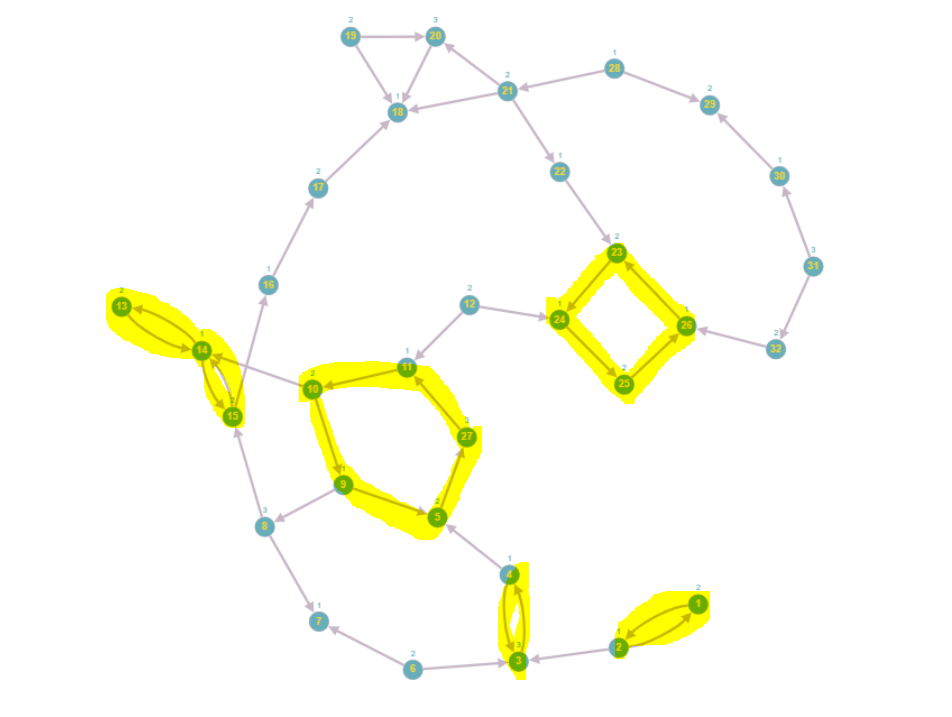
**Q3 a. Assumptions**

We are tasked to write a function to check if there exists a route between any two points on the map. A breadth first search is done from the starting point to get all its visitable vertices. These vertices are stored in a list that is iterated through to check if the destination point is visitable. If this point is present in the list, the route exists on the map.

**Q3 b. Assumptions**

We are tasked with writing a function that finds all the cycles in the graph. To accomplish this, we start a depth first search (DFS) from a starting vertex and marked the current vertex as visited and push the visited vertex in a recursion stack. Then, we are recursively calling DFS for all the adjacent vertices and checking if the vertex is in the recursion stack. If yes, the graph is cyclic. Below, we can distinctly see the cycles in the graph, which the output of our function returns.



**Figure 1: Directed Graph with Highlighted Cycles**

**Q3 c. Assumptions**

We have written functions that find all paths from one point to another using a depth first search. Using these paths, we can determine at which point the paths split and converge to a common point. The points that the paths are dependent on are the points leading towards the paths splitting and converging.

e.g., the paths from 1 to 13 are:  
[1, 2, 3, 4, 5, 27, 11, 10, 14, 13]  
[1, 2, 3, 4, 5, 27, 11, 10, 9, 8, 15, 14, 13]

The point, in this case, where the paths split is 10. Therefore, the dependent points are:  
[1, 2, 3, 4, 5, 27, 11, 10]