Bahria University,

Karachi Campus

A picture containing text, room

Description automatically generated

LAB EXPERIMENT NO.

**11**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
|  | **Graphs** |
| 1 | Create a program to implement Graphs With Adjacency matrix |
| 2 | Create a program to implement Graphs With Adjacency list |
| 3 | Create a program to implement BFS. |
| 4 | Create a program to implement DFS. |

Submitted On:

(Date: DD/MM/YY)

**TASK NO 1:** Create a program to implement Graphs With Adjacency matrix

**SOLUTION:**

namespace GRAPH

{

class Program

{

public void printAdjMatrix(int[,] AdjencyMatrix, int vertex)

{

Console.WriteLine("--------------Adjency Matrix-----------");

for (int i = 0; i < vertex; i++)

{

for (int j = 0; j < vertex; j++)

{

Console.Write("\t" + AdjencyMatrix[i, j] + " ");

}

Console.WriteLine();

}

}

public void createadjanceymatrix(int vertex) {

int[,] a = new int[vertex,vertex];

for (int i = 0; i <vertex; i++)

{

for (int j = 0; j < vertex; j++)

{

Console.Write(a[i,j]+"\t");

}

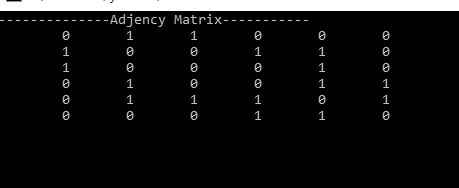
Console.WriteLine();

}

}

static void Main(string[] args)

**OUTPUT**:



{

Program p = new Program();

int[,] matrix = {

{0,1,1,0,0,0},

{1,0,0,1,1,0},

{1,0,0,0,1,0},

{0,1,0,0,1,1},

{0,1,1,1,0,1},

{0,0,0,1,1,0}};

p.printAdjMatrix(matrix, 6);

**TASK NO 2:** Create a program to implement Graphs With Adjacency list

**SOLUTION:**

private LinkedList<int>[] adjlist = new LinkedList<int>[6];

public void create\_adjancylist() {

for (int i = 0; i < 6; i++)

{

adjlist[i] = new LinkedList<int>();

}

}

public void printList()

{

for (int i = 0; i < 6; i++)

{

Console.Write(i + " |");

foreach (int item in adjlist[i])

{

Console.Write(" -> ");

Console.Write(item);

}

Console.WriteLine();

}

}

void addedge(int SOURCES,int destination) {

adjlist[SOURCES].AddLast(destination);

}

static void Main(string[] args)

{

Program p = new Program();

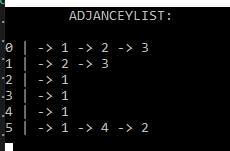
Console.WriteLine("\tADJANCEYLIST:\n");

p.create\_adjancylist();

p.addedge(0, 1);

p.addedge(0, 2);

**OUTPUT:**

****

p.addedge(0, 3);

p.addedge(1, 2);

p.addedge(1, 3);

p.addedge(2, 1);

p.addedge(3, 1);

p.addedge(4, 1);

p.addedge(5, 1);

p.addedge(5, 4);

p.addedge(5, 2);

p.printList();

Console.ReadLine();

**TASK NO 3:** Create a program to implement BFS.

**SOLUTION:**

private int \_V;

//Adjacency Lists

LinkedList<int>[] \_adj;

public Graph(int V)

{

\_adj = new LinkedList<int>[V];

for (int i = 0; i < \_adj.Length; i++)

{

\_adj[i] = new LinkedList<int>();

}

\_V = V;

}

// Function to add an edge into the graph

public void AddEdge(int v, int w)

{

\_adj[v].AddLast(w);

}

// Prints BFS traversal from a given source s

public void BFS(int s)

{

for (int i = 0; i < \_V; i++)

visited[i] = false;

LinkedList<int> queue = new LinkedList<int>();

visited[s] = true;

queue.AddLast(s);

while (queue.Any())

{

Console.Write(s + " ");

queue.RemoveFirst();

LinkedList<int> list = \_adj[s];

foreach (var val in list)

{

if (!visited[val])

{

visited[val] = true;

queue.AddLast(val);

}

}

}

}

class Program

{

static void Main(string[] args)

{

Graph g = new Graph(4);

g.AddEdge(0, 1);

g.AddEdge(0, 2);

g.AddEdge(1, 2);

g.AddEdge(2, 0);

g.AddEdge(2, 3);

g.AddEdge(3, 3);

Console.Write("Following is Breadth First " +

"Traversal(starting from " +

"vertex 2)\n");

g.BFS(1);

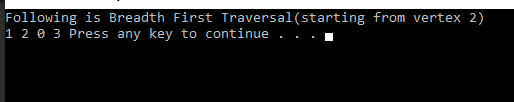
}

}

}

}

**OUTPUT:**

****

**TASK NO 4** Create a program to implement DFS

**SOLUTION:**

class Graph

{

private int V; // No. of vertices

private List<int>[] adj;

// Constructor

Graph(int v)

{

V = v;

adj = new List<int>[v];

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

adj[i] = new List<int>();

}

// Function to Add an edge into the graph

void AddEdge(int v, int w)

{

adj[v].Add(w); // Add w to v's list.

}

// A function used by DFS

void DFSUtil(int v, bool[] visited)

{

visited[v] = true;

Console.Write(v + " ");

List<int> vList = adj[v];

foreach (var n in vList)

{

if (!visited[n])

DFSUtil(n, visited);

}

}

void DFS(int v)

{

bool[] visited = new bool[V];

DFSUtil(v, visited);

}

static void Main(string[] args)

{

Graph g = new Graph(4);

g.AddEdge(0, 1);

g.AddEdge(0, 2);

g.AddEdge(1, 2);

g.AddEdge(2, 0);

g.AddEdge(2, 3);

g.AddEdge(3, 3);

Console.WriteLine(

"Following is Depth First Traversal "

+ "(starting from vertex 2)");

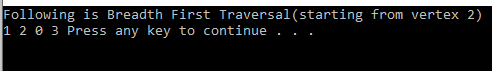
g.DFS(2);

Console.ReadKey();

}

}

**OUTPUT:**

****