MICROSOFT PREPARATION

DAY : 03

Detect Cycle in a Directed Graph :

Problem Link :
https://practice.geeksforgeeks.org/problems/detect-cycle-in-a-directed-graph/1?utm_source=geeksforgeeks&utm_medium=ml_article_practice_tab&utm_campaign=article_practice_tab

Test Cases Passed : 410 / 410

Time Used : 04.39

Difficulty Level : MEDIUM

Approach Used :

- Declare vector for counting indegree initialized with indegree of all elements
- Create an empty queue
- Insert all elements having 0 indegree to the queue
- Create a counter variable to count the topological sort elements
- Traverse until the queue becomes empty :
    - Extract the first element of the queue
    - Pop the first element of the queue
    - Increment the counter
    - Traverse for adjacent elements :
        - Decrease the indegree of elements by 1
        - Check if the indegree of any adjacent element becomes 0 :
            - If indegree becomes 0 push it to the queue
    - If the counter is equal to number of nodes then there is no cycle : return false
    - Return true // there is a cycle

Solution :

```
// Function to detect cycle in a directed graph.
    bool isCyclic(int n, vector<int> adj[]) {
        // Declare a indegree vector
        vector<int> indegree(n,0);
        // initializing indegree
        for(int i=0;i<n;i++)
        {
            for(auto it : adj[i])
            {
                indegree[it]++;
            }
        }

        // creating an empty queue

        queue<int> q;

        // creating a counter variable

        int counter = 0;

        // initializing the queue

        for(int i =0;i<n;i++)
        {
            if(indegree[i]==0)
            {
                q.push(i);
            }
        }

        // traversing while the queue becomes empty

        while(!q.empty())
        {
            int node = q.front();
            q.pop();
            counter++;

            // traversing through the adjacent elements
            for(int i:adj[node])
            {
                indegree[i]-=1;
                if(indegree[i]==0)
                {
                    q.push(i);
                }
            }
        }
        if(counter==n)return false;
        return true;
    }
```