Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Реферат на тему:

Определениемаксимальных потоков в сети

Выполнил: студент 3 курса 341 группы  
направления 010500.62 Математическое обеспечение и администрирование информационных систем (профиль Параллельное программирование)

факультета компьютерных наук и информационных технологий

Акимов Артемий Андреевич

Саратов 2015

Содержание

[Содержание 2](#_Toc438144177)

[ВВЕДЕНИЕ 3](#_Toc438144178)

[1 История задачи о максимальном потоке 4](#_Toc438144179)

[2 Математическая модель в виде графа 5](#_Toc438144180)

[2.1 Представление сети 5](#_Toc438144181)

[2.2 Сети и потоки 5](#_Toc438144182)

[2.3 Пример сети 7](#_Toc438144183)

[3 Постановка задачи 8](#_Toc438144184)

[4 Алгоритм Эдмондса-Карпа 9](#_Toc438144185)

[4.1 Описание 9](#_Toc438144186)

[4.2 Применение алгоритма на примере 10](#_Toc438144187)

[5 Представление графа в памяти компьютера 13](#_Toc438144188)

[6 Визуализация алгоритма 14](#_Toc438144189)

[ЗАКЛЮЧЕНИЕ 16](#_Toc438144190)

[ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc438144191)

[Приложение А РЕАЛИЗАЦИЯ НАХОЖДЕНИЯ МАКСИМАЛЬНОГО ПОТОКА В СЕТИ 18](#_Toc438144192)

# ВВЕДЕНИЕ

Теория графов широко применима в реальной жизни. Существует множество бытовых задач, которые решаются с помощью неё. Одной из таких задач является «определение максимального потока в сети».

Целью является решение задачи о нахождении максимального потока.

Данная цель предполагает решение следующих основных задач:

1. Выбор алгоритма для решения задачи (в нашем случае это алгоритм Эдмондса-Карпа);
2. Графическое представления графа;

# История задачи о максимальном потоке

После вступления США во Вторую мировую войну в 1941 году математик Джордж Бернард Данциг поступил на работу в отдел статистического управления Военно-воздушных сил США в Вашингтоне. С 1941 по 1946 годы он возглавлял подразделение анализа военных действий, где работал над различными математическими проблемами. Впоследствии c использованием работы Данцига задача о максимальном потоке была впервые решена в ходе подготовки воздушного моста во время блокады Западного Берлина, происходившей в 1948—1949 году.

В 1951 году Джордж Данциг впервые сформулировал задачу в общем виде.

В 1955 году, Лестер Форд и Делберт Фалкерсон впервые построили алгоритм, специально предназначенный для решения этой задачи. Их алгоритм получил название алгоритм Форда-Фалкерсона. В дальнейшем решение задачи много раз улучшалось.

Впервые был опубликован алгоритм Эдмондса-Карпа в [1970 году](https://ru.wikipedia.org/wiki/1970_%D0%B3%D0%BE%D0%B4) советским учёным [Е. А. Диницом](https://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B8%D0%BD%D0%B8%D1%86,_%D0%95%D1%84%D0%B8%D0%BC_%D0%90%D0%B1%D1%80%D0%B0%D0%BC%D0%BE%D0%B2%D0%B8%D1%87&action=edit&redlink=1). Позже, в [1972 году](https://ru.wikipedia.org/wiki/1972_%D0%B3%D0%BE%D0%B4), был независимо открыт Эдмондсом и [Карпом](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BF,_%D0%A0%D0%B8%D1%87%D0%B0%D1%80%D0%B4).

В 2010 году исследователи Джонатан Кёлнер и Александер Мондры из МТИ вместе со своими коллегами Дэниелем Спилманом из Йельского университета и Шень-Хуа Тенем из Южно-Калифорнийского университета продемонстрировали очередное улучшение алгоритма, впервые за 10 лет.[2]

# Математическая модель в виде графа

## Представление сети

Рассмотрим ориентированный Граф. Будем рассматривать его как сеть связей, по которым информация движется от истока (где оно производится с некоторой постоянной скоростью) к стоку (где оно потребляется — с той же скоростью).

Как и в задаче о кратчайших путях, на каждом ребре графа мы пишем число. Но если раньше это число означало длину пути, то теперь это скорее пропускная способность связи — максимальная скорость потока в этой связи. Мы считаем, что в вершинах данные не накапливаются — сколько приходит, столько и уходит (если вершина не является истоком или стоком). Это свойство называется «законом сохранения потока». Для электрического тока это свойство называется первым правилом Кирхгофа (Kirchhoff's Current Law).

А задача о максимальном потоке для данной сети полностью совпадает с задачей, поставленной ранее.

## Сети и потоки

Назовем сетью ориентированный граф , каждому ребру которого поставлено в соответствие число , называемое пропускной способностью ребра. В случае мы полагаем *.* В графе выделены две вершины: исток и сток. Для простоты мы предполагаем, что в графе нет «бесполезных» вершин (каждая вершина лежит на каком-то пути . из истока в сток) (в таком случае граф связен).

