import java.util.\*;

/\*

\*Enter the number of vertices: 10

Enter the number of edges: 13

Enter edges (vertex1 vertex2):

0 1

0 3

3 2

2 9

2 8

1 2

1 4

1 6

1 7

4 5

4 7

4 6

6 7

Enter the starting vertex for traversal: 0

Breadth-First Search (BFS):

0 1 3 2 4 6 7 9 8 5

Depth-First Search (DFS) :

0 3 2 1 7 6 4 5 8 9

\*/

public class BfsAndDfs {

private int V; // Number of vertices

private LinkedList<Integer>[] adj; // Adjacency list representation

public BfsAndDfs(int v) {

V = v;

adj = new LinkedList[V];

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

adj[i] = new LinkedList<>();

}

}

// Add an edge to the graph

public void addEdge(int v, int w) {

adj[v].add(w);

adj[w].add(v);

}

// Breadth-First Search

public void bfs(int start) {

boolean[] visited = new boolean[V];

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

visited[start] = true;

queue.add(start);

while (!queue.isEmpty()) {

int current = queue.poll();

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

for (int neighbor : adj[current]) {

if (!visited[neighbor]) {

visited[neighbor] = true;

queue.add(neighbor);

}

}

}

}

// Depth-First Search

public void dfs(int start) {

boolean[] visited = new boolean[V];

Stack<Integer> stack = new Stack<>();

stack.push(start);

while (!stack.isEmpty()) {

int current = stack.pop();

if (!visited[current]) {

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

visited[current] = true;

for (int neighbor : adj[current]) {

if (!visited[neighbor]) {

stack.push(neighbor);

}

}

}

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.*in*);

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

int numVertices = scanner.nextInt();

BfsAndDfs graph = new BfsAndDfs(numVertices);

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

int numEdges = scanner.nextInt();

System.*out*.println("Enter edges (vertex1 vertex2):");

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

int vertex1 = scanner.nextInt();

int vertex2 = scanner.nextInt();

graph.addEdge(vertex1, vertex2);

}

System.*out*.print("Enter the starting vertex for traversal: ");

int startVertex = scanner.nextInt();

System.*out*.println("Breadth-First Search (BFS):");

graph.bfs(startVertex);

System.*out*.println("\nDepth-First Search (DFS) :");

graph.dfs(startVertex);

}

}