

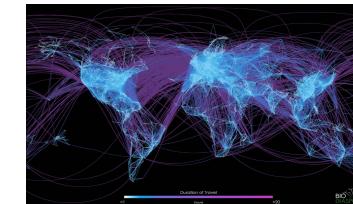
# Connectivity and Robustness in Networks

**Network robustness:** the ability of a network to maintain its general structural properties when it faces failures or attacks.

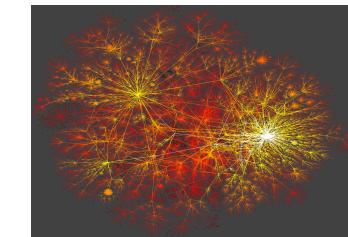
**Type of attacks:** removal of nodes or edges.

**Structural properties:** connectivity.

**Examples:** airport closures, internet router failures, power line failures.



Network of direct flights around the world  
[Bio.Diaspora]



Internet Connectivity [K. C. Claffy ]

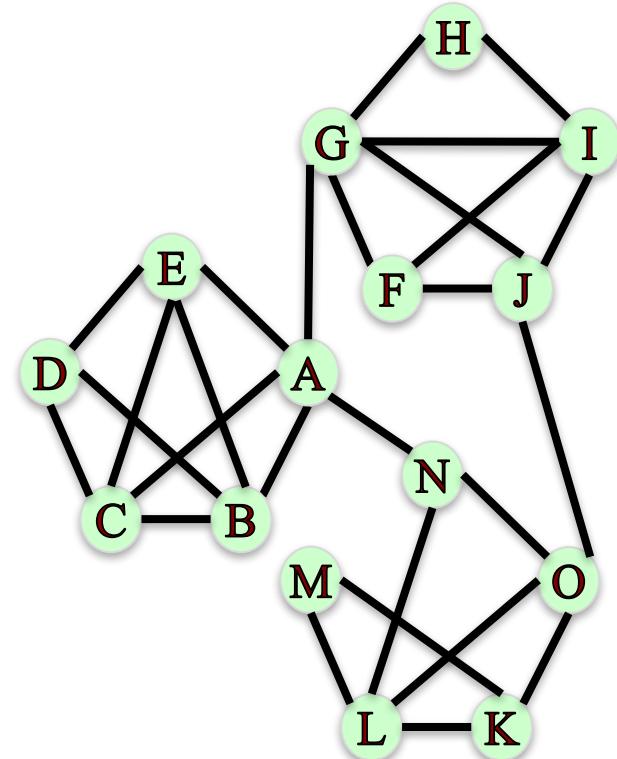
# Disconnecting a Graph

What is the smallest number of nodes that can be removed from this graph in order to disconnect it?

```
In: nx.node_connectivity(G_un)  
Out: 1
```

Which node?

```
In: nx.minimum_node_cut(G_un)  
Out: {'A'}
```



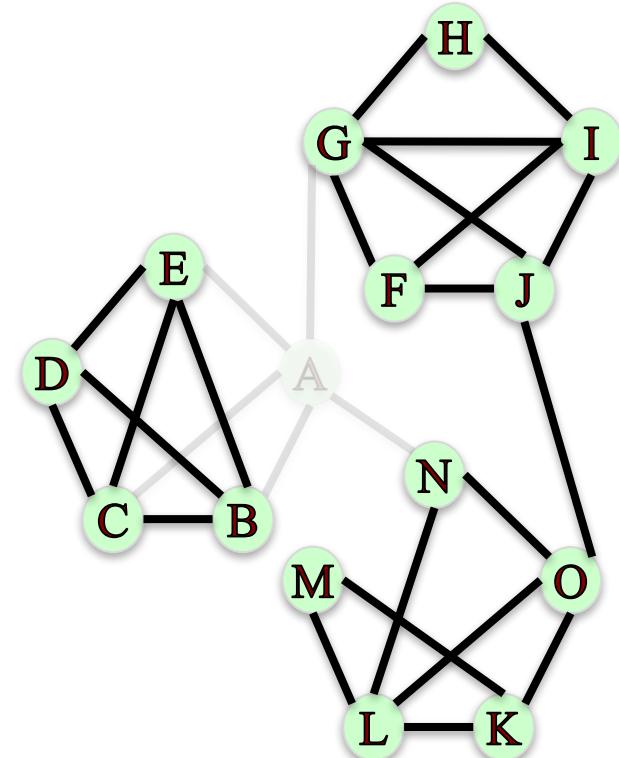
# Disconnecting a Graph

What is the smallest number of nodes that can be removed from this graph in order to disconnect it?

```
In: nx.node_connectivity(G_un)  
Out: 1
```

