Q:-1

**Code : BFS Traversal**

Send Feedback

Given an undirected and connected 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.

**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

Q:-2

**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

Q:-3

**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

Q:-4

**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

Q:-5

**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 vertices 0,1,3 to vertex 2.

Q:-6

**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

**Q:-7**

**Islands**

Send Feedback

An island is a small piece of land surrounded by water . A group of islands is said to be connected if we can reach from any given island to any other island in the same group . Given N islands (numbered from 1 to N) and two lists of size M (u and v) denoting island u[i] is connected to island v[i] and vice versa . Can you count the number of connected groups of islands.

**Constraints :**

1<=N<=100

1<=M<=(N\*(N-1))/2

1<=u[i],v[i]<=N

**Input Format :**

Line 1 : Two integers N and M

Line 2 : List u of size of M

Line 3 : List v of size of M

**Output Return Format :**

The count the number of connected groups of islands

**Sample Input :**

2 1

1

2

**Sample Output :**

1

**Q:-8**

**Coding Ninjas**

Send Feedback

Given a NxM matrix containing Uppercase English Alphabets only. Your task is to tell if there is a path in the given matrix which makes the sentence “CODINGNINJA” .

There is a path from any cell to all its neighbouring cells. A neighbour may share an edge or a corner.

**Input Format :**

Line 1 : Two space separated integers N and M, where N is number of rows and M is number of columns in the matrix.

Next N lines : N rows of the matrix. First line of these N line will contain 0th row of matrix, second line will contain 1st row and so on

**Assume input to be 0-indexed based**

**Output Format :**

Return 1 if there is a path which makes the sentence “CODINGNINJA” else return 0.

**Constraints :**

1 <= N <= 100

1 <= M <= 100

**Sample Input :**

2 11

CXDXNXNXNXA

XOXIXGXIXJX

**Sample Output :**

1

**Q:-9**

**Connecting Dots**

Send Feedback

Gary has a board of size NxM. Each cell in the board is a coloured dot. There exist only 26 colours denoted by uppercase Latin characters (i.e. A,B,...,Z). Now Gary is getting bore and wants to play a game. The key of this game is to find a cycle that contain dots of same colour. Formally, we call a sequence of dots d1, d2, ..., dk a cycle if and only if it meets the following condition:

1. These k dots are different: if i ≠ j then di is different from dj.

2. k is at least 4.

3. All dots belong to the same colour.

4. For all 1 ≤ i ≤ k - 1: di and di + 1 are adjacent. Also, dk and d1 should also be adjacent. Cells x and y are called adjacent if they share an edge.

Since Gary is colour blind, he wants your help. Your task is to determine if there exists a cycle on the board.

**Assume input to be 0-indexed based.**

**Input Format :**

Line 1 : Two integers N and M, the number of rows and columns of the board

Next N lines : a string consisting of M characters, expressing colors of dots in each line. Each character is an uppercase Latin letter.

**Output Format :**

Return 1 if there is a cycle else return 0

**Constraints :**

2 ≤ N, M ≤ 50

**Sample Input :**

3 4

AAAA

ABCA

AAAA

**Sample Output :**

1

**Q:-10**

**Largest Piece**

Send Feedback

Its Gary's birthday today and he has ordered his favourite square cake consisting of '0's and '1's . But Gary wants the biggest piece of '1's and no '0's . A piece of cake is defined as a part which consist of only '1's, and all '1's share an edge with eachother on the cake. Given the size of cake N and the cake , can you find the size of the biggest piece of '1's for Gary ?

**Constraints :**

1<=N<=50

**Input Format :**

Line 1 : An integer N denoting the size of cake

Next N lines : N characters denoting the cake

**Output Format :**

Size of the biggest piece of '1's and no '0's

**Sample Input :**

2

11

01

**Sample Output :**

3

**Q:11**

**3 Cycle**

Send Feedback

Given a graph with N vertices and Two Lists (U,V) of size M where (U[i],V[i]) and (V[i],U[i]) are connected by an edge , then count the distinct 3-cycles in the graph. A 3-cycle PQR is a cycle in which (P,Q), (Q,R) and (R,P) are connected an edge.

**Input Format :**

Line 1 : Two integers N and M

Line 2 : List u of size of M

Line 3 : List v of size of M

**Return Format :**

The count the number of 3-cycles in the given Graph

**Constraints :**

1<=N<=100

1<=M<=(N\*(N-1))/2

1<=u[i],v[i]<=N

**Sample Input:**

3 3

1 2 3

2 3 1

**Sample Output:**

1