Program –

*#include* <iostream>

*#include* <bits/stdc++.h>

*using* *namespace* std;

*class* Graph

{

*// Number of vertex*

*int* v;

*// Number of edges*

*int* e;

*// Adjacency matrix*

*int* *\*\**adj;

*public:*

*// To create the initial adjacency matrix*

    Graph(*int* *v*, *int* *e*);

*// Function to insert a new edge*

*void* addEdge(*int* *start*, *int* *e*);

*// Function to display the BFS traversal*

*void* BFS(*int* *start*);

};

*// Function to fill the empty adjacency matrix*

Graph::Graph(*int* *v*, *int* *e*)

{

*this*->v *=* *v*;

*this*->e *=* *e*;

    adj *=* *new* *int* *\**[*v*];

*for* (*int* row *=* 0; row *<* *v*; row*++*)

    {

        adj[row] *=* *new* *int*[*v*];

*for* (*int* column *=* 0;

             column *<* *v*; column*++*)

        {

            adj[row][column] *=* 0;

        }

    }

}

*// Function to add an edge to the graph*

*void* Graph::addEdge(*int* *start*, *int* *e*)

{

*// Considering a bidirectional edge*

    adj[*start*][*e*] *=* 1;

    adj[*e*][*start*] *=* 1;

}

*// Function to perform BFS on the graph*

*void* Graph::BFS(*int* *start*)

{

*// Visited vector to so that*

*// a vertex is not visited more than once*

*// Initializing the vector to false as no*

*// vertex is visited at the beginning*

    vector*<bool>* visited(v, false);

    vector*<int>* q;

    q.push\_back(*start*);

*// Set source as visited*

    visited*[start]* *=* true;

*int* vis;

*while* (*!*q.empty())

    {

        vis *=* q*[*0*]*;

*// Print the current node*

        cout *<<* vis *<<* " ";

        q.erase(q.begin());

*// For every adjacent vertex to the current vertex*

*for* (*int* i *=* 0; i *<* v; i*++*)

        {

*if* (adj[vis][i] *==* 1 *&&* (*!*visited*[*i*]*))

            {

*// Push the adjacent node to the queue*

                q.push\_back(i);

*// Set*

                visited*[*i*]* *=* true;

            }

        }

    }

}

*// Driver code*

*int* main()

{

*int* v *=* 5, e *=* 4;

*// Create the graph*

    Graph G(v, e);

    G.addEdge(0, 1);

    G.addEdge(0, 2);

    G.addEdge(1, 3);

    G.BFS(0);

}

Output-

