

2.

public class DFS {

public static void dfs(int i, int[][] graph, boolean[] visited) {

if (!visited[i]) {

visited[i] = true; // Mark node as "visited"

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

for (int j = 0; j < graph[i].length; j++) {

if (graph[i][j] == 1 && !visited[j]) {

dfs(j, graph, visited); // Visit node

}

}

}

}

public static void main(String[] args) {

int[][] adjMatrix =

{

{0, 1, 1, 0, 0, 1, 0, 0, 0},

{1, 0, 0, 0, 0, 1, 0, 0, 0},

{1, 0, 0, 0, 0, 1, 1, 0, 0},

{0, 0, 0, 0, 1, 0, 0, 0, 1},

{0, 0, 0, 1, 0, 0, 0, 0, 1},

{1, 1, 1, 0, 0, 0, 0, 1, 0},

{0, 0, 1, 0, 0, 0, 0, 1, 0},

{0, 0, 0, 0, 0, 1, 1, 0, 0},

{0, 0, 0, 1, 1, 0, 0, 0, 0}

};

boolean[] visited = new boolean[9];

int count = 0;

for (int i = 0; i < adjMatrix.length; i++) {

if (!visited[i]) {

System.out.println("\nCompnent: ");

dfs(i, adjMatrix, visited);

++count;

}

}

System.out.println("\nTotal number of Components: " + count);

}

}

3.

import java.util.LinkedList;

import java.util.Queue;

public class BFS {

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

public static void bfs(int i, int[][] graph, boolean[] visited) {

for (int j = 0; j < graph[i].length; j++) {

if (graph[i][j] == 1 && !visited[j]) {

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

visited[j] = true;

queue.add(j);

}

}

while (!queue.isEmpty()) {

bfs(queue.poll(), graph, visited); // Visit node

}

}

public static void main(String[] args) {

int[][] adjMatrix =

{

{0, 1, 1, 0, 0, 1, 0, 0, 0},

{1, 0, 0, 0, 0, 1, 0, 0, 0},

{1, 0, 0, 0, 0, 1, 1, 0, 0},

{0, 0, 0, 0, 1, 0, 0, 0, 1},

{0, 0, 0, 1, 0, 0, 0, 0, 1},

{1, 1, 1, 0, 0, 0, 0, 1, 0},

{0, 0, 1, 0, 0, 0, 0, 1, 0},

{0, 0, 0, 0, 0, 1, 1, 0, 0},

{0, 0, 0, 1, 1, 0, 0, 0, 0}

};

boolean[] visited = new boolean[9];

int count = 0;

for (int i = 0; i < adjMatrix.length; i++) {

if (!visited[i]) {

System.out.println("\nCompnent: ");

bfs(i, adjMatrix, visited);

++count;

}

}

System.out.println("\nTotal number of Components: " + count);

}

}