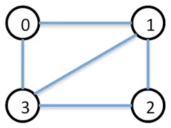
Activity 1. Backtracking and the Graph Colouring Problem

Watch the following video where you can find an explanation about the backtracking algorithm applied to a Graph Colouring Problem:



https://www.youtube.com/watch?v=miCYGGrTwFU

Answer the following questions after watching the video:

1. **n**: refers to the number of nodes (cities) and **m?**:
2. What does the content of the adjacency matrix represent?

n	**0**	**1**	**2**	**3**
0	1	1	0	1
1	1	1	1	1
2	0	1	1	1
3	1	1	1	1

3. Take a look at the code, what do the following variables represent? Link definitions with variables and constants:

a node to be coloured •	• 3
every colour •	• k
the colour assignment for each node •	• x
a node to check if it is adjacent to other •	• G
blue colour •	• c
green colour •	• 2
red colour •	• i
the adjacency matrix •	• 1

4. True or False:
   - **0** means that two nodes are connected
   - Nodes 0 and 2 are not connected
   - **k** is the node we're trying to colour
   - ***return*** breaks the recursion
   - A node is adjacent to itself
   - ***isSafe*** function checks if the node *k* is adjacent to the node *i* that is being checked in the loop and and whether the colour *c* has been already assigned
   - ***Eventually*** is synonym of Finally
   - ***Edges*** are the same as Arcs between nodes
   - ***Edges*** are vertices

5. Explain the meaning of the following sentence, with your own words.

```
If G[k][i]==1 && c==x[i]
     return false
```

6. Assuming the following state of the problem, give a trace of the execution of the backtracking code:

   x=[2 3 0 0]

   k=3

   graph(k)

7. Write this code in Matlab/Octave and check that everything is ok **debugging** the program.
8. Improve the code, removing return instructions and changing loops when needed. Use specific sentences of Matlab/Octave such as *all, find, etc.*