Which node?

```
In: nx.minimum_node_cut(G_un)  
Out: {'A'}
```



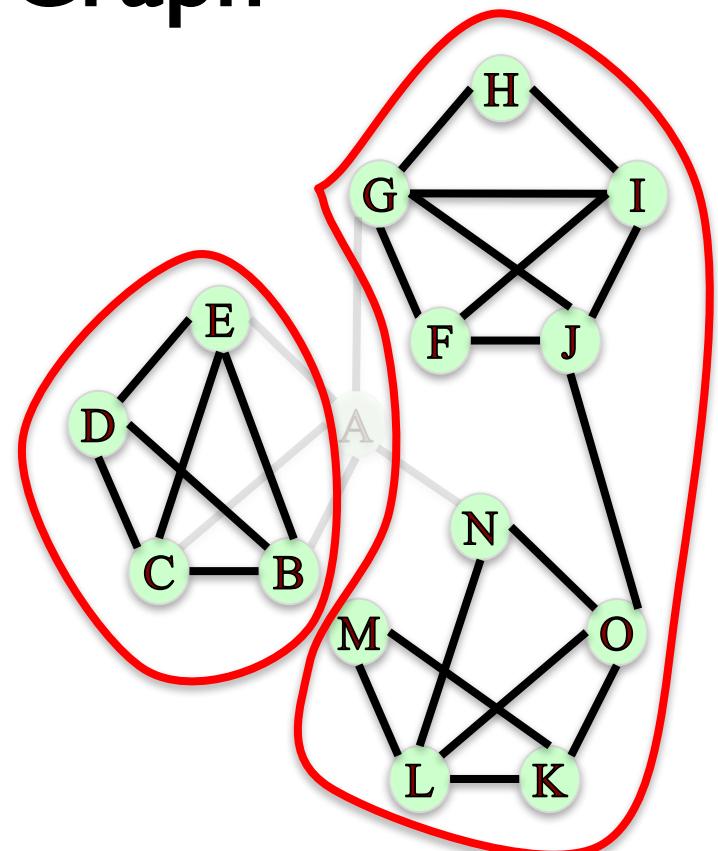
# Disconnecting a Graph

What is the smallest number of **nodes** that can be removed from this graph in order to disconnect it?

```
In: nx.node_connectivity(G_un)  
Out: 1
```

Which node?

```
In: nx.minimum_node_cut(G_un)  
Out: {'A'}
```



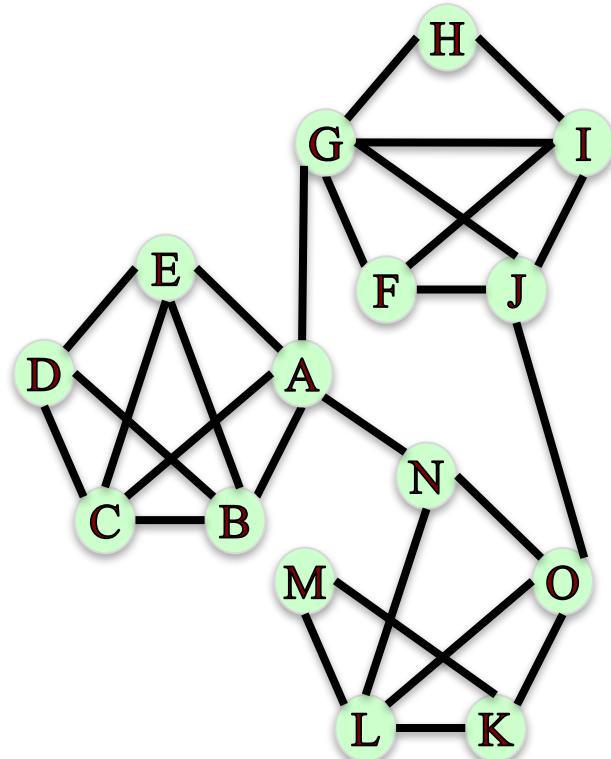
# Disconnecting a Graph

What is the smallest number of **edges** that can be removed from this graph in order to disconnect it?

```
In: nx.edge_connectivity(G_un)  
Out: 2
```

