#include <iostream>

#include <vector>

#include <stack>

#include <omp.h>

using namespace std;

const int MAX = 100000;

vector<int> graph[MAX];

bool visited[MAX];

void dfs(int node) {

stack<int> s;

s.push(node);

while (!s.empty()) {

int curr\_node = s.top();

s.pop();

if (!visited[curr\_node]) {

visited[curr\_node] = true;

if (visited[curr\_node]) {

cout << curr\_node << " ";

}

#pragma omp parallel for

for (int i = 0; i < graph[curr\_node].size(); i++) {

int adj\_node = graph[curr\_node][i];

if (!visited[adj\_node]) {

s.push(adj\_node);

}

}

}

}

}

int main() {

int n, m, start\_node;

cout << "Enter No of Node,Edges,and start node:" ;

cin >> n >> m >> start\_node;

//n: node,m:edges

cout << "Enter Pair of edges:" ;

for (int i = 0; i < m; i++) {

int u, v;

cin >> u >> v;

//u and v: Pair of edges

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

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

}

#pragma omp parallel for

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

visited[i] = false;

}

dfs(start\_node);

/\* for (int i = 0; i < n; i++) {

if (visited[i]) {

cout << i << " ";

}

}\*/

return 0;

}

Output:

Here's an example input and output for a small graph with 6 nodes and 5 edges:

