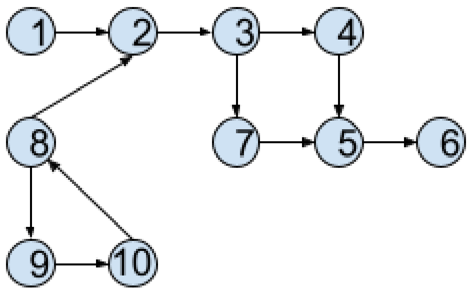
MEDIUM

Detect Cycle in a Directed Graph

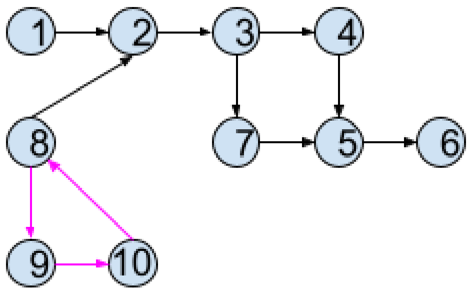
Intuition



We will be using DFS to solve this particular algorithm

Does this graph have a cycle ?

Yes



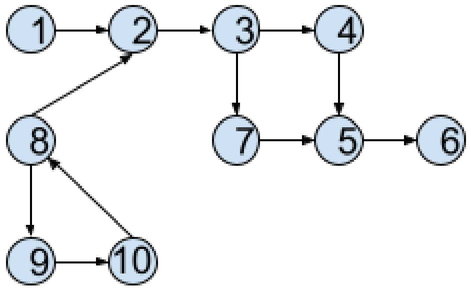
The same algorithm will not work for Undirected and Directed graph, therefore we need to modify it a little.

When we are at 7 we cannot go to other components after traversal and this will tell us that there is a cycle but which is not the case the cycle is present at 8 | 9 | 10

Inference : On the same path the node has to be visited again

We will create an adjacency list and perform some modification of the DFS traversal technique to solve this particular problem

Eg.



1 - [2]
2 - [3]
3 - [4,7]
4 - [5]
5 - [6]
6 - []
7 - [5]
8 - [2,9]
9 - [10]
10 - [8]

We will have two visited arrays :

- Visited array
- Path visited array

Visited = [1 1 1 1 1 1 1 1 1 1]
Path = [0 0 0 0 0 0 1 1 1 1]

Component - 1

DFS(1)
DFS(2)
DFS(3)
DFS(4)DFS(7)
DFS(5)DFS(5) - Visited | Path Unvisited hence not a cycle
DFS(6)
Return false

Component - 2

DFS(8)
DFS(2) - Visited | Path Unvisited hence not a cycle DFS(9)
DFS(10)

DFS(8) - Visited and path visited hence it is a cycle - return true

Approach

Function Code

```
bool dfs(int node,vector<int> adj[],int visited[],int pathvisited[])
    {
        //mark the source node as 1 on visited and the path visited array
        visited[node] = 1;
        pathvisited[node] = 1;
        //traverse through the adjacent node
        for(int i:adj[node])
        {
            // check if the adjacent node is still unvisited
            if(!visited[i])
            {
                //check if the node leads to a cycle if yes return true
                if(dfs(i,adj,visited,pathvisited) == true)return true;
            }
            else
            {
                //if its visited and also pathvisited it has a cycle
                if(pathvisited[i])
                {
                    // return true
                    return true;
                }
            }
        }
        //unvisit the path of the node after traversal as now path has been
changed

        pathvisited[node] = 0;
        //return false outside the loop
        return false;
    }
    //Function to detect cycle in a directed graph.
    bool isCyclic(int V, vector<int> adj[]) {
        //Declare
        //Visited array of size V having intilization of all elements as 0
        int visited[V] = {0};
        //Pathvisited array of size v having initialization of all elements
as 0
        int pathvisited[V] = {0};
        //traverse for all components of the graph
        for(int i=0;i<V;i++)
        {
            //check if the component is not visited
            if(!visited[i])
            {
                //if its unvisited check if there is a cycle using dfs call
                if(dfs(i,adj,visited,pathvisited)==true)
                {
                    //return true that there is a cycle
                    return true;
                }
            }
        }
        //return false that there is no cycle
        return false;
    }
```

Time Complexity

O(v+E)