# Which edges?

In: `nx.minimum_edge_cut(G_un)`  
Out: `{('A', 'G'), ('O', 'J')}`



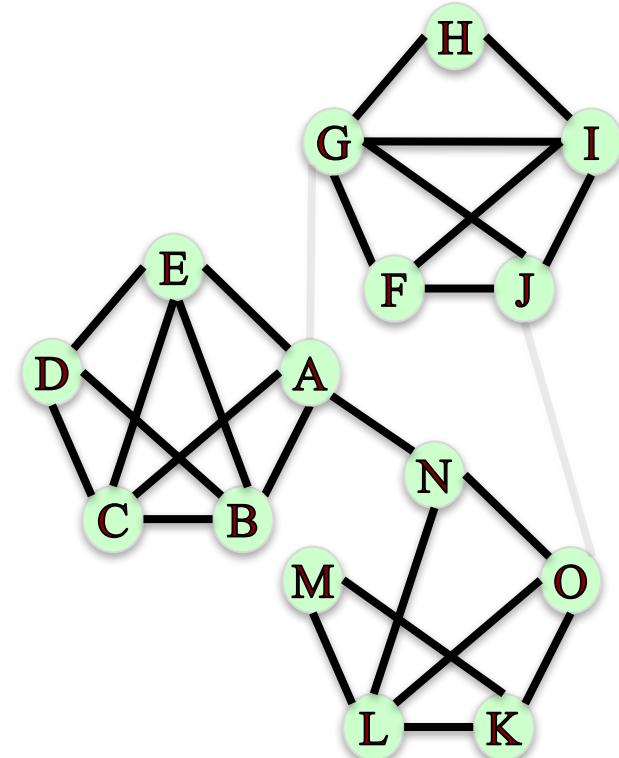
# Disconnecting a Graph

What is the smallest number of **edges** that can be removed from this graph in order to disconnect it?

In: `nx.edge_connectivity(G_un)`  
Out: 2

Which edges?

In: `nx.minimum_edge_cut(G_un)`  
Out: `{('A', 'G'), ('O', 'J')}`



# Disconnecting a Graph

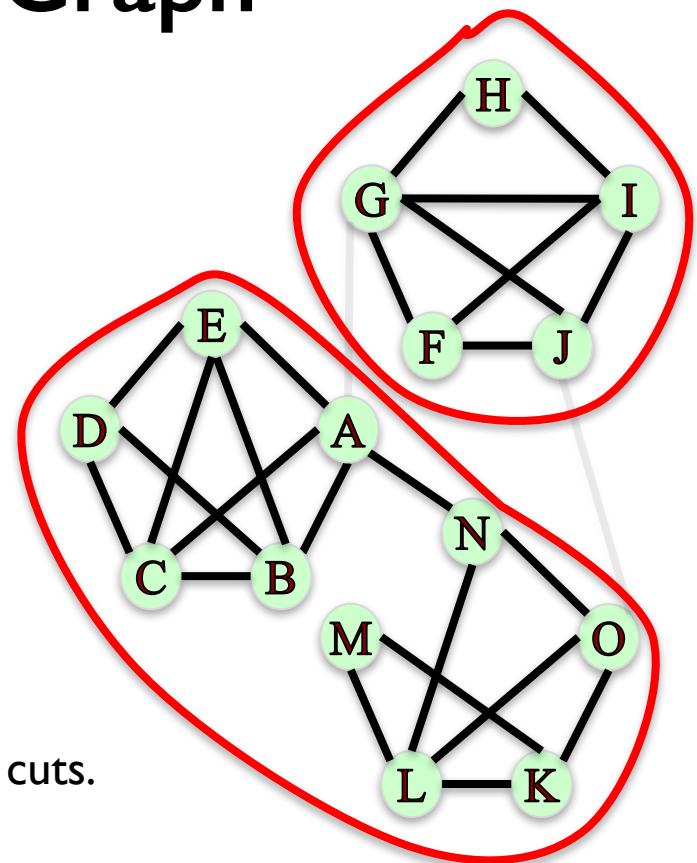
What is the smallest number of **edges** that can be removed from this graph in order to disconnect it?

In: `nx.edge_connectivity(G_un)`  
Out: 2

Which edges?

In: `nx.minimum_edge_cut(G_un)`  
Out: `{('A', 'G'), ('O', 'J')}`

Robust networks have large minimum node and edge cuts.



