En este ejercicio se plantea realizar una planificación de caminos para un robot móvil, siendo necesario planificar el camino a realizar entre dos puntos o localizaciones muy distantes en el espacio. Para resolver este problema de planificación, dado que la ubicación inicial y final están separadas por largas distancias, se utilizan mapas topológicos y algoritmos de búsqueda en grafos.

En esta práctica se plantea resolver el problema de encontrar un camino o ruta mediante el algoritmo de Dijkstra. Este algoritmo es un método de búsqueda desinformada en grafos, únicamente considerando el coste de los caminos que unen los nodos del grafo. El método de Dijkstra proporciona la ruta de menor coste entre un origen y un destino, siendo, por tanto, el camino óptimo.

Se pide:

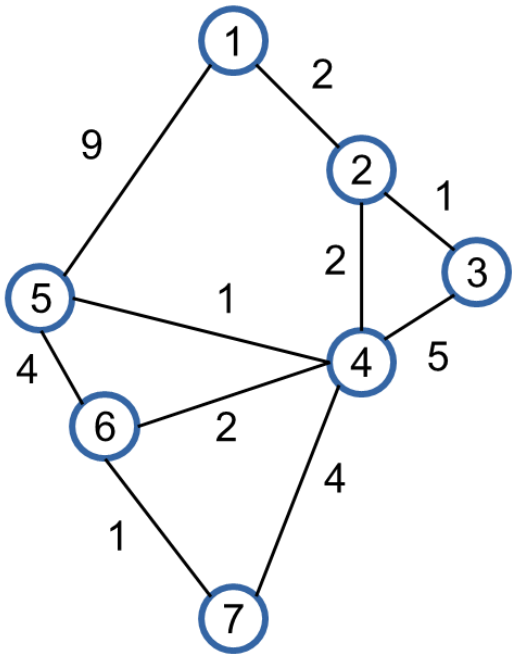
1. Implementar el algoritmo de Dijkstra como una función de Matlab que devuelva el coste y la ruta óptima a partir de un origen y un destino pasados como parámetros, además del mapa topológico o grafo, que se le pasará a la función como una matriz NxN, que almacena el coste de llegar del nodo n1, como fila, al nodo n2, como columna. La función se debe implementar de forma que la llamada:

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

devuelva el coste de llegar desde el nodo origen al nodo destino, y un vector con la lista de nodos que componen la ruta (incluidos los nodos inicial y final).

Por ejemplo, dado el mapa topológico y la matriz de costes correspondiente que se muestran a continuación:

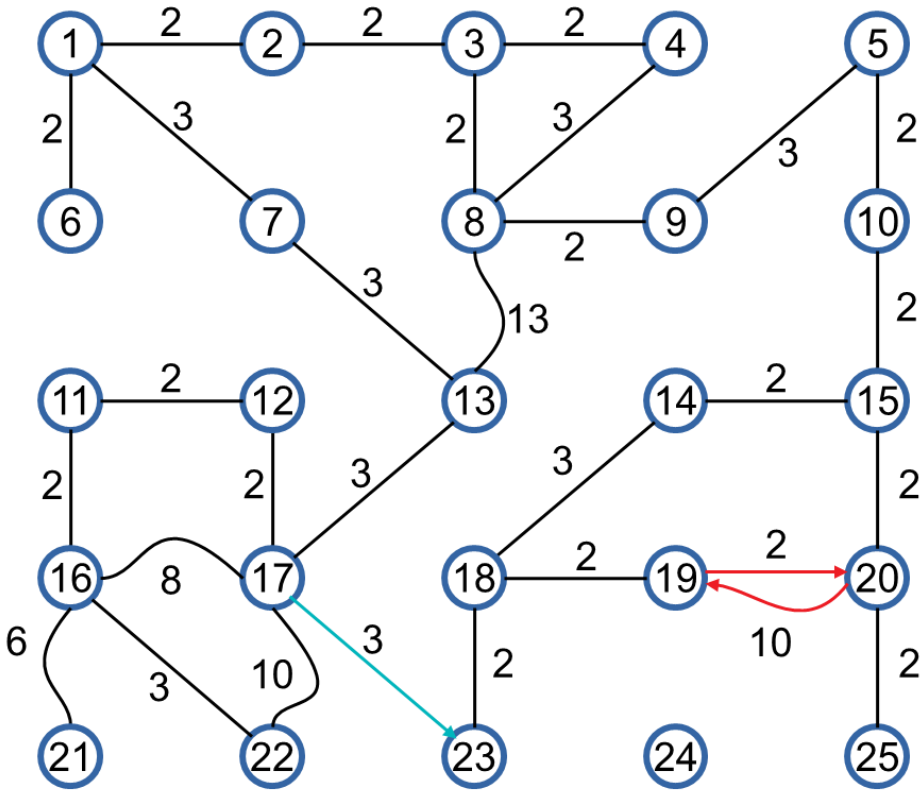
```
G = [0 2 0 0 9 0 0;
     2 0 1 2 0 0 0;
     0 1 0 5 0 0 0;
     0 2 5 0 1 2 4;
     9 0 0 1 0 4 0;
     0 0 0 2 4 0 1;
     0 0 0 4 0 1 0;
     ];
```



