The 2 paths that visit all nodes once are : [1,2,3,5,4,6] and [1,2,5,3,4,6]

Abstraction

We want to develop an algorithm that will give us a general solution to this problem, so that we can simply change the network and run our program to solve the problem.

Some of the key components of the problem are :

- There are objects called nodes. 6 in this example.
- Each node has a name and a list of other nodes to which you can travel.
- There is a start node (node1) and an end node (node6).
- Once we leave a node to travel to a new node, the problem is really just a sub-problem i.e. similar but smaller.
- There are many paths that can go from the start to the end but we are only interested in paths that go through all (6) nodes.
-

Writing your Thinking

Take 2 minutes to think about how you tackled this problem.

- **Did you use pen and paper to help visualise possible solutions?**
- Did you draw a diagram to represent the problem?
- **Did you find a few different solutions, some Hamiltonian and some shorter? How many paths go from start to end while only visiting a node once?**
- Did you use Trial and Error to investigate possible solutions?
- **Did you verify your solution?**

Using Think-Pair-Share-Square (TPSS), go through how you and your partner were thinking about how to solve the problem.

Write an algorithm to solve this problem in a computational way.

Try to write a version of your algorithm in pseudo-code.

Pseudo-Code

Set up the network by creating 6 nodes with their numbers and connected nodes, &c. (use dictionaries to store the attributes)

Assign the startNode and endNode. Initialise the list of possible paths.

Travel the Network (firstNode, currentPath) {

Add the firstNode to the currentPath;

If we are at the endNode **then** {

Store the currentPath in a finalPathList

Return

}

For each node in the connected nodes list {

If the node is in the currentPath do Nothing

Else Travel the Network (node, currentPath)

}

Return

}

For each element of the finalPathList {

print out the paths that are the same length as the network list.

(and possibly also draw an animated version of the solution)

}

Key Note

The pseudo code above is using a Recursive Algorithm, which you will have learned how to use in previous lessons.

Problems that can be solved using recursion have 2 key features:

1. They break down into identical sub-problems
   - .. each sub-network within the main network is identical in every way, only smaller.
2. They have a terminating condition (base case)
   - … is the current node the endNode?