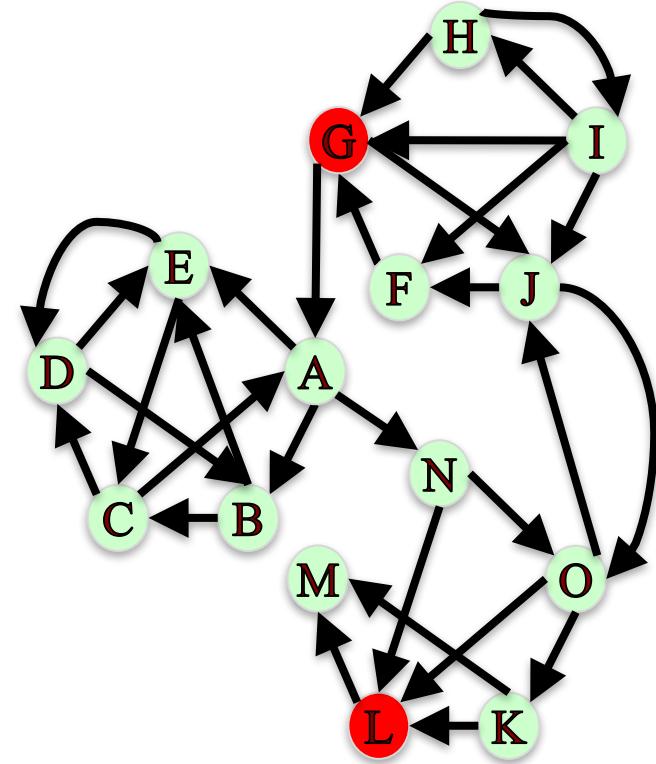
# Simple Paths

Imagine node G wants to send a message to node L by passing it along to other nodes in this network.

What options does G have to deliver the message?

```
In: sorted(nx.all_simple_paths(G, 'G', 'L'))
```

```
Out: [[['G', 'A', 'N', 'L'],
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]
```



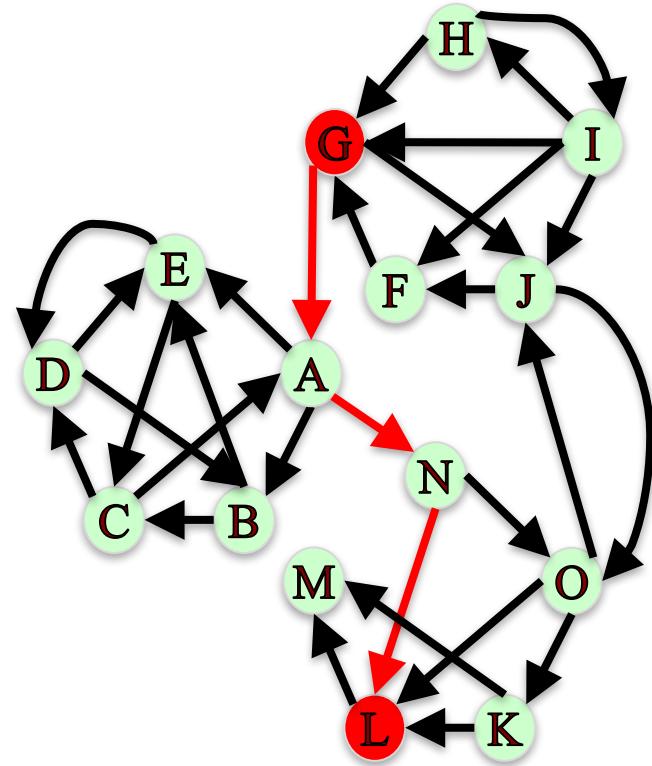
# Simple Paths

Imagine node G wants to send a message to node L by passing it along to other nodes in this network.

What options does G have to deliver the message?

```
In: sorted(nx.all_simple_paths(G, 'G', 'L'))
```

```
Out: [['G', 'A', 'N', 'L'],  
      ['G', 'A', 'N', 'O', 'K', 'L'],  
      ['G', 'A', 'N', 'O', 'L'],  
      ['G', 'J', 'O', 'K', 'L'],  
      ['G', 'J', 'O', 'L']]
```



# Simple Paths

Imagine node G wants to send a message to node L by passing it along to other nodes in this network.

What options does G have to deliver the message?

In: sorted(nx.all\_simple\_paths(G, 'G', 'L'))

