=======================================================

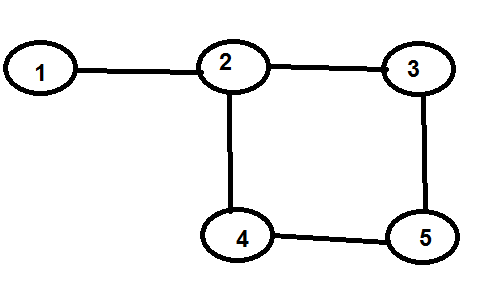
[1] Detection of the critical node based on Global network data

[2] Detection of the critical node based on K-hop neighbors data

=======================================================

[1] Detection of the critical node based on Global network data

To understand the algorithm we will use the next network as a sample



\* Create table (Two-dimension array) for all connections

|  |  |  |
| --- | --- | --- |
| Row ID | [1] First Node | [2] Second Node |
| 1 | 1 | 2 |
| 2 | 2 | 1 |
| 3 | 2 | 3 |
| 4 | 2 | 4 |
| 5 | 3 | 2 |
| 6 | 3 | 5 |
| 7 | 4 | 2 |
| 8 | 4 | 5 |
| 9 | 5 | 3 |
| 10 | 5 | 4 |

Note that the Row ID column is added just to explain the table (During programming we don’t need this column because we already access an array item using an Index)

And the titles (First Node) & (Second Node) is just added to explain the array items meaning

The previous table contains a list of connections based on knowing the 1-hop neighbors for each node.

We have redundancy in this table for example the Row number (1) Shows a connection between node 1 and node 2 and the Row number (2) Shows a connection between node 2 and node 1 which gives the same meaning as the Row number (1)

In practical work this redundancy is avoid and the array looks like this

|  |  |
| --- | --- |
| 1 | 2 |
| 2 | 3 |
| 2 | 4 |
| 3 | 5 |
| 4 | 5 |

Connections Array

* Count the nodes in the network and store this in a variable (nNodesCount)
* For each node in the network
  + Create empty array called (ActiveNodes)
  + Set logical variable called (lContinue) to True
  + While (lContinue == True)
    - Set lContinue = False
    - For each connection in the connections Array
      * Avoid the connection if it contains the node that we check by moving to the next iteration in this for loop and passing the next instructions
      * If the size of the Array (ActiveNodes) = 0 then add the two nodes inside the active connection to the array (ActiveNodes) then we set the variable lContinue to True
        + Else

For each item in the array ActiveNodes

If the current item exist in the active connection we can use this connection by adding it’s items to the ActiveNodes without making duplications (we add the other item in the connection if it’s not already added) then we set lContinue to True

End IF

End For

* + - * End IF
    - End For
  + End While
  + If the count of the items in the array ActiveNodes < nNodesCount – 1 and the number of connections to the active node is greater than 1 then the active node is critical node
    - Else
      * (the active node is not critical
  + End IF
* End For

Now let us test our algorithm

* **Count the nodes in the network and store this in a variable (nNodesCount)**
  + In our example nNodesCount = 5
* **For each node in the network** 
  + In the first iteration the ActiveNode = Node Number (1) then (2) then (3) then (4) then (5)
  + Let us look at the first iteration where the ActiveNode = 1

* + **Create empty array called (ActiveNodes)**

|  |
| --- |
|  |

ActiveNodes Array (Empty = No Nodes in this array)

* + **Set logical variable called (lContinue) to True**

Now we have a variable called lContinue and it’s value is True

* + **While (lContinue == True)**

The loop condition is satisfied and the instructions inside this loop will be executed

* + **Set lContinue = False**

Change the value of the variable lContinue from True to False

This mean that at this state the while loop condition is not satisfied and the loop instructions will not executed again unless the variable value is changed again to True

* + **For each connection in the connections Array**

We will go through the connections array connection by connection

In our example we have 5 connnections and the first connection is the connection between node 1 and node 2

* + **Avoid the connection if it contains the node that we check by moving to the next iteration in this for loop and passing the next instructions**

The ActiveNode = 1 and the Active connection is between node 1 and node 2

So we will avoid the connection and we will move to the next connection

The second connection will become the active connection

The active connection is between node 2 and node 3

Now this connection doesn’t contain the ActiveNode which = 1

So the next instructions will be executed

* + **If the size of the Array (ActiveNodes) = 0 then add the two nodes inside the active connection to the array (ActiveNodes) then we set the variable lContinue to True**

The condition is satisfied because we don’t have items/nodes in the array ActiveNodes so we will add the two nodes from the active connection to the ActiveNodes array

The active connection is between node 2 and node 3 so we will add 2 and 3 to the ActiveNodes array

|  |
| --- |
| 2 |
| 3 |

ActiveNodes Array

Then we will set the variable lContinue to True

And this means that the while loop instructions will be executed again for another time

We still in the connections loop and we will move to the third item in the connections array

The third connection will become the active connection

The active connection now is the connection between node 2 and node 4

* + **Avoid the connection if it contains the node that we check by moving to the next iteration in this for loop and passing the next instructions**

The active node is node number 1 and the active connection is the connection between node 2 and node 4 so this condition is not satisfied

* + **If the size of the Array (ActiveNodes) = 0 then add the two nodes inside the active connection to the array (ActiveNodes) then we set the variable lContinue to True**

This time the size (Count of items inside the array) of the array activeNodes is not zero because we have two items in the array

|  |
| --- |
| 2 |
| 3 |

ActiveNodes Array

So the condition is not satisfied

* + **Else**
    - **For each item in the array ActiveNodes**
      * **If the current item exist in the active connection we can use this connection by adding it’s items to the ActiveNodes without making duplications (we add the other item in the connection if it’s not already added) then we set lContinue to True**
      * **End IF**
    - **End For**

