QUINTA PRÁCTICA

PLANIFICACIÓN DE CAMINOS I (DIJKSTRA)

MIGUEL IAN GARCÍA POZO

GIERM

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE MÁLAGA

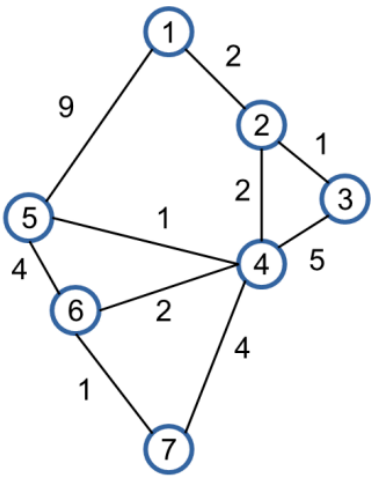
Índice

Algoritmo de Dijkstra

- Desarrollo

En esta práctica, vamos a desarrollar en Matlab el algoritmo de planificación de caminos Dijkstra. Para este apartado vamos a utilizar el siguiente grafo:

```
G = [0 2 0 0 9 0 0;
     2 0 1 2 0 0 0;
     0 1 0 5 0 0 0;
     0 2 5 0 1 2 4;
     9 0 0 1 0 4 0;
     0 0 0 2 4 0 1;
     0 0 0 4 0 1 0;
     ];
```



Antes de empezar con el código, debemos explicar en qué consiste el algoritmo.

Este algoritmo recorre cada uno de los nodos del grafo desde un nodo inicial hasta un nodo final y trata de descubrir cuál es el camino a seguir con menor coste, es decir, el óptimo. Para ello, el algoritmo va recorriendo los nodos calculando el coste del camino desde el nodo en el que está hasta el nodo inicial. La fórmula que sigue este algoritmo viene a ser:

$$f(n) = f(nodo\ padre) + arco$$

Donde n es el nodo que estamos comprobando y $arco$ es el número encima de las líneas que representa el coste de ir de un nodo a otro.

La metodología que sigue el algoritmo consiste en calcular el coste entre el nodo en el que está y todos los nodos con los que esté conectado, después, elimina ese nodo, pasa al siguiente nodo cuyo coste sea el menor de todos y vuelve a empezar el proceso. Estas iteraciones terminan cuando el nodo al que le toca ir al algoritmo es el nodo destino.

Ahora que entendemos cómo funciona el algoritmo, vamos a ver el código que lo lleva a cabo.

```
function [coste,ruta] = dijkstra(g,s,x)


    [m,n] = size(g); % Obtenemos el tamaño del grafo de entrada

    caminos = Inf(1,m); % Definimos un vector de infinitos
    % Creamos una matriz con: columna de nodos, columna de costes y columna
    % de nodo anterior
    caminos = [1:m;caminos;zeros(1,m)]';
    % Cambiamos la primera fila por la del nodo inicio y cambiamos su coste
    % a cero
    caminos([1 s],:) = caminos([s 1],:);
    caminos(1,2) = 0;

    % Creamos la matriz donde vamos a guardar los nodos que ya hemos
    % revisado y la matriz donde pondremos la ruta solución
    revisado = zeros(m,3);
    solucion = zeros(m,3);

    % i es el nodo que estamos revisando y p es la posición en la matriz de
    % revisados donde vamos a poner el nodo que acabemos de ver
    i = s;
    p = 1;
```

```
while i ~= x
    % Creamos un bucle para recorrer las columnas de la matriz del
    % grafo
    for j = 1:length(g(1,:))
        % Este bucle nos devolverá la fila donde está ubicado el nodo
        % que buscamos en la matriz "caminos"
        for l = 1:length(caminos(:,1))
           if (caminos(l,1) == j)
                nodo = l;
           end
        end
        % Comprobamos si el elemento que estamos viendo es distinto de
        % cero y si el coste hasta el nodo es menor que el coste
        % anterior y guardamos la información en la matriz "caminos"
        if (g(i,j) ~= 0 )
            if (g(i,j)+caminos(1,2) < caminos(nodo,2))
                caminos(nodo,2) = g(i,j) + caminos(1,2);
                caminos(nodo,3) = i;
            end
        end
    end
    % Guardamos en "revisado" el nodo, volvemos a cero la columna del nodo en la matriz del grafo
    % eliminamos la primera fila para no volver a comprobarla y ordenamos la matriz
    g(:,caminos(1,1)) = 0;
    revisado(p,:) = caminos(1,:);
    p = p+1;
    caminos(1,:) = [];
    caminos = sortrows(caminos,2);
    i = caminos(1,1);
end
```

```
        % Guardamos la fila del nodo destino
        revisado(m,:) = caminos(1,:);
        % Guardamos el nodo destino en la solución y creamos la variable con la
        % que vamos a ver el nodo del que hemos llegado
        solucion(1,:) = revisado(m,:);
        ant = m;
        % Este bucle se va a encargar de recorrer la matriz "revisado" y
        % encontrar cuál es la ruta a seguir
        for j = 2:length(revisado)
            for l = 1:length(revisado(:,1))
               if (revisado(l,1) == revisado(ant,3) && revisado(ant,3)~= 0)
                    ant = l;
                    solucion(j,:) = revisado(l,:);
               end
            end
        end

        % Elimino las filas de la matriz que siguen valiendo 0
        solucion(sum(abs(solucion),2)==0,:)=[ ];
        solucion(:,sum(abs(solucion),1)==0)=[ ];

        % Defino el coste y la ruta, siendo esta la columna de los nodos de la
        % solución invertida, para poder ver el recorrido desde la salida a la
        % llegada
        coste = solucion(1,2);
        ruta = fliplr(solucion(:,1)');
    end
```

