posibles salidas desde el nodo 4.

***Etiqueta nodo 2 = [40+15,4] 3***

***Etiqueta nodo 5 = [90+4] 3***

Quedando lo siguiente:

Diagrama

Descripción generada automáticamente

**Iteración 4:** En este paso evaluamos las posibles salidas desde el nodo 2 y el 5. Sin embargo, el nodo 2 solo tiene un posible destino, el 2, el cual ya tiene una etiqueta permanente, de manera que no puede ser reetiquetado. Ahora, evaluando el nodo 5 es un nodo que no tiene destinos. Así entonces, su etiqueta temporal para a ser permanente, en este caso cuenta con 2 etiquetas que tienen el mismo valor, es decir, alternativas óptimas. Así se concluye el algoritmo de Dijkstra.

Diagrama

Descripción generada automáticamente

**Algoritmo de Bellman-Ford**

Este algoritmo determina la tura más corta desde un nodo origen hacia los demás nodos para ello es requerido como entrada un grafo cuyas aristas posean pesos. La diferencia de este algoritmo con otros algoritmos es que los pesos pueden tener valores negativos ya que este algoritmo permite detectar la existencia de un ciclo negativo.

Este algoritmo parte de un vértice origen que será ingresado, a diferencia de Dijkstra que utiliza una técnica voraz para seleccionar vértices de menor peso y actualizar sus distancias mediante el paso de relajación. Bellman-Ford simplemente relaja todas las aristas y lo hace |V|-1 veces, siendo |V| el número de vértices del grafo.

Para la detección de ciclos negativos realizamos el paso de relajación una vez más y si se obtuvieron mejores resultados es porque existe un ciclo negativo, para verificar porque tenemos un ciclo podemos seguir relajando las veces que queramos y seguiremos obteniendo mejores resultados.

**Funcionamiento en redes**

* Cada ruteador mantiene una tabla (un vector) que almacena las mejores distancias conocidas a cada destino y las líneas a usar para cada destino. Se actualizan las tablas intercambiando información con los vecinos
* La tabla de un ruteador almacena una entrada para cada uno de los ruteadores en la subred (los ruteadores son los índices). Las entradas almacenan la línea preferida de salida y una estimación del tiempo o la distancia al destino. Se pueden usar métricas distintas (saltos, retrasos, etc.)
* Cada ruteador tiene que medir las distancias a sus vecinos. Por ejemplo, si la métrica es el retraso, el ruteador la puede medir usando paquetes de eco
* Cada T msegs los ruteadores intercambian sus tablas con sus vecinos. Un ruteador usa las tablas de sus vecinos y sus mediciones de las distancias a sus vecinos para calcular una nueva tabla.

Gráfico

Descripción generada automáticamente

**Bibliografía**

1. uma, (2018). “Algoritmo de Bellman Ford”. Obtenido de: <https://neo.lcc.uma.es/evirtual/cdd/tutorial/red/bellman.html>
2. CodinGame, (2019). “El algoritmo de Dijkstra”. Obtenido de: <https://www.codingame.com/playgrounds/7656/los-caminos-mas-cortos-con-el-algoritmo-de-dijkstra/el-algoritmo-de-dijkstra>
3. López, S. (2019). “Algoritmo de Dijkstra”. Obtenido de: <https://www.ingenieriaindustrialonline.com/investigacion-de-operaciones/algoritmo-de-dijkstra/>