import java.util.\*;

// Node class representing a node in the graph

class Node {

int value;

List<Node> neighbors;

public Node(int value) {

this.value = value;

this.neighbors = new ArrayList<>();

}

public void addNeighbor(Node neighbor) {

this.neighbors.add(neighbor);

}

}

// Graph class representing a graph

class Graph {

List<Node> nodes;

public Graph() {

this.nodes = new ArrayList<>();

}

public void addNode(Node node) {

this.nodes.add(node);

}

}

public class Main {

// Depth First Search (DFS) algorithm

public static void DFS(Node start) {

Set<Node> visited = new HashSet<>();

DFSHelper(start, visited);

}

private static void DFSHelper(Node node, Set<Node> visited) {

if (node == null || visited.contains(node))

return;

System.out.print(node.value + " ");

visited.add(node);

for (Node neighbor : node.neighbors) {

DFSHelper(neighbor, visited);

}

}

// Breadth First Search (BFS) algorithm

public static void BFS(Node start) {

Queue<Node> queue = new LinkedList<>();

Set<Node> visited = new HashSet<>();

queue.add(start);

visited.add(start);

while (!queue.isEmpty()) {

Node current = queue.poll();

System.out.print(current.value + " ");

for (Node neighbor : current.neighbors) {

if (!visited.contains(neighbor)) {

queue.add(neighbor);

visited.add(neighbor);

}

}

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of nodes in the graph: ");

int numNodes = scanner.nextInt();

Graph graph = new Graph();

Map<Integer, Node> nodeMap = new HashMap<>();

// Create nodes

for (int i = 1; i <= numNodes; i++) {

Node node = new Node(i);

graph.addNode(node);

nodeMap.put(i, node);

}

System.out.print("Enter the number of edges in the graph: ");

int numEdges = scanner.nextInt();

System.out.println("Enter the edges (node1 node2):");

// Add edges

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

int node1 = scanner.nextInt();

int node2 = scanner.nextInt();

nodeMap.get(node1).addNeighbor(nodeMap.get(node2));

nodeMap.get(node2).addNeighbor(nodeMap.get(node1)); // if the graph is undirected

}

System.out.println("Depth First Search (DFS) traversal:");

DFS(nodeMap.get(1));

System.out.println("\nBreadth First Search (BFS) traversal:");

BFS(nodeMap.get(1));

}

}