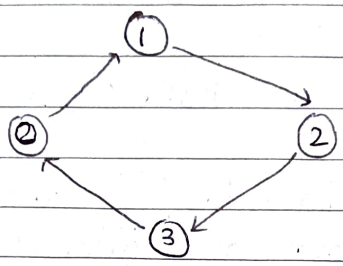
Date _____ Page _____

Day - 202

Graph - 6

* **Cycle in a Directed Graph:**



⇉ So, cycle detection is used in the detection of the deadlock.

⇉ Here, we can't use our previous logic of cycle detection. (that visited approach)

⇉ So, now, if at the time if traversing a path if we get any vertex twice then that means cycle exists.

⇉ Now, how to track current path?

⇉ We will create a path array of size V.

⇉ Then we will start traversing our graph.

⇉ We will write 1 for visited node in path.

⇉ So, if we get any visited node ~~we~~ that means cycle present.

Date _____ Page _____

Code

```
bool detectCycle (int node, vector< int> adj[],
        vector < bool > & path ){
        path [ Node ] = 1;
        for(int j= 0; j < adj[node ].size() ; j++){
            if( path[ adj [node ][j] )
                return 1;
            if( detectcycle( adj [node][j], adj, path))
                return 1;
        }
        path [node ] = 0;
        return 0;
}
```

⇉ This is not our most optimized code.

⇉ Suppose, if we visit a path that don't have any ~~else~~ cycle then in the next visit, there will be no cycle too.

⇉ So, visiting this path again, increases our time complexity.

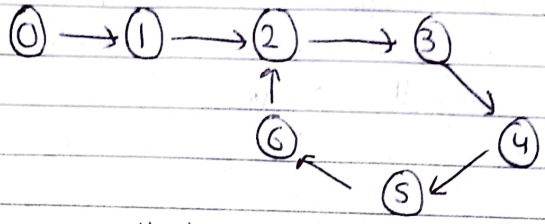
⇉ So, we will use a visited array.

⇉ First, we check that node for again in the path.

⇉ If not then check that node if it is visited or not.

Date _____ Page _____

* Cycle Detection using BFS:



⇉ We know that, we can't ~~apply~~ find Topological Sort of Directed Cyclic Graph.

⇒ So, we can use this logic to find cycle in the graph.

⇉ We will use Kahn's algo.

indegree

0	1	2	3	4	5	6
0	1	2	1	1	1	1

0 1 [0 1] → [0 0 1 1 1 1 1]

⇒ Now, we only get 0 & 1.

⇉ There will be no 0 indegree nodes remaining.

⇉ So, if DAG → count == V

DCG → count != V