Теперь дадим определение потока. Пусть дана сеть *.* Пропускная способность которой задаётся функцией. Сеть имеет исток и сток. Потоком в сети назовем функцию*,* обладающую следующими тремя свойствами.

**Ограничение, связанное с пропускной способностью**:

для всех .

**Кососимметричность:** для всех .

**Сохранение потока:** *.*

Величинаможет быть как положительной, так и отрицательной. Она определяет, сколько информации движется из вершины в вершину (отрицательные значения соответствуют движению в обратную сторону).

**Величина** потока определяется как сумма  (складываем потоки по всем рёбрам, выходящим из истока). Она обозначается  
 (не спутайте с абсолютной величиной числа!). Задача о максимальном потоке состоит в следующем: для данной сети с истоком и стоком найти поток максимальной величины.

Поясним смысл трёх наших свойств. Первое означает, что поток из одной вершины в другую не превышает пропускной способности ребра. Второе представляет собой соглашение о том, что отрицательные числа соответствуют потоку в обратную сторону. Из него следует также, что для любой вершины и (положим ). Третье свойство означает, что для любой вершины и (кроме стока и истока) сумма потоков во все другие вершины равна нулю. Учитывая кососимметричность, это свойство можно переписать как

.

Заметим также, что если вершины и не соединены ребром, то поток  
между ними, т.е. равен нулю.

Действительно, если и , то . Тогда из первого свойства следует, что . Вспоминая, что (кососимметричность), мы видим, что *.*

Разделим объем информации, поступающий в данную вершину и объем информации, из неё выходящее (т.е. положительные и отрицательные значения )*.* Сумму

назовем входящим (в вершину) потоком. Выходящий поток определяется симметрично. Теперь закон сохранения потока можно сформулировать так: для любой вершины, кроме истока и стока, входящий поток равен исходящему.[1]

## Пример сети

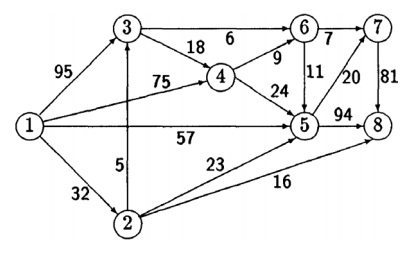


Рис. 1 Пример сети

# Постановка задачи

Дан файл с описанием графа. Каждая связь имеет вес – объем, который можно прокачать через данную дугу за единицу времени. Возможный вариант структуры файла приведён ниже (комментарии в реальном файле будут отсутствовать):

8 // число вершин

1 2(32) 3(95) 4(75) 5(57) //вершина и список смежных с указанием веса

2 3(5) 5(23) 8(16)

3 4(18) 6(6)

4 5(24) 6(9)

5 7(20) 8(94)

6 5(11) 7(7)

7 8(81)

# Алгоритм Эдмондса-Карпа

Алгоритм Эдмондса — Карпа — это вариант алгоритма Форда — Фалкерсона, при котором на каждом шаге выбирают кратчайший дополняющий путь из в остаточной сети (полагая, что каждое ребро имеет единичную длину). Кратчайший путь находится поиском в ширину.

## Описание

1. Обнуляем все потоки. Остаточная сеть изначально совпадает с исходной сетью.
2. В остаточной сети находим кратчайший путь из источника в сток. Если такого пути нет, останавливаемся.
3. Пускаем через найденный путь (он называется увеличивающим путём или увеличивающей цепью) максимально возможный поток:
   1. На найденном пути в остаточной сети ищем ребро с минимальной пропускной способностью c_\min.
   2. Для каждого ребра на найденном пути увеличиваем поток на c_\min, а в противоположном ему — уменьшаем на c_\min.
   3. Модифицируем остаточную сеть. Для всех рёбер на найденном пути, а также для противоположных им рёбер, вычисляем новую пропускную способность. Если она стала ненулевой, добавляем ребро к остаточной сети, а если обнулилась, стираем его.
4. Возвращаемся на шаг 2.

Чтобы найти кратчайший путь в графе, используем поиск в ширину:

1. Создаём очередь вершин О. Вначале О состоит из единственной вершины .
2. Отмечаем вершину как посещённую, без предка, а все остальные как непосещённые.
3. Пока очередь непуста, выполняем следующие шаги:
   1. Удаляем первую в очереди вершину .
   2. Для всех рёбер , исходящих из вершины , таких что вершина v ещё не посещена, выполняем следующие шаги:
4. Отмечаем вершину v как посещённую, с предком u.
5. Добавляем вершину v в конец очереди
6. Если, выходим из обоих циклов: мы нашли кратчайший путь.
7. Если очередь пуста, возвращаем ответ, что пути нет вообще и останавливаемся.
8. Если нет, идём от , каждый раз переходя к предку. Возвращаем путь в обратном порядке.