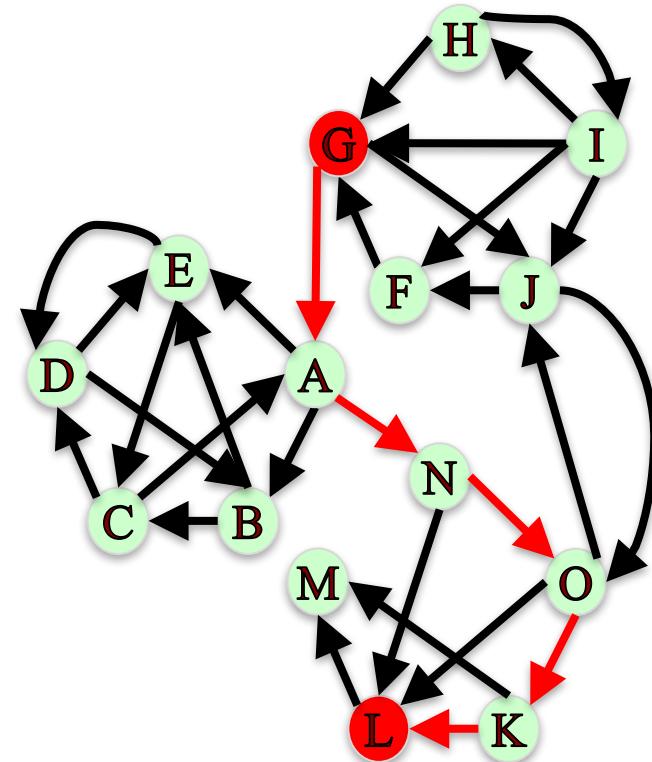
Out: `['G', 'A', 'N', 'L']`

**[‘G’, ‘A’, ‘N’, ‘O’, ‘K’, ‘L’],**

['G', 'A', 'N', 'O', 'L'],

['G', 'J', 'O', 'K', 'L'],

['G', 'J', 'O', 'L']]



# Simple Paths

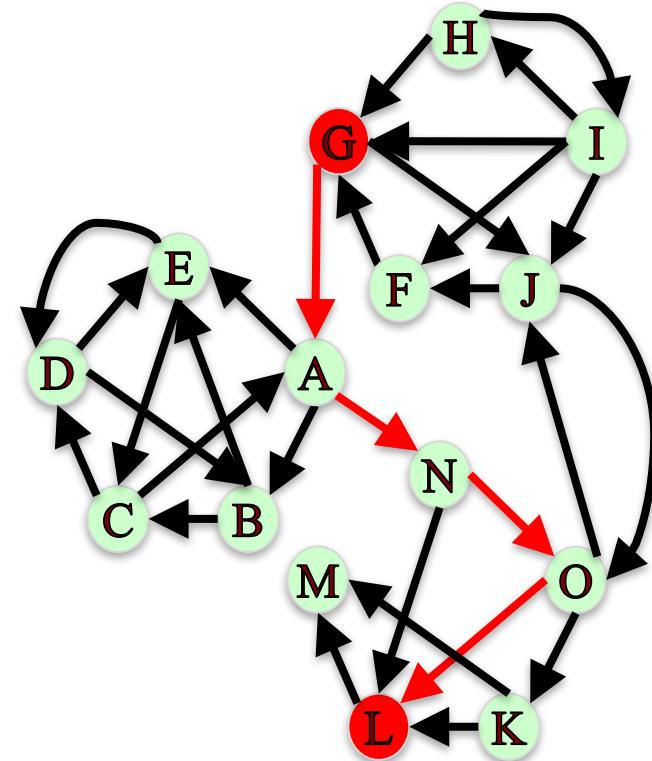
Imagine node G wants to send a message to node L by passing it along to other nodes in this network.

What options does G have to deliver the message?

In: `sorted(nx.all_simple_paths(G, 'G', 'L'))`

Out:

```
[['G', 'A', 'N', 'L'],
 ['G', 'A', 'N', 'O', 'K', 'L'],
 ['G', 'A', 'N', 'O', 'L'],
 ['G', 'J', 'O', 'K', 'L'],
 ['G', 'J', 'O', 'L']]
```



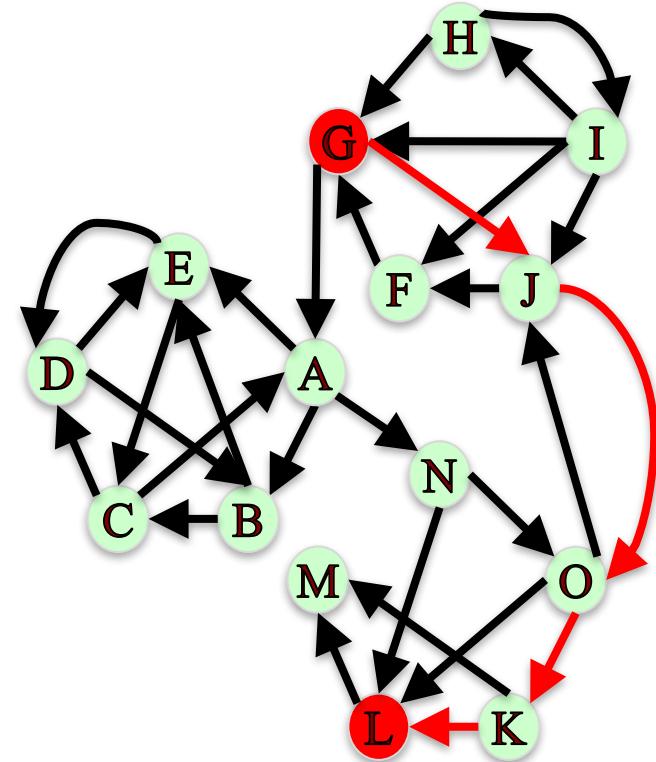
# Simple Paths

Imagine node G wants to send a message to node L by passing it along to other nodes in this network.

