***Week – 8 (******06.06.2021 – 12.06.2021)***

***CODES IN PDF***

1. ***All Paths From Source to Target:***

class Solution {

public:

void dfs(vector<vector<int>>& graph, vector<vector<int>>& res, vector<int> path, int node)

{

path.push\_back(node);

if(node == graph.size()-1) res.push\_back(path);

else for(int neigh: graph[node]) dfs(graph, res, path, neigh);

path.pop\_back();

}

vector<vector<int>> allPathsSourceTarget(vector<vector<int>>& graph) {

vector<vector<int>> res;

dfs(graph, res, {}, 0);

return res;

}

};

1. ***Cycle in Directed Graph:***

bool dfs(map<int,vector<int>> &m, int s, vector<bool> &visited,vector<bool> &temp)

{

visited[s]=true;

temp[s]=true;

for(int u:m[s])

{

if(visited[u]==false && dfs(m,u,visited,temp)==true)

return true;

else if(temp[u]==true)

return true;

}

temp[s]=false;

return false;

}

int Solution::solve(int A, vector<vector<int> > &B) {

map<int,vector<int>> m;

vector<bool> visited(A, false);

vector<bool> temp(A, false);

for(auto i : B)

m[i[0]].push\_back(i[1]);

for(int i=0;i<A;i++)

{

if(visited[i]==false)

if(dfs(m,i,visited,temp)==true)

return 1;

}

return 0;

}

1. ***Cycle in Undirected Graph:***

int findP(int check [],int x)

{

while(check[x]!=-1) x=check[x];

return x;

}

int Solution::solve(int A, vector<vector<int> > &B) {

int check[A+1], x, y, i;

memset(check,-1,sizeof(check));

for(i=0;i<B.size();i++)

{

x=findP(check,B[i][0]);

y=findP(check,B[i][1]);

if(x==y) return 1;

check[x]=y;

}

return 0;

}