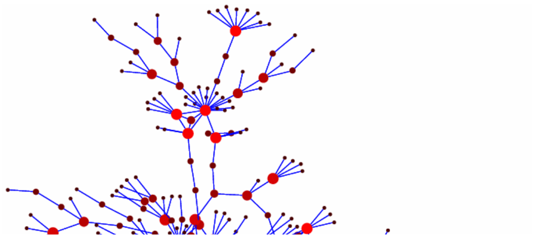


Práctica 4: Árboles generadores de mínimo coste, el algoritmo de Prim

Profesor responsable: Sergio Alonso.
Dificultad: media.
Tutorización: semana del 17 de abril   Corrección: semana del 24 de abril

1. Objetivo

El objetivo de esta práctica es la implementación de un algoritmo capaz de construir el árbol generador de mínimo coste, MST, de un grafo no dirigido con costes o pesos en sus aristas. Para resolver el problema MST usaremos el algoritmo de Prim (1957) que lo construye, por lo que debemos inducir un orden entre las aristas según sus costes, para examinarlas en ese orden y decidir si entran o no en la solución, siempre evitando los ciclos.

2. El Algoritmo de Prim (1957)

El algoritmo de Prim, selecciona en cada fase un nodo nuevo a entrar en la solución, y siempre el que suponga un menor coste para que se incorpore.

Un esquema del mismo se expresa en el siguiente pseudocódigo:

```
T = ∅
M = {1}
Mientras en T no haya n-1 aristas hacer
  Sea e la arista de menor coste con un extremo en M, y el otro, j, en V-M
  T = T U {e}
  M = M U {j}
```

En el pseudocódigo, en T dispondremos las aristas que formarán parte de la solución, añadiendo una en cada uno de los n-1 pasos; el conjunto M almacena los nodos que ya han sido conectados por las aristas en T, e, inicialmente, puede ser cualquier nodo, por ejemplo, el 1; la condición se establece sobre las aristas que conectan un nodo de M con otro fuera de M, y de ellas, escogemos la que menos coste nos supone; finalmente, entra en ese paso la arista señalada y el nodo fuera de M que tal arista conecta.

Una vez entendida la dinámica, pasemos a expresar el algoritmo en un pseudocódigo un poco más elaborado y cercano a una implementación. En este caso, usamos dos vectores de nodos para almacenar la siguiente información:

- etiqueta de coste, `coste[nodo]`, almacena el coste que incorporamos al peso del árbol generador si introducimos `nodo` en la solución.
- el predecesor, `pred[nodo]`, el nodo en M desde el que conectamos `nodo`.

Así, en cada caso, un nodo `i` fuera de M, si lo introdujera en la solución parcial que estoy construyendo, me supondría incrementar `coste[i]` el peso del árbol parcial que voy armando, y sería con la arista `(i, pred[i])`.

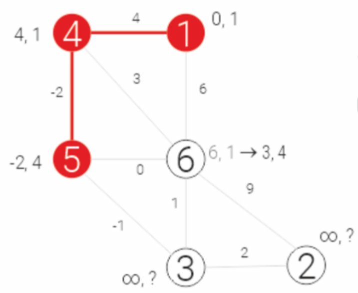
Por tanto, en cada paso, selecciono el nodo que no esté en la solución (fuera de M), que me aporte menos coste, y en el conjunto M voy metiendo los nodos que voy conectando.

Así, tendríamos de forma más detallada el algoritmo de Prim:

```
T = ∅
Para todo nodo i de V hacer coste[i] = ∞
M = {1}
coste[1] = 0
pred[1] = 1
Mientras en T no haya n-1 aristas hacer
  sea u el último nodo que entró en M //u va a mejorar costes
  para todo j adyacente a u en V-M hacer
      si coste[j] > w(u, j) entonces
            coste[j] = w(u,j) //Esta arista es menos costosa
            pred[j] = u //cambio el nodo de conexión
  sea u = nodo con menor coste en V-M
  M = M U {u}
  T = T U {(u, pred[u])]
```

Por tanto, inicialmente, todos los nodos, salvo el inicial en M que tiene coste 0, tienen coste infinito. El nodo inicial está en M, esto es, viene incorporado a la solución sin coste. A partir de ese nodo inicial, se intenta conectar el resto de nodos, y sólo el que me signifique menos coste, entrará en la solución, en M, y la arista en el árbol que estoy conformando.

Para ver una detallada traza de Prim, consulta [aquí](#) la presentación dinámica que puedes consultar en el campus virtual, y en la que, paso a paso se construye el árbol generador de mínimo coste con el algoritmo de Prim.




```
Algoritmo de Prim
Para todo nodo i de V hacer coste[i] = ∞;

T= ∅; M={1}; coste[1]=0; pred[1]=1

Mientras en T no haya n-1 aristas hacer
        sea u el último nodo que entró en M
        para todo j adyacente a u en V-M hacer
           si coste[j] es peor que w(u, j) entonces
                  coste[j]=w(u,j)
                  pred[j] = u
        sea u = arg min coste [j] para todo j en V-M
        M = M ∪ {u}
        T = T ∪ {(u, pred[u])}
```

Presentamos el grafo problema, con la inicialización: Prim desde el nodo 1.

	1	2	3	4	5	6
coste	0	∞	∞	4	-2	3
pred	1	?	?	1	4	4

Revisamos la adyacencia de u=1, {6, 4};
el nodo 6, nunca alcanzado, actualizamos coste[6]=6 y pred[6]=1
el nodo 4, nunca alcanzado, actualizamos coste[4]=4 y pred[4]=1
el nodo fuera de M con menor coste es u=4: actualizamos

Revisamos la adyacencia de u=4, {1, 6, 5};
el nodo 1 está en M
el nodo 6, comparamos coste desde 4, **actualizamos** coste[6]=3 y pred[6]=4
el nodo 5, nunca alcanzado, actualizamos coste[5]=-2 y pred[5]=4
el nodo fuera de M con menor coste es u=5: actualizamos

$$\Gamma_1 = \{6,4\}$$
$$\Gamma_2 = \{6,3\}$$
$$\Gamma_3 = \{5,6,2\}$$
$$\Gamma_4 = \{1,6,5\}$$
$$\Gamma_5 = \{4,6,3\}$$
$$\Gamma_6 = \{1,4,5,3,2\}$$

Evaluación

Para superar esta práctica en el laboratorio debe implementarse el algoritmo de Prim para la construcción del MST, que debe funcionar correctamente.

Para poder acceder al apto+ se evaluará la defensa de la práctica por parte del alumno o alumna, la respuestas a las preguntas durante la corrección, el código de la práctica y se podrá plantear una modificación para que sea resuelta durante la corrección para optar al apto+.

Asimismo, al finalizar la evaluación de la práctica, si es apto -, apto o apto +, el alumno responderá a un corto cuestionario para evaluar sus conocimientos sobre la práctica y los contenidos teóricos que la sustentan.