What options does G have to deliver the message?

```
In: sorted(nx.all_simple_paths(G, 'G', 'L'))
```

```
Out: [[['G', 'A', 'N', 'L'],
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]
```



# Simple Paths

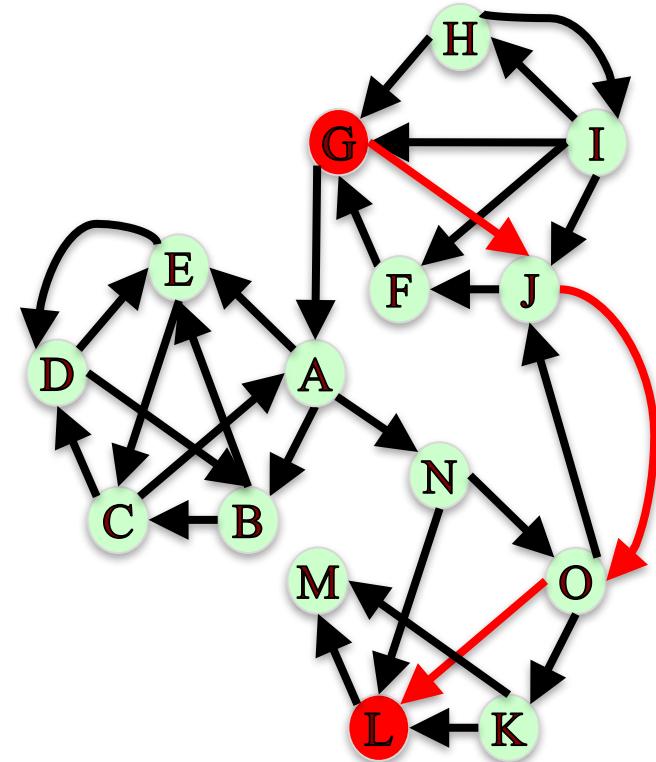
Imagine node G wants to send a message to node L by passing it along to other nodes in this network.

What options does G have to deliver the message?

In: `sorted(nx.all_simple_paths(G, 'G', 'L'))`

Out:

```
[['G', 'A', 'N', 'L'],
 ['G', 'A', 'N', 'O', 'K', 'L'],
 ['G', 'A', 'N', 'O', 'L'],
 ['G', 'J', 'O', 'K', 'L'],
 ['G', 'J', 'O', 'L']]
```



# Node Connectivity

If we wanted to block the message from G to L by removing **nodes** from the network, how many nodes would we need to remove?

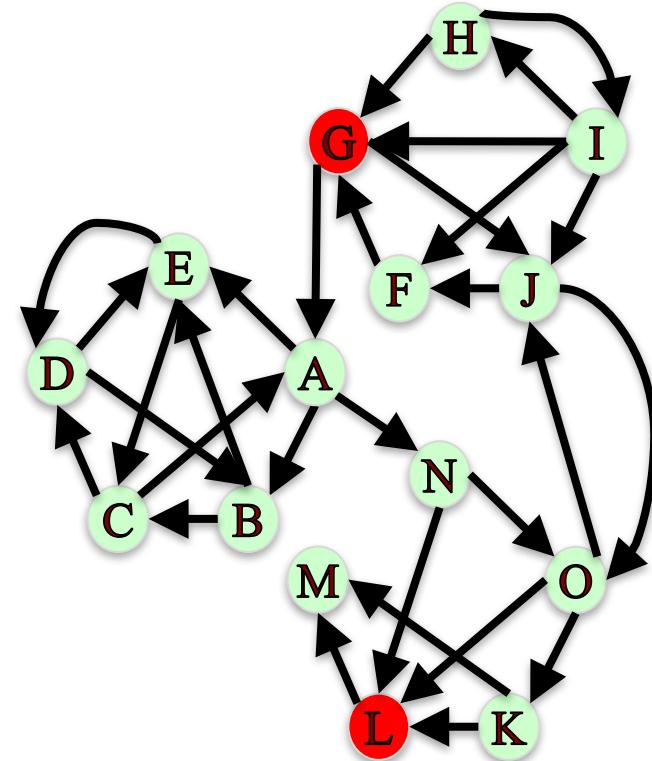
In: `nx.node_connectivity(G, 'G', 'L')`

Out: 2

Which nodes?

