Lab 2 - Task 2

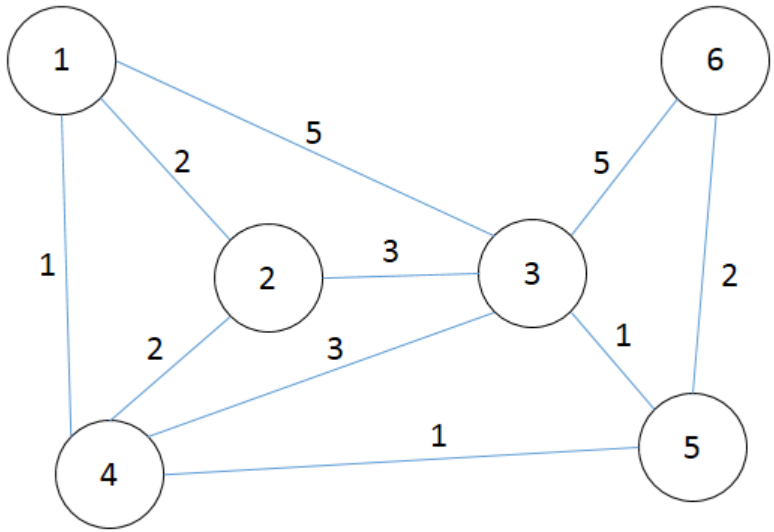
In Task 1, we used routing tables stored at each node to enable packets to get from their source nodes to their destination nodes in the network by the process of forwarding at each node. Note that forwarding is a purely local process. Each node decides what to do with an incoming packet based on information stored in its own routing table. In Task 1, we assumed that the required routing tables had been pre-computed and stored at each node. In a real network, the required routing tables are unknown and possibly time varying as the state of the network changes. Thus, these routing tables need to be computed and dynamically updated by each node.

In the following two tasks, we will see how the nodes in the network can iteratively compute their routing tables by gathering information from their neighbors and integrating this information to update their routing tables using the distance vector algorithm. Specifically, in this task, we will first learn we can use a matrix to represent the links between nodes in the network and their costs, and to use this information to initialize the routing table. Then, in Task 3, we will learn how nodes update their routing tables iteratively.

INSTRUCTIONS

In this exercise, your task is to initialize the routing table of each node based on knowledge about the links leaving the node and their costs. This information about the nodes neighbors might be obtained through the HELLO protocol discussed in lecture.

Consider the same network as in Task 1 as shown below.



As shown in the lab demo, the connections between the nodes in this network can be represented by an **n_nodes**-by-**n_nodes** symmetric

matrix, where **n_nodes** is the number of nodes in the network (6 in this example). For example, the matrix representation of the network above is given by

```
graph = [  0   2   5   1  inf inf;
           2   0   3   2  inf inf;
           5   3   0   3   1   5 ;
           1   2   3   0   1  inf;
          inf inf  1   1   0   2 ;
          inf inf  5  inf  2   0 ];
```

The element **graph(i,j)** contains the cost of the link between nodes **i** and **j**. For example, the element **graph(1,2)**, which has the value 2, indicates that node 1 is directly connected to node 2 by a link and that the cost of taking that link is 2. When two nodes are not directly connected, for example nodes **1** and **5**, the corresponding element in the matrix is set to infinity (**inf**). Finally, the cost of connection between a node and itself is zero, i.e. **graph(i,i)=0** for all **i**. Note that the **i**th row of **graph** contains information about the neighbors of node **i** and the costs of the links from node **i** to those neighbors.

We will initialize the routing table at each node using the list of neighbors at the node and the costs of the links between the node and its neighbours. In other words, initially, each node only knows that it can reach its immediate neighbors in the network, and the costs associated with reaching them. In a real network, each node can obtain this information about its neighbors can be obatained by broadcasting (e.g. the HELLO protocol). In this task, we will extract this information from the matrix **graph**, which we assume to be known.

Using the information stored in the matrix **graph**, your code should define three variables:

neighbors: This is a cell array containing **n_nodes** cells, where the cell **neighbors{i}** contains a column vector whose elements are the indices of the nodes that are neighbors of node **i**.

costs: This is a cell array containing **n_nodes** cells, where the cell **costs{i}** contains a column vector whose elements are the costs associated with each link from node **i** to the neighbor in the corresponding entry in **neighbors{i}**. Thus **costs{i}** and **neighbors{i}** have the same dimension.

RT: This is a cell array containing **n_nodes** cells, where the cell **RT{i}** contains an initial estimate of the routintg table for node **i**, which is obtained from purely local information:

1. The row **RT{i}(i,:) = [0 0]**, indicating that packets destined for node **i** and arriving at node **i** do not need to be forwarded.
2. Packets destined for a node **j**, where **j** is a neighbor of **i**, are forwarded along the link leading directly to **j**, and the associated cost of getting to **j** from **i** is the cost of the link from **i** to **j**.
3. For packets destined for a node **j**, where **j** is neither **i** nor a neighbor of **i**, the row in the routing table ls **RT{i}(j,:) = [0 inf]**, indicating that the packet does not need to be forwarded (the packet is lost so it is discarded), and that the cost of getting to node **j** from **i** is infinite (it cannot get to **j**).

For example, by looking at the first row of the matrix, we note that node 1 is directly connected with nodes 2, 3, and 4 with cost 2, 5, and 1

respectively. Thus, we should have

```
neighbors{1} = [   2      costs{1} = [ 2        RT{1} = [ 0  0
                   3                     5                2  2
                   4 ];                  1 ];             3  5
                                                          4  1
                                                          0 inf
                                                          0 inf ];
```

Your task is to initialize the routing table for all nodes in the network. For that purpose, there are three steps you need to complete for each node **i**.

Step 1: Identify the neighbouring nodes for node i, which are nodes with non-zero and less than infinity link-cost from node i, and store them in **neighbors{i}**. You may find the function "**find**" useful for this step.

Step 2: Extract the corresponding link costs and store them in the variable **costs{i}**.

Step 3: Initialize the routing table for node **i** using the information in **neighbors{i}** and **costs{i}**.

Please revise the code between the lines

```
% % % % Revise the following code   % % % %
```

and

```
% % % % Do not change the code below % % % %
```

Do not change other parts of the code.

Your Solution

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
38          %disp(neighbors{n});
39          %disp(RT{n});
40          %disp(rowsColsOfGraph{n});
41               
42
43      % % % % Do not change the code below % % % %
44  end
```

Assessment Tests: Passed

- ✔ Is the problem unchanged?
- ✔ Are the neighbors correct?
- ✔ Are the costs correct?
- ✔ Are the routing tables correct?

Output

```
Code ran without output.
```