La función debe dar como resultado: **coste = 7**

ruta = [1 2 4 6 7]

2. Una vez realizada la función y comprobado que funciona adecuadamente, comprobar el algoritmo implementado con el siguiente mapa topológico:



La matriz de coste del grafo anterior se proporciona en el archivo ***'grafos.mat'***, y se muestra a continuación:

```
J= [
    0   2   0   0   0   2   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0;
    2   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0;
    0   2   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0;
    0   0   2   0   0   0   0   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0;
    0   0   0   0   0   0   0   0   3   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0;
    2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0;
    3   0   0   0   0   0   0   0   0   0   0   0   3   0   0   0   0   0   0   0   0   0   0   0   0;
    0   0   2   3   0   0   0   0   2   0   0   0  13   0   0   0   0   0   0   0   0   0   0   0   0;
    0   0   0   0   3   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0;
    0   0   0   0   2   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0;
    0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   2   0   0   0   0   0   0   0   0   0;
    0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   2   0   0   0   0   0   0   0   0;
    0   0   0   0   0   0   3  13   0   0   0   0   0   0   0   0   3   0   0   0   0   0   0   0   0;
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   3   0   0   0   0   0   0   0;
    0   0   0   0   0   0   0   0   0   2   0   0   0   2   0   0   0   0   2   0   0   0   0   0   0;
    0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   8   0   0   0   6   3   0   0   0;
    0   0   0   0   0   0   0   0   0   0   0   2   3   0   0   8   0   0   0   0   0  10   3   0   0;
    0   0   0   0   0   0   0   0   0   0   0   0   0   3   0   0   0   0   2   0   0   0   2   0   0;
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   2   0   0   0   0   0;
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0  10   0   0   0   0   0   2;
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   6   0   0   0   0   0   0   0   0   0;
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   3  10   0   0   0   0   0   0   0   0;
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0;
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0;
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0]
```

Observar la asimetría de la matriz en los costes de los arcos dirigidos.

Comprobar que los resultados obtenidos con los siguientes nodos iniciales y finales son:

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]

25 – 19

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

19 – 25

coste = 4, ruta = [19 20 25]

1 - 231

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

23 - 1

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

1 - 24

coste = Inf, ruta = []

23 - 22

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

Algunos conceptos útiles de MATLAB para la implementación:

```
% AÑADIR UNA FILA A UNA MATRIZ
>>a = [1 2 3 ; 4 5 6; 7 8 9];
>>b=[5 5 5];
>>c=[a;b]  % añade una fila

c=   1     2     3
     4     5     6
     7     8     9
     5     5     5

% ORDENAR LAS FILAS DE UNA MATRIZ POR EL VALOR DE UNA COLUMNA
>>a = [5 2 3 ; 4 5 6; 7 8 9];
>>columna=1;
>>b = sortrows(a,columna)

b=   4     5     6
     5     2     3
     7     8     9

% OBTENER LA PRIMERA FILA DE UNA MATRIZ
>>a = [1 2 3 ; 4 5 6; 7 8 9];
>>b=a(1,:)

b = 1 2 3

% BORRAR LA PRIMERA FILA DE UNA MATRIZ
>>a = [5 2 3 ; 4 5 6; 7 8 9];
```

```
>>a = a([2:end],:)

a=   4     5     6
     7     8     9

% O bien:
>>a(1,:)=[]
```

% CREAR UN VECTOR DE INFINITOS

```
>>n=4;
>>a = Inf (n,1)

a = Inf
    Inf
    Inf
    Inf
```

% CREAR UN VECTOR DE CEROS

```
>>n=4;
>>a = zeros(n,1)

a = 0
    0
    0
    0
```