In: `nx.minimum_node_cut(G, 'G', 'L')`

Out: `{'N', 'O'}`



# Node Connectivity

If we wanted to block the message from G to L by removing **nodes** from the network, how many nodes would we need to remove?

In: `nx.node_connectivity(G, 'G', 'L')`

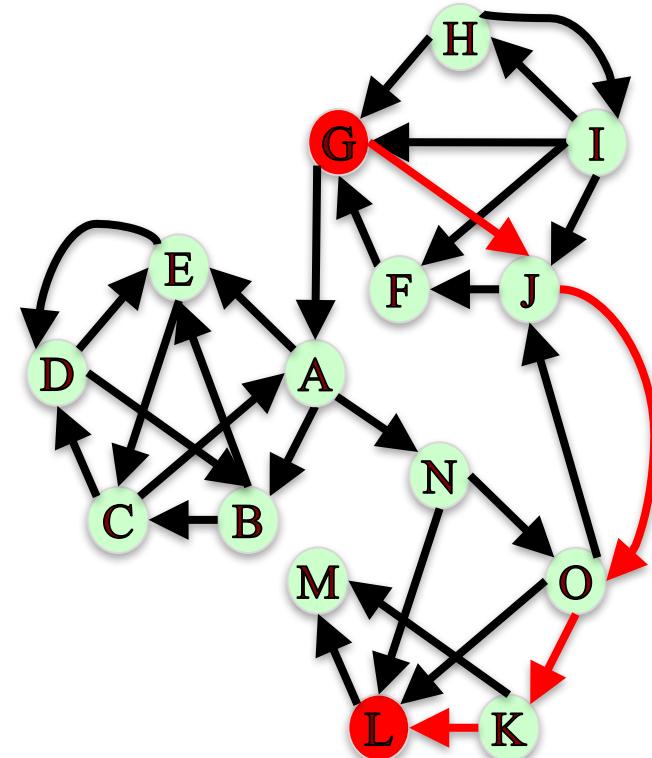
Out: 2

Which nodes?

In: `nx.minimum_node_cut(G, 'G', 'L')`

Out: `{'N', 'O'}`

If we only remove node N, message can go on path  
 $G \rightarrow J \rightarrow O \rightarrow K \rightarrow L$ .



# Node Connectivity

If we wanted to block the message from G to L by removing **nodes** from the network, how many nodes would we need to remove?

```
In: nx.node_connectivity(G, 'G', 'L')
```

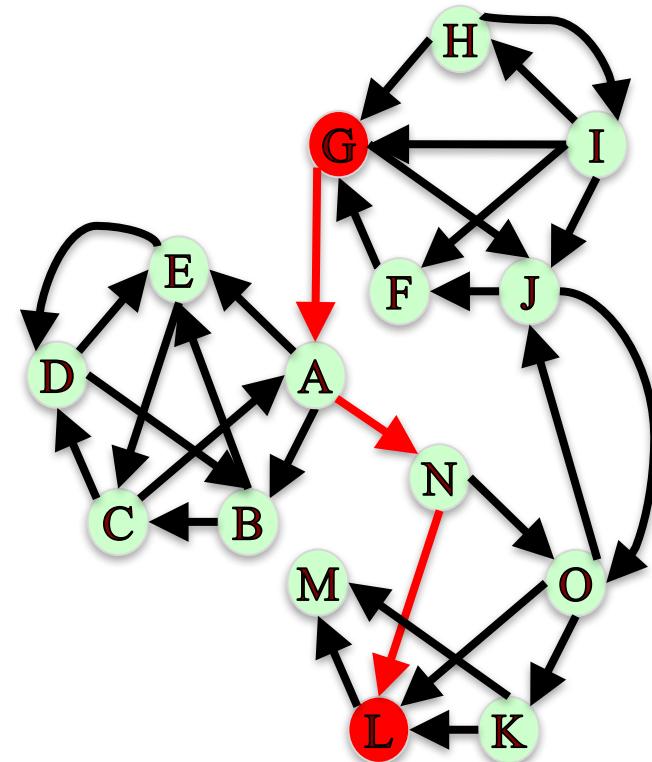
Out: 2

## Which nodes?

```
In: nx.minimum_node_cut(G, 'G', 'L')
```

Out: {'N', 'O'}

If we only remove node O, message can go on path G → A → N → L.



# Edge Connectivity

If we wanted to block the message from G to L by removing **edges** from the network, how many edges would we need to remove?

