Graphs – 1

1. **Code : BFS Traversal**

**Send Feedback**

Given an undirected and disconnected graph G(V, E), print its BFS traversal.

Here you need to consider that you need to print BFS path starting from vertex 0 only.

V is the number of vertices present in graph G and vertices are numbered from 0 to V-1.

E is the number of edges present in graph G.

**Note : 1. Take graph input in the adjacency matrix.**

**2. Handle for Disconnected Graphs as well**

**Input Format :**

Line 1: Two Integers V and E (separated by space)

Next 'E' lines, each have two space-separated integers, 'a' and 'b', denoting that there exists an edge between Vertex 'a' and Vertex 'b'.

**Output Format :**

BFS Traversal (separated by space)

**Constraints :**

2 <= V <= 1000

1 <= E <= 1000

**Sample Input 1:**

4 4

0 1

0 3

1 2

2 3

**Sample Output 1:**

0 1 3 2

1. **Code : Has Path**

**Send Feedback**

Given an undirected graph G(V, E) and two vertices v1 and v2(as integers), check if there exists any path between them or not. Print true or false.

V is the number of vertices present in graph G and vertices are numbered from 0 to V-1.

E is the number of edges present in graph G.

**Input Format :**

Line 1: Two Integers V and E (separated by space)

Next E lines : Two integers a and b, denoting that there exists an edge between vertex a and vertex b (separated by space)

Line (E+2) : Two integers v1 and v2 (separated by space)

**Output Format :**

true or false

**Constraints :**

2 <= V <= 1000

1 <= E <= 1000

0 <= v1, v2 <= V-1

**Sample Input 1 :**

4 4

0 1

0 3

1 2

2 3

1 3

**Sample Output 1 :**

true

**Sample Input 2 :**

6 3

5 3

0 1

3 4

0 3

**Sample Output 2 :**

false

1. **Code : Get Path - DFS**

**Send Feedback**

Given an undirected graph G(V, E) and two vertices v1 and v2(as integers), find and print the path from v1 to v2 (if exists). Print nothing if there is no path between v1 and v2.

Find the path using DFS and print the first path that you encountered.

V is the number of vertices present in graph G and vertices are numbered from 0 to V-1.

E is the number of edges present in graph G.

Print the path in reverse order. That is, print v2 first, then intermediate vertices and v1 at last.

**Note : Save the input graph in Adjacency Matrix.**

**Input Format :**

Line 1: Two Integers V and E (separated by space)

Next E lines : Two integers a and b, denoting that there exists an edge between vertex a and vertex b (separated by space)

Line (E+2) : Two integers v1 and v2 (separated by space)

**Output Format :**

Path from v1 to v2 in reverse order (separated by space)

**Constraints :**

2 <= V <= 1000

1 <= E <= 1000

0 <= v1, v2 <= V-1

**Sample Input 1 :**

4 4

0 1

0 3

1 2

2 3

1 3

**Sample Output 1 :**

3 0 1

**Sample Input 2 :**

6 3

5 3

0 1

3 4

0 3

**Sample Output 2 :**

1. **Code : Get Path - BFS**

**Send Feedback**

Given an undirected graph G(V, E) and two vertices v1 and v2(as integers), find and print the path from v1 to v2 (if exists). Print nothing if there is no path between v1 and v2.

Find the path using BFS and print the shortest path available.

V is the number of vertices present in graph G and vertices are numbered from 0 to V-1.

E is the number of edges present in graph G.

Print the path in reverse order. That is, print v2 first, then intermediate vertices and v1 at last.

**Note : Save the input graph in Adjacency Matrix.**

**Input Format :**

Line 1: Two Integers V and E (separated by space)

Next E lines : Two integers a and b, denoting that there exists an edge between vertex a and vertex b (separated by space)

Line (E+2) : Two integers v1 and v2 (separated by space)

**Output Format :**

Path from v1 to v2 in reverse order (separated by space)

**Constraints :**

2 <= V <= 1000

1 <= E <= 1000

0 <= v1, v2 <= V-1

**Sample Input 1 :**

4 4

0 1

0 3

1 2

2 3

1 3

**Sample Output 1 :**

3 0 1

**Sample Input 2 :**

6 3

5 3

0 1

3 4

0 3

**Sample Output 2 :**

1. **Code : Is Connected ?**

**Send Feedback**

Given an undirected graph G(V,E), check if the graph G is connected graph or not.

V is the number of vertices present in graph G and vertices are numbered from 0 to V-1.

E is the number of edges present in graph G.

**Input Format :**

Line 1: Two Integers V and E (separated by space)

Next 'E' lines, each have two space-separated integers, 'a' and 'b', denoting that there exists an edge between Vertex 'a' and Vertex 'b'.

**Output Format :**

"true" or "false"

**Constraints :**

2 <= V <= 1000

1 <= E <= 1000

**Sample Input 1:**

4 4

0 1

0 3

1 2

2 3

**Sample Output 1:**

true

**Sample Input 2:**

4 3

0 1

1 3

0 3

**Sample Output 2:**

false

**Sample Output 2 Explanation**

The graph is not connected, even though vertices 0,1 and 3 are connected to each other but there isn’t any path from ve

1. **Code : All connected components**

**Send Feedback**

Given an undirected graph G(V,E), find and print all the connected components of the given graph G.

V is the number of vertices present in graph G and vertices are numbered from 0 to V-1.

E is the number of edges present in graph G.

You need to take input in main and create a function which should return all the connected components. And then print them in the main, not inside function.

Print different components in new line. And each component should be printed in increasing order (separated by space). Order of different components doesn't matter.

**Input Format :**

Line 1: Two Integers V and E (separated by space)

Next 'E' lines, each have two space-separated integers, 'a' and 'b', denoting that there exists an edge between Vertex 'a' and Vertex 'b'.

**Output Format :**

Different components in new line

**Constraints :**

2 <= V <= 1000

1 <= E <= 1000

**Sample Input 1:**

4 2

0 1

2 3

**Sample Output 1:**

0 1

2 3

**Sample Input 2:**

4 3

0 1

1 3

0 3

**Sample Output 2:**

0 1 3

2