**Q) DFS**

**Execution Time:0.36**

vector<int> v;

void stk(int t,vector<int> g[], int N,bool visited[])

{

visited[t]=true;

v.push\_back(t);

for(auto i=g[t].begin();i!=g[t].end();++i)

{

if(!visited[\*i])

{

visited[\*i]=true;

stk(\*i,g,N,visited);

}

}

}

vector <int> dfs(vector<int> g[], int N)

{

v.clear();

bool \* visited= new bool[N];

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

visited[i]=false;

stk(0,g,N,visited);

return v;

}

**Check if the directed graph is connected or not?**

// C++ implementation of the approach

#include <bits/stdc++.h>

using namespace std;

#define N 100000

// To keep correct and reverse direction

vector<int> gr1[N];

// Function to add edges

void Add\_edge(int u, int v)

{

gr1[u].push\_back(v);

}

void dfs(int t, bool visited[],vector<int> gr1[])

{

visited[t]=true;

for(auto i=gr1[t].begin();i!=gr1[t].end();++i)

{

if(!visited[\*i])

{

visited[\*i]=true;

dfs(\*i,visited,gr1);

}

}

}

bool Is\_Connected(int n)

{

bool \*visited= new bool[n];

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

visited[i]=false;

dfs(0,visited,gr1);

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

if( visited[i]==false)

return false;

return true;

}

// Driver code

int main()

{

int n = 4;

// Add edges

Add\_edge(0, 1);

Add\_edge(0, 2);

Add\_edge(1, 2);

// Function call

if (Is\_Connected(n))

cout << "Yes";

else

cout << "No";

return 0;

}