Міністерство освіти і науки України
Національний технічний університет України „КПІ”
Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки
інформації та управління

ЗВІТ

до лабораторної роботи № 1

з дисципліни “Основи Web-програмування”

Виконав студент *ІП-61 Каджая Володимир Миколайович*

(№ групи, прізвище, ім’я, по батькові)

Прийняв *Ліщук К. І.*

(посада, прізвище, ім’я, по батькові)

Київ 2018

ЗМІСТ

1. ПОСТАНОВКА ЗАДАЧІ

Имеется n городов, и для каждой пары известна стоимость соединения их дорогой (либо известно, что соединить их нельзя). Определить сеть автодорог, которая соединить все города так, чтобы можно было доехать из любого города в другой, а при этом стоимость прокладки дорог была бы максимальной

2. РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

```
Output
0 0 0 1 0
0 0 0 1 0
0 0 1 1 0
0 0 0 0 0
LONGEST 16
 0, 1, 2, 1, 15,
5, 6, 3, 1, 14,
7, 4, 1, 1, 13,
8, 9, 10, 11, 12,
```

3. КОД ПРОГРАМИ

```
using System;
using System.Collections.Generic;

namespace task1
{
public class Node
{
        public int r,c;
        public int data;
        public Node[] neighbors;

        public Node(int r, int c, int data)
        {
                this.r = r;
                this.c = c;
                this.data = data;
                neighbors = new Node[8];
        }
}

public class Map
{
        public static int[] xDir = {1,1,1,0,-1,-1,-1,0};
        public static int[] yDir = {-1,0,1,1,1,0,-1,-1};

        public Node[,] map;
```

```
public int[,] mapData;

private int width,height;
private int[,] marks;
public Map(int[,] mapData,int width,int height)
{
	this.mapData = mapData;
	this.width = width;
	this.height = height;

	createNodes();
	initNeighbors();
}
//INFO create nodes
private void createNodes()
{
	map = new Node[height,width];
	for(int i = 0; i < height; i++ )
	{
		for(int j = 0; j < width; j++)
		{
			map[i,j] = new Node(i,j,mapData[i,j]);
		}
	}
}
//INFO assign neighbor nodes
private void initNeighbors()
```

```
        {
            for(int i = 0; i < height; i++ )
            {
                for(int j = 0; j < width; j++)
                {
                    for(int k = 0; k < 8; k++)
                    {
                        if(inRange(i+yDir[k],j+xDir[k]))
                        {
                            map[i,j].neighbors[k] =
map[i+yDir[k],j+xDir[k]];
                        }
                    }
                }
            }
        }

        private bool inRange(int r, int c)
        {
            return r < height && r >= 0 && c < width && c >= 0;
        }

        public List<Node> longestPath()
        {
            marks = new int[height,width];
            List<Node> nodes = new List<Node>();
            int c = dfs(map[0,0],nodes);
```

```
            //INFO Iterasions count
            //Console.WriteLine("COUNT " + c);
            return nodes;
        }

        private int dfs(Node node, List<Node> nodes)
        {
            int i = 1;
            List<Node> longest = new List<Node>();
            List<Node> list = null;
            marks[node.r,node.c] = 1;

            for(int n = 0; n < 8; n++)
            {
                //INFO if the neighbor node is not null and same type with parent node do dfs for neighbor node
                if(node.neighbors[n] != null &&
                    marks[node.neighbors[n].r,node.neighbors[n].c] == 0 &&
                    node.neighbors[n].data == node.data)
                {
                    list = new List<Node>();
                    i += dfs(node.neighbors[n],list);
                    //INFO if the found nodes count is more than previous nodes count set new nodes to best nodes
                    if(list.Count > longest.Count)
```

```
                                longest = list;
                        }
                }

                marks[node.r,node.c] = 0;
                longest.Add(node);
                nodes.AddRange(longest);

                return i;
        }
}

public class MainClass
{
        public static void Main (string[] args)
        {
                Console.WriteLine ("Hello World!");

                Console.WriteLine ("Enter h");
                int num = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine (num);

                Console.WriteLine ("Enter w");
                int num2 = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine (num2);

                Console.WriteLine ("Enter array");
```

```
int[,] matrix1 = new int[2, 2];

int[,] data = new int[num, num2];
for (int i = 0; i < num; i++) {
        for (int j = 0; j < num2; j++) {
                data [i, j] = Int32.Parse(Console.ReadLine());
        }
}

Console.WriteLine ("Output");
for (int i = 0; i < num; i++) {
        for (int j = 0; j < num2; j++) {
                Console.Write(data[i, j] + " ");
        }
        Console.WriteLine ();
}

/*const int w = 5;
const int h = 4;
int[,] data = new int[h,w]{
        {0,0,0,1,0},
        {0,0,0,1,0},
        {0,0,1,1,0},
        {0,0,0,0,0}};*/
        /*{0,0,0,0,1},
        {0,0,0,1,0},
```

```
            {0,0,1,0,0},
            {0,1,0,0,0}};*/

        Map map = new Map(data,num2,num);

        List<Node> longest = map.longestPath();
        Console.WriteLine("LONGEST " + longest.Count + " ");

        //INFO this doing for printing array with steps
        int c = longest.Count-1;
        foreach(Node n in longest)
            data[n.r,n.c] = c--;

        string st = " ";
        for(int i = 0; i < num; i++ )
        {
            for(int j = 0; j < num2; j++)
            {
                st += data[i,j] + ", ";
            }
            st += "\n";
        }

        Console.WriteLine(st);
    }

}
```

}