## Применение алгоритма на примере

Решение: с помощью алгоритма Эдмондса-Карпа найдем наибольший поток из 1 в 8 (см. рис 1).

Шаг 1. Выбираем путь 1-2-8. Его пропускная способность равна минимальной из всех пропускных способностей, входящих в него дуг, то есть 16. Уменьшаем пропускные способности увеличивающих дуг этого потока на 16, насыщенную дугу 2-8 вычеркиваем и создаем уменьшающие дуги на всем пути и увеличиваем их на 16.  Поток равен 16.

Шаг 2. Выбираем путь 1-5-8. Его пропускная способность равна минимальной из всех пропускных способностей, входящих в него дуг, то есть 57. Уменьшаем пропускные способности увеличивающих дуг этого потока на 57, насыщенную дугу 1-5 вычеркиваем и создаем уменьшающие дуги на всем пути и увеличиваем их на 57. Увеличиваем значение потока на 57, т.е. суммируем 16 и 57: 73.

Шаг 3. Выбираем путь 1-2-5-8. Его пропускная способность равна минимальной из всех пропускных способностей, входящих в него дуг, то есть 16. Уменьшаем пропускные способности увеличивающих дуг этого потока на 16, насыщенную дугу 1-2 вычеркиваем и создаем уменьшающие дуги на всем пути (если они не существовали ранее) и увеличиваем их на 16. Величина потока равна 89.

Шаг 4. Выбираем путь 1-4-5-8. Его пропускная способность равна минимальной из всех пропускных способностей, входящих в него дуг, то есть 21. Уменьшаем пропускные способности увеличивающих дуг этого потока на 21, насыщенную дугу 5-8 вычеркиваем и создаем уменьшающие дуги на всем пути (если они не существовали ранее) и увеличиваем их на 21. Величина потока равна 110.

Шаг 5. Выбираем путь 1-3-6-7-8. Его пропускная способность равна минимальной из всех пропускных способностей, входящих в него дуг, то есть 6. Уменьшаем пропускные способности увеличивающих дуг этого потока на 6, насыщенную дугу 3-6 вычеркиваем и создаем уменьшающие дуги на всем пути (если они не существовали ранее) и увеличиваем их на 6. Величина потока равна 116.

Шаг 6. Выбираем путь 1-4-5-7-8. Его пропускная способность равна минимальной из всех пропускных способностей, входящих в него дуг, то есть 3. Уменьшаем пропускные способности увеличивающих дуг этого потока на 3, насыщенную дугу 4-5 вычеркиваем и создаем уменьшающие дуги на всем пути (если они не существовали ранее) и увеличиваем их на 3. Величина потока равна 119.

Шаг 7. Выбираем путь 1-4-6-7-8. Его пропускная способность равна минимальной из всех пропускных способностей, входящих в него дуг, то есть 1. Уменьшаем пропускные способности увеличивающих дуг этого потока на 1, насыщенную дугу 6-7 вычеркиваем и создаем уменьшающие дуги на всем пути (если они не существовали ранее) и увеличиваем их на 1. Величина потока равна 120.

Шаг 8. Выбираем путь 1-4-6-5-7-8. Его пропускная способность равна минимальной из всех пропускных способностей, входящих в него дуг, то есть 8. Уменьшаем пропускные способности увеличивающих дуг этого потока на 8, насыщенную дугу 7-8 вычеркиваем и создаем уменьшающие дуги на всем пути (если они не существовали ранее) и увеличиваем их на 1. Величина потока равна 128.

Шаг 9. Новый путь из 1 в 8 не найден, следовательно, величина потока на шаге 8 является максимальным поток из 1 в 8.

# Представление графа в памяти компьютера

Для хранения графа я использовал словарь (dictionary) представляет собой сложную структуру данных, позволяющую обеспечить доступ к элементам по ключу. Главное свойство словарей — быстрый поиск на основе ключей. Можно также свободно добавлять и удалять элементы, подобно тому, как это делается в List<T>, но без накладных расходов производительности, связанных с необходимостью смещения последующих элементов в памяти. [4]

Ключом является элемент типа int, хранящий смежные вершины, а значение – координаты вершины, лист смежных вершин и соответствующие веса.

public Dictionary<int, List<Tuple<int, int>>> array1Weighted = new Dictionary<int, List<Tuple<int, int>>>();[5]

# Визуализация алгоритма

Windows Forms — интерфейс программирования приложений, отвечающий за графический интерфейс пользователя и являющийся частью Microsoft . NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде.

Для того чтобы построить схему сети, вычислим координаты по формулам:

где – номер вершины, – количество вершин, и – сдвиги по соответствующим осям.

На рис.2 изображена форма, на которой и представлена визуализация.

