```
using System;
using System.Collections.Generic;
using System.Linq;
Using System.Text;
Using System.Threading.Tasks;


Namespace AlgoPracticeCSharp.Graphs
{
   //Directed graph- Adjacency list representation
   class clsGraph
   {
      //total no of vertices
      private int Vertices;
      //adjency list array for all vertices.
      private List<Int32>[] adj;
      /* Example : vertices=4
       *      0->[1,2]
       *      1->[2]
       *      2->[0,3]
       *      3->[]
       */

      //constructor
      public clsGraph(int v)
      {
         Vertices = v;
         adj = new List<Int32>[v];
         //Instantiate adjacecny list for all vertices
         for (int i = 0; i < v; i++)
         {
            adj[i] = new List<Int32>();
         }

      }
```

```
//Add edge from v->w
public void AddEdge(int v, int w)
{
    adj[v].Add(w);
}

//Print BFS traversal
//s-> start node
//BFS uses queue as a base.
void BFS(int s)
{
    bool[] visited = new bool[Vertices];

    //create queue for BFS
    Queue<int> queue = new Queue<int>();
    visited[s] = true;
    queue.Enqueue(s);

    //loop through all nodes in queue
    while (queue.Count != 0)
    {
        //Deque a vertex from queue and print it.
        s = queue.Dequeue();
        Console.WriteLine("next->" + s);

        //Get all adjacent vertices of s
        foreach (Int32 next in adj[s])
        {
            if (!visited[next])
            {
                visited[next] = true;
                queue.Enqueue(next);
            }
        }
```

```
    }
}

//DFS traversal
// DFS uses stack as a base.
public void DFS(int s)
{
    bool[] visited = new bool[Vertices];

    //For DFS use stack
    Stack<int> stack = new Stack<int>();
    visited[s] = true;
    stack.Push(s);

    while (stack.Count != 0)
    {
        s = stack.Pop();
        Console.WriteLine("next->" + s);
        foreach (int i in adj[s])
        {
            if (!visited[i])
            {
                visited[i] = true;
                stack.Push(i);
            }
        }
    }
}

public void PrintAdjacecnyMatrix()
{
    for (int i = 0; i < Vertices; i++)
    {
        Console.Write(i + ":[");
```

```
            string s = "";
            foreach (var k in adj[i])
            {
                s = s + (k + ",");
            }
            s = s.Substring(0, s.Length - 1);
            s = s + "]";
            Console.Write(s);
            Console.WriteLine();
        }
    }
    //Calling methods
    public static void Main()
    {
        clsGraph graph = new clsGraph(4);
        graph.AddEdge(0, 1);
        graph.AddEdge(0, 2);
        graph.AddEdge(1, 2);
        graph.AddEdge(2, 0);
        graph.AddEdge(2, 3);
        graph.AddEdge(3, 3);
        //Print adjacency matrix
        graph.PrintAdjacecnyMatrix();

        Console.WriteLine("BFS traversal starting from vertex 2:");
        graph.BFS(2);
        Console.WriteLine("DFS traversal starting from vertex 2:");
        graph.DFS(2);
    }
  }
}
```