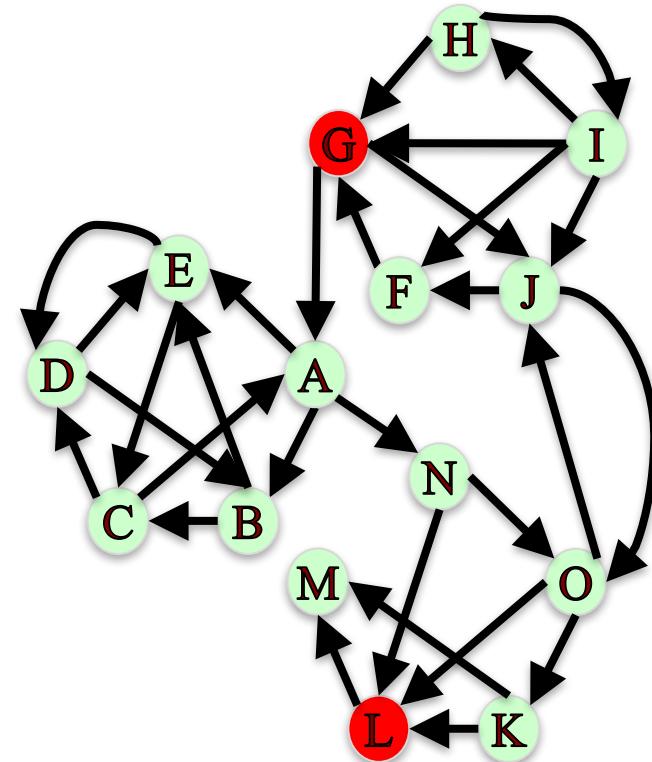
```
In: nx.edge_connectivity(G, 'G', 'L')
```

Out: 2

# Which edges?

In: nx.minimum edge cut(G, 'G', 'L')

Out:  $\{('A', 'N'), ('J', 'O')\}$



# Edge Connectivity

If we wanted to block the message from G to L by removing **edges** from the network, how many edges would we need to remove?

In: `nx.edge_connectivity(G, 'G', 'L')`

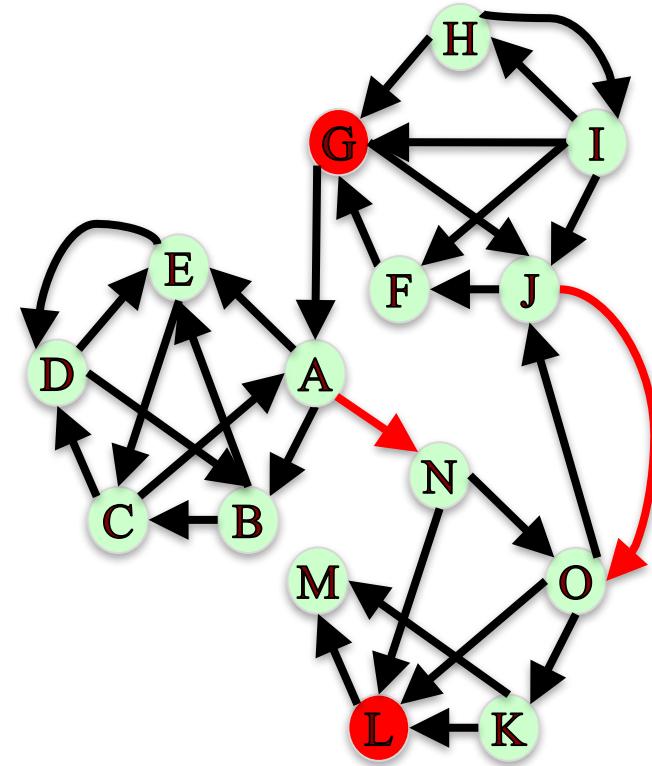
Out: 2

Which edges?

In: `nx.minimum_edge_cut(G, 'G', 'L')`

Out: `{('A', 'N'), ('J', 'O')}`

We need to remove  $A \rightarrow N$  and  $J \rightarrow O$  to block messages from G to L.



# Summary

**Node connectivity:** Minimum number of nodes needed to disconnect a graph or pair of nodes.

```
nx.node_connectivity(G, 'G', 'L')  
nx.minimum_node_cut(G, 'G', 'L')
```

**Edge connectivity:** Minimum number of edges needed to disconnect a graph or pair of nodes.

```
nx.edge_connectivity(G, 'G', 'L')  
nx.minimum_edge_cut(G, 'G', 'L')
```

Graphs with large node and edge connectivity are more robust to the loss of nodes and edges.

