Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації та управління

ЗВІТ

про виконання лабораторного практикуму №8(1)

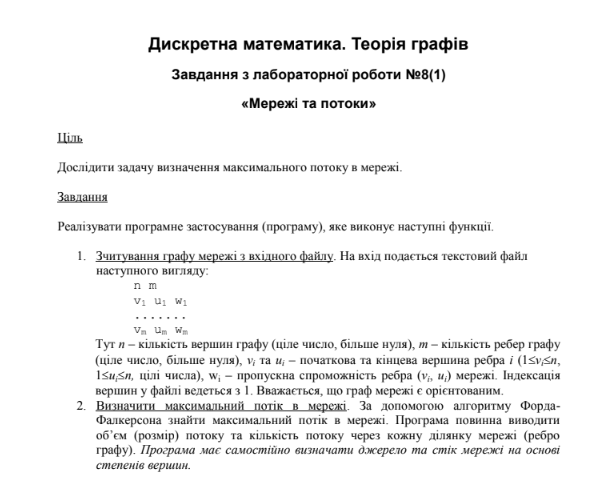
Виконав:

студент 1- го курсу ФІОТ

групи *ІП-91*

*Кінчур Вадим Вікторович*

Київ 2020

1. Умова лабораторної роботи
2. Програмний код (C#):

using System.IO;

using System.Linq;

using System.Collections.Generic;

namespace LAB\_8\_1\_

{

class Program

{

static void Main(string[] args)

{

var adjM = GetInput("input.txt");

(var maxFlow, var newAdj) = ProcessFulkerson(adjM);

PrintFulkerson(maxFlow, newAdj, adjM);

}

static int[,] GetInput(string fileName)

{

string path = "../../../" + fileName;

using (var sr = new StreamReader(path))

{

var fisrstRow = sr.ReadLine().Split().ToList().ConvertAll(s => int.Parse(s));

(var vNum, var eNum) = (fisrstRow[0], fisrstRow[1]);

var adjM = new int[vNum, vNum];

for (var i = 0; i < vNum; ++i)

for (var j = 0; j < vNum; ++j)

adjM[i, j] = -1;

for(var i = 0; i < eNum; ++i)

{

(var start, var end, var flow) = ParseRow(sr.ReadLine());

adjM[start - 1, end - 1] = flow;

}

return adjM;

}

}

static (int,int, int) ParseRow(string row)

{

var nums = row.Split().ToList().ConvertAll(s => int.Parse(s));

return (nums[0], nums[1], nums[2]);

}

static (int, int) GetSourceAndSick(int[,] adjM)

{

var size = adjM.GetLength(0);

var inDegree = new int[size];

var outDegree = new int[size];

(var sick, var source) = (0, 0);

for(var i = 0; i < size; ++i)

{

for(var j = 0; j < size; ++j)

{

if (adjM[i, j] != 0)

{

++outDegree[i];

++inDegree[j];

}

source = (inDegree[j] == 0) ? j : (inDegree[i] == 0) ? i : source;

sick = (outDegree[i] == 0) ? i : (outDegree[j] == 0) ? j : sick;

}

}

return (source, sick);

}

static bool BFS(int[,] adjM, int source, int sick, ref int[] parent)

{

var visited = Enumerable.Repeat(false, adjM.Length).ToArray();

var queue = new Queue<int>();

visited[source] = true; queue.Enqueue(source);

while(queue.Count != 0)

{

var u = queue.Dequeue();

for(var v = 0; v < adjM.GetLength(0); ++v)

{

if (!visited[v] && adjM[u, v] > 0)

{

queue.Enqueue(v);

visited[v] = true;

parent[v] = u;

}

}

}

return (visited[sick]);

}

static (int, int[,]) ProcessFulkerson(int[,] adjM)

{

(var source, var sick) = GetSourceAndSick(adjM);

var parent = Enumerable.Repeat(-1, adjM.Length).ToArray();

var changeAdjM = adjM.Clone() as int[,];

var maxFlow = 0;

while(BFS(changeAdjM, source, sick, ref parent))

{

var currentFlow = int.MaxValue;

var start = sick;

while(start != source)

{

currentFlow = System.Math.Min(currentFlow, changeAdjM[parent[start], start]);

start = parent[start];

}

maxFlow += currentFlow;

var v = sick;

while(v != source)

{

var u = parent[v];

changeAdjM[u, v] -= currentFlow;

changeAdjM[v, u] += currentFlow;

v = parent[v];

}

}

return (maxFlow, changeAdjM);

}

static void PrintFulkerson(int maxFlow, int[,] newadjM, int [,] adjM)

{

System.Console.WriteLine($"Maximal flow is equal to {maxFlow}");

System.Console.WriteLine("Flow through each edge:");

for(var i = 0; i < adjM.GetLength(0); ++i)

{

for(var j = 0; j < adjM.GetLength(0); ++j)

{

if (adjM[i, j] != -1)

{

var flow = (newadjM[i, j] < 0) ? adjM[i, j]

: adjM[i, j] - newadjM[i, j];

System.Console.WriteLine($"f({i + 1}, {j + 1}) = {flow}");

}

}

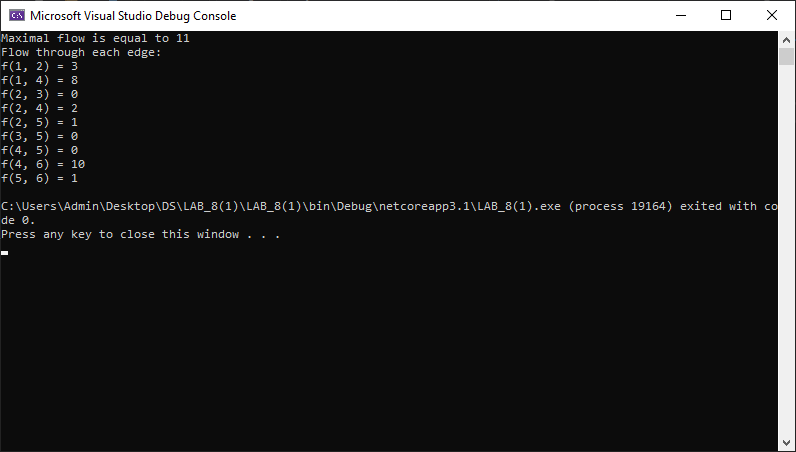
}

}

}

}

1. Результат виконання:

**Вхідні дані:**

6 9

1 2 3

1 4 8

2 3 7

2 4 5

2 5 3

3 5 10

4 5 5

4 6 10

5 6 10