1 .BFS

```
#include <iostream>
#include <bits/stdc++.h>

using namespace std;

vector<bool> v;
vector<vector<int>> g;

void bfsTraversal(int b)
{
   //Declare a queue to store all the nodes connected to b
   queue<int> q;

   //Insert b to queue
   q.push(b);

   //mark b as visited
   v[b] = true;

   cout << "\n\nThe BFS Traversal is:  ";

   while (!q.empty())
   {
      int a = q.front();
      q.pop(); //delete the first element form queue

      for (auto j = g[a].begin(); j != g[a].end(); j++)
      {
            if (!v[*j])
            {
                v[*j] = true;
                q.push(*j);
            }
        }
        cout << a << "  ";
    }
}

void makeEdge(int a, int b)
{
    g[a].push_back(b); //an edge from a to b (directed graph)
}

int main()
{
    cout << "\n\nWelcome to Studytonight :-)\n\n\n";
    cout << " =====  Program to demonstrate the Breadth First Search Algorithm, in CPP  ===== \n\n";

    cout << " =====  Note; The vertices are numbered from 0 to n-1.  ===== \n\n";

    int n, e;

    cout << "Enter the number of vertices: ";

    cin >> n;

    cout << "\n\nEnter the number of edges: ";
    cin >> e;

    v.assign(n, false);
    g.assign(n, vector<int>());

    int a, b, i;

    cout << "Enter the edges with source and target vetex: \n ";

    for (i = 0; i < e; i++)
    {
        cin >> a >> b;
        makeEdge(a, b);
    }

    for (i = 0; i < n; i++)
    {
        //if the node i is unvisited
        if (!v[i])
        {
            bfsTraversal(i);
        }
    }

    cout << "\n\n\n";

    return 0;
}
```