- Simulación

Para simular este código, primero debemos cargar los grafos:

```
load('grafos.mat')
```

Una vez tenemos los grafos, ya solo queda ejecutar el comando que nos dice el enunciado para ver cuál es el camino que encuentra:

```
>> [coste,ruta] = dijkstra(G,1,7)

coste =

     7


ruta =

     1     2     4     6     7
```

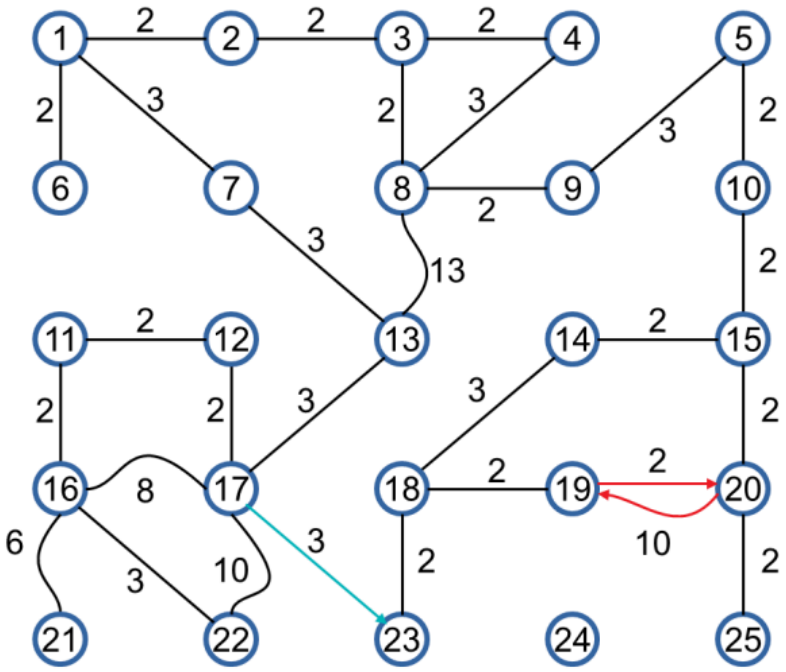
Comprobamos en la imagen que obtenemos el mismo valor que el enunciado, además, si vemos el grafo, nos damos cuenta de que ese es el camino óptimo. Vamos a probar con otros nodos de salida y destino.

```
>> [coste,ruta] = dijkstra(G,5,3)

coste =

     4


ruta =

     5     4     2     3
```

Con estos dos ejemplos, podemos ver con claridad que el código lleva a cabo con precisión el algoritmo de Dijkstra.

Comprobación del Algoritmo

Ahora queremos probar nuestro algoritmo en un mapa topológico en el que hay caminos solo en un sentido. El grafo con el que vamos a probar Dijkstra es el siguiente:



En este ejercicio, nos piden comprobar los resultados que se obtienen con ciertos nodos de salida y de destino. Veamos cómo responde el algoritmo:

1 - 19

coste = 16, ruta = [1 7 13 17 23 18 19]

```
>> [coste,ruta] = dijkstra(J,1,19)

coste =

    16


ruta =

     1     7    13    17    23    18    19
```

19 - 1

coste = 19, ruta = [19 20 15 10 5 9 8 3 2 1]

```
>> [coste,ruta] = dijkstra(J,19,1)

coste =

    19


ruta =

    19    20    15    10     5     9     8     3     2     1
```

25 – 19

coste = 11, ruta = [25 20 15 14 18 19]

```
>> [coste,ruta] = dijkstra(J,25,19)

coste =

    11


ruta =

    25    20    15    14    18    19
```

19 – 25

coste = 4, ruta = [19 20 25]

```
>> [coste,ruta] = dijkstra(J,19,25)

coste =

     4


ruta =

    19    20    25
```

1 - 231

coste = 12, ruta = [1 7 13 17 23]

```
>> [coste,ruta] = dijkstra(J,1,23)

coste =

    12


ruta =

     1     7    13    17    23
```

23 - 1

coste = 22, ruta = [23 18 14 15 10 5 9 8 3 2 1]

```
>> [coste,ruta] = dijkstra(J,23,1)

coste =

    22


ruta =

    23    18    14    15    10     5     9     8     3     2     1
```

1 - 24

coste = Inf, ruta = []

```
>> [coste,ruta] = dijkstra(J,1,24)

coste =

   Inf


ruta =

    24
```

23 - 22

coste = 40, ruta = [23 18 14 15 10 5 9 8 3 2 1 7 13 17 12 11 16 22]

```
>> [coste,ruta] = dijkstra(J,23,22)

coste =

    40


ruta =

    23    18    14    15    10     5     9     8     3     2     1     7    13    17    12    11    16    22
```

Tras haber realizado todos los casos y habiendo obtenido el resultado en todos los casos, ya sí que podemos declarar que hemos implementado con éxito el algoritmo de planificación de caminos de Dijkstra.