import java.util.\*;

public class GraphTraversal {

static final int MAXN = 100; // maximum number of nodes

static ArrayList<Integer>[] adjList = new ArrayList[MAXN]; // adjacency list

static int[] dist = new int[MAXN]; // distance array for BFS

static boolean[] visited = new boolean[MAXN]; // visited array for DFS

// DFS function

static void dfs(int node) {

visited[node] = true;

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

for (int i : adjList[node]) {

if (!visited[i]) {

dfs(i);

}

}

}

// BFS function

static void bfs(int node) {

Arrays.fill(dist, -1);

dist[node] = 0;

Queue<Integer> q = new LinkedList<>();

q.add(node);

while (!q.isEmpty()) {

int u = q.poll();

System.out.print(u + " ");

for (int v : adjList[u]) {

if (dist[v] == -1) {

dist[v] = dist[u] + 1;

q.add(v);

}

}

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n, m; // number of nodes and edges

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

n = scanner.nextInt();

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

m = scanner.nextInt();

// initialize adjacency list

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

adjList[i] = new ArrayList<>();

}

// add edges to the adjacency list

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

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

int u, v;

System.out.print("Enter edge " + (i+1) + " (u v): ");

u = scanner.nextInt();

v = scanner.nextInt();

adjList[u].add(v);

adjList[v].add(u); // Assuming undirected graph

}

// perform DFS and BFS on the graph

int startNode;

System.out.print("Enter the starting node for DFS and BFS: ");

startNode = scanner.nextInt();

System.out.print("DFS: ");

Arrays.fill(visited, false);

dfs(startNode);

System.out.println();

System.out.print("BFS: ");

bfs(startNode);

System.out.println();

}

}

**Output:**

**Tree:**

Enter the number of nodes: 7

Enter the number of edges: 6

Enter the edges:

Enter edge 1 (u v): 0 1

Enter edge 2 (u v): 0 2

Enter edge 3 (u v): 0 3

Enter edge 4 (u v): 1 4

Enter edge 5 (u v): 1 6

Enter edge 6 (u v): 2 5

Enter the starting node for DFS and BFS: 0

DFS: 0 1 4 6 2 5 3

BFS: 0 1 2 3 4 6 5

**Graph**:

Enter the number of nodes: 6

Enter the number of edges: 8

Enter the edges:

Enter edge 1 (u v): 1 0

Enter edge 2 (u v): 1 2

Enter edge 3 (u v): 0 2

Enter edge 4 (u v): 0 3

Enter edge 5 (u v): 2 4

Enter edge 6 (u v): 3 4

Enter edge 7 (u v): 3 5

Enter edge 8 (u v): 4 5

Enter the starting node for DFS and BFS: 1

DFS: 1 0 2 4 3 5

BFS: 1 0 2 3 4 5