Now at this time this block of code will be executed because the else condition is satisfied

* + - **For each item in the array ActiveNodes**

We will start by (2) then (3) because the ActiveNodes array contains 2 and 3

* + - **If the current item exist in the active connection we can use this connection by adding it’s items to the ActiveNodes without making duplications (we add the other item in the connection if it’s not already added) then we set lContinue to True**

The active connection between node 2 and 4

So we can add the node (4) to the ActiveNodes array

|  |
| --- |
| 2 |
| 3 |
| 4 |

ActiveNodes Array

We still in the connections loop and we will move to the connection number 4 in the connections array

The connection number 4 will become the active connection

The active connection now is the connection between node 3 and node 5

* + - **For each item in the array ActiveNodes**

We will start by (2) then (3) then (4) because the ActiveNodes array contains 2, 3 and 4

* + - **If the current item exist in the active connection we can use this connection by adding it’s items to the ActiveNodes without making duplications (we add the other item in the connection if it’s not already added) then we set lContinue to True**

The active connection between node 3 and 5

So we can add the node (5) to the ActiveNodes array

|  |
| --- |
| 2 |
| 3 |
| 4 |
| 5 |

ActiveNodes Array

We still in the connections loop and we will move to the connection number 5 in the connections array

The connection number 5 will become the active connection

The active connection now is the connection between node 4 and node 5

Using this connection we don’t have anything new to add to the ActiveNodes Array

The while loop will be executed for another time but nothing will be changed to the ActiveNodes array because we already used all the connections

* + **If the count of the items in the array ActiveNodes < nNodesCount – 1 and the number of connections to the active node is greater than** 1 **then the active node is critical node**

Count of the items in the array ActiveNodes = 4

nNodesCount = 5

the condition ActiveNodes < nNodesCount – 1 is not satisfied

* + **Else**
    - **(the active node is not critical**

The else branch will be executed

And the node number (1) is not critical

* **For each node in the network**

At this time the active node will become the node number (2)

Using the algorithm we will see that the first three connections will be avoided because each one of them contains the node number (2) and the first connection that will be used is the connection number four which is the connection between node (3) and node (5)

|  |
| --- |
| 3 |
| 5 |

ActiveNodes Array

Then the connection number 5 will be used which is the connection between node 4 and node 5 so the node 4 will be added

|  |
| --- |
| 3 |
| 5 |
| 4 |

ActiveNodes Array

* + **If the count of the items in the array ActiveNodes < nNodesCount – 1 and the number of connections to the active node is greater than 1** **then the active node is critical node**

Count of the items in the array ActiveNodes = 3

nNodesCount = 5

the condition ActiveNodes < nNodesCount – 1 is satisfied

Then the node number (2) is a **critical node**

========================================================================

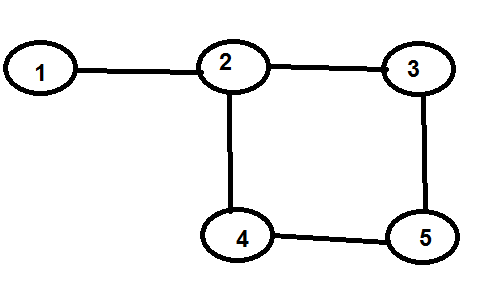
[2] Detection of the critical node based on K-hop neighbors data

The algorithm is similar to the previous algorithm with three simple changes

* The first change is in the connections array which will includes only the connections of the K-Hop neighbors
* The second change is in the nNodesCount which will be

the count of K-hop neighbors + 1

* The third change is related to ActiveNodes array which will be filtered to includes only nodes in the 1-hop neighbors list when K > 1 ( 2-hop detection, 3-hop-detection and so on)



* + For example for 1-hop algorithm and using the same network when we test the node number (1) to know it’s critical or not the connections array will be

|  |  |
| --- | --- |
| 1 | 2 |

ConnectionsArray

And nNodesCount = the count of K-hop neighbors + 1

= the count of 1-hop neighbors for node (1) which is connected to node (2) only so the count of 1-hop neighbors = 1 + 1 = 2

When we test node number (1) using our algorithm we cann’t use the connection between node number (1) and node number (2) because the connection contains the node number (1) and the algorithm avoid connections contains the node that we test

So the ActiveNodes array will be empty

* + **If the count of the items in the array ActiveNodes < nNodesCount – 1 and the number of connections to the active node is greater than 1** **then the active node is critical node**

count of the items in the array ActiveNodes = 0

**nNodesCount – 1 = 2 – 1 = 1**

**But the number of connections to the active node is not greater than 1**

**The number of connections = 1**

The condition is not satisfied so the node is not critical

And when we test the node number (2) to know it’s critical or not the connections array will be

|  |  |
| --- | --- |
| 1 | 2 |
| 2 | 3 |
| 2 | 4 |

We cann’t use any of these connections because each one of them contains the node number (2)

So the ActiveNodes array will be empty

And nNodesCount = the count of K-hop neighbors + 1

= the count of 1-hop neighbors for node (2) + 1 = 3 + 1 = 4

* + **If the count of the items in the array ActiveNodes < nNodesCount – 1 and the number of connections to the active node is greater than 1** **then the active node is critical node**

count of the items in the array ActiveNodes = 0

**nNodesCount – 1 = 4 – 1 = 3**

**number of connections to the active node = 3 which is > 1**

The condition is satisfied so the node is critical