Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

\_\_\_11\_\_\_

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 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:

\_\_\_\_27/1/2022\_\_\_\_\_\_\_\_

(Date: DD/MM/YY)

**GRAPHS**

**Task 1:** Create a program to implement graphs with adjacency matrix.

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp14

{

class Graph

{

public bool[,] adjMatrix;

public int vertices;

public Graph(int vertices)

{

this.vertices = vertices;

adjMatrix = new bool[this.vertices, this.vertices];

}

public void AddEdge(int i, int j)

{

adjMatrix[i, j] = true;

adjMatrix[j, i] = true;

}

public void RemoveEdge(int i, int j)

{

adjMatrix[i, j] = false;

adjMatrix[j, i] = false;

}

public void Display()

{

StringBuilder s = new StringBuilder();

for (int i = 0; i < this.vertices; i++)

{

s.Append(i + ": ");

for (int j = 0; j < this.vertices; j++)

{

s.Append((adjMatrix[i, j] ? 1 : 0) + " ");

}

s.Append("\n");

}

Console.WriteLine(s);

}

}

class Program

{

static void Main(string[] args)

{

Graph graph = new Graph(5);

graph.AddEdge(0, 1);

graph.AddEdge(0, 2);

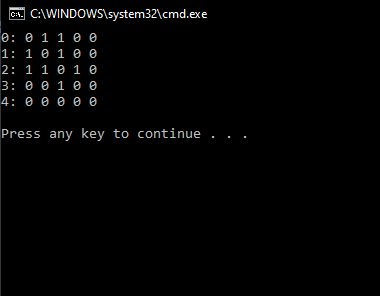
graph.AddEdge(1, 2);

graph.AddEdge(2, 3);

graph.AddEdge(2, 0);

graph.Display(); } } }

**Output:**



**Task 2:** Create a program to implement graphs with adjacency list.

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp14

{

class Graph

{

LinkedList<int>[] adj;

public Graph(int vertices)

{

adj = new LinkedList<int>[vertices];

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

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

}

public void AddEdge(int i, int j)

{

adj[i].AddLast(j);

adj[j].AddLast(i);

}

public void RemoveEdge(int i, int j)

{

adj[i].Remove(j);

adj[j].Remove(i);

}

public void Display()

{

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

{

Console.WriteLine("\nAdjacency list of vertex "

+ i);

Console.Write("head");

foreach (var item in adj[i])

{

Console.Write(" -> " + item);

}

Console.WriteLine();

}

}

}

class Program

{

static void Main(string[] args)

{

Graph graph = new Graph(5);

graph.AddEdge(0, 1);

graph.AddEdge(0, 2);

graph.AddEdge(1, 2);

graph.AddEdge(2, 3);

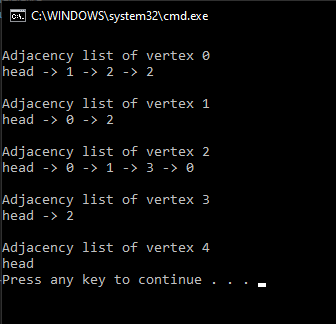
graph.AddEdge(2, 0);

graph.Display();

}

}

}



**Output:**

**Task 3:** Create a program to implement BFS.

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp14

{

class Graph

{

public LinkedList<int>[] adj;

public int vertices;

public Graph(int vertices)

{

this.vertices = vertices;

adj = new LinkedList<int>[this.vertices];

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

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

}

public void AddEdge(int i, int j)

{

adj[i].AddLast(j);

adj[j].AddLast(i);

}

public void RemoveEdge(int i, int j)

{

adj[i].Remove(j);

adj[j].Remove(i);

}

public void Display()

{

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

{

Console.WriteLine("\nAdjacency list of vertex " + i);

Console.Write("head");

foreach (var item in adj[i])

{

Console.Write(" -> " + item);

}

Console.WriteLine();

}

}

public void BFTraversal(int s)

{

Console.WriteLine("\nBF Traversal");

int data = s;

bool[] visited = new bool[this.vertices];

for (int i = 0; i < this.vertices; i++)

visited[i] = false;

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

visited[s] = true;

queue.AddLast(s);

while (queue.Any())

{

s = queue.First();

Console.Write(s + " ");

queue.RemoveFirst();

LinkedList<int> list = adj[s];

foreach (var val in list)

{

if (!visited[val])

{

visited[val] = true;

queue.AddLast(val);

}

}

}

Console.WriteLine("\n");

}

}

class Program

{

static void Main(string[] args)

{

Graph graph = new Graph(5);

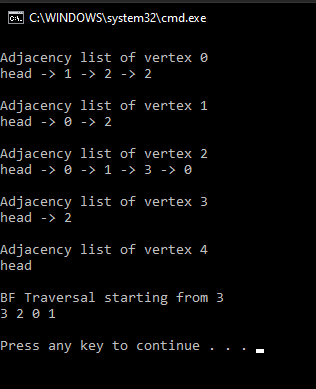
graph.AddEdge(0, 1);

graph.AddEdge(0, 2);

graph.AddEdge(1, 2);

graph.AddEdge(2, 3);

graph.AddEdge(2, 0);

 graph.Display();

graph.BFTraversal(0);

}

}

}

**Output:**

**Task 4:** Create a program to implement DFS.

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp14

{

class Graph

{

public LinkedList<int>[] adj;

public int vertices;

public Graph(int vertices)

{

this.vertices = vertices;

adj = new LinkedList<int>[this.vertices];

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

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

}

public void AddEdge(int i, int j)

{

adj[i].AddLast(j);

adj[j].AddLast(i);

}

public void RemoveEdge(int i, int j)

{

adj[i].Remove(j);

adj[j].Remove(i);

}

public void Display()

{

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

{

Console.WriteLine("\nAdjacency list of vertex " + i);

Console.Write("head");

foreach (var item in adj[i])

{

Console.Write(" -> " + item);

}

Console.WriteLine();

}

}

public void DFTraversal(int v)

{

Console.WriteLine("\nDF Traversal from " + v);

bool[] visited = new bool[this.vertices];

DFSUtil(v, visited);

}

void DFSUtil(int v, bool[] visited)

{

visited[v] = true;

Console.Write(v + " ");

LinkedList<int> vList = adj[v];

foreach (var n in vList)

{

if (!visited[n])

DFSUtil(n, visited);

}

}

}

class Program

{

static void Main(string[] args)

{

Graph graph = new Graph(5);

graph.AddEdge(0, 1);

graph.AddEdge(0, 2);

graph.AddEdge(1, 2);

graph.AddEdge(2, 3);

graph.AddEdge(2, 0);

graph.Display();

graph.DFTraversal(3);

}

}

}

**Output:**

