Competitive Programming

(https://github.com/MalavikaJayakumar/Competitive-Programming-Problems)

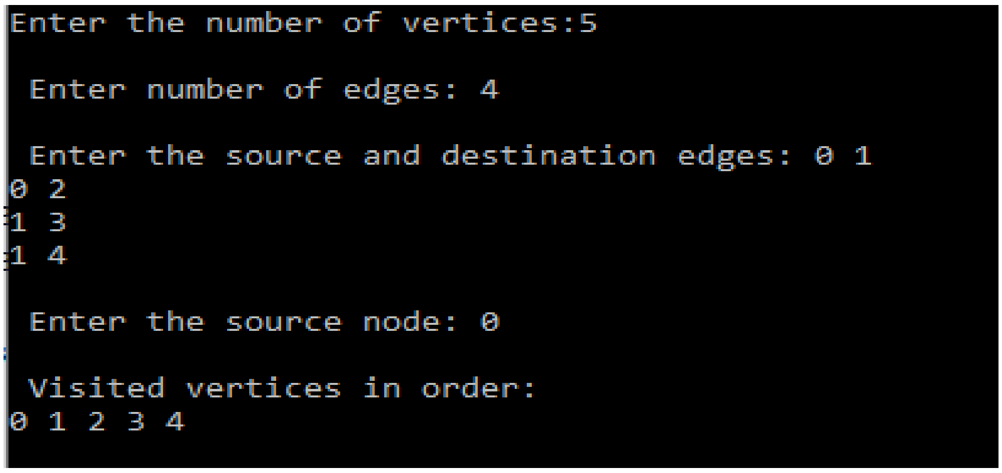
1. **BFS**
   a. **Print the nodes in the order they are visited**

Code:

```
#include <iostream>
#include <list>

using namespace std;

class Graph
{
    int numVertices;
    list <int> *adjLists;
    bool* visited;
public:
    Graph(int vertices);
    void addEdge(int e);
    void BFS(int startVertex);
};

Graph::Graph(int vertices)
{
    numVertices = vertices;
    adjLists = new list<int>[vertices];
}

void Graph::addEdge(int e)
{
          int src,dest;
          cout<<"\n Enter the source and destination edges: ";
          for(int i=0;i<e;i++)
          {
                 cin>>src>>dest;
                 adjLists[src].push_back(dest);
                 adjLists[dest].push_back(src);
          }
}

void Graph::BFS(int startVertex)
{
    visited = new bool[numVertices];
    for(int i = 0; i < numVertices; i++)
        visited[i] = false;

    list <int>queue;
```

```
        visited[startVertex] = true;
        queue.push_back(startVertex);

        list<int>::iterator i;

              cout<<"\n Visited vertices in order: \n";
        while(!queue.empty())
        {
            int currVertex = queue.front();
            cout << currVertex << " ";
            queue.pop_front();

            for(i = adjLists[currVertex].begin(); i != adjLists[currVertex].end();++i)
            {
                int adjVertex = *i;
                if(!visited[adjVertex])
                {
                    visited[adjVertex] = true;
                    queue.push_back(adjVertex);
                }
            }
        }
    }

    int main()
    {
              int n,e,s;
              cout<<"Enter the number of vertices:";
              cin>>n;
        Graph g(n);
        cout<<"\n Enter number of edges: ";
        cin>>e;
        g.addEdge(e);
        cout<<"\n Enter the source node: ";
        cin>>s;
        g.BFS(s);

        return 0;
    }
```

Output:



```
Enter the number of vertices:5

 Enter number of edges: 4

 Enter the source and destination edges: 0 1
0 2
1 3
1 4

 Enter the source node: 0

 Visited vertices in order:
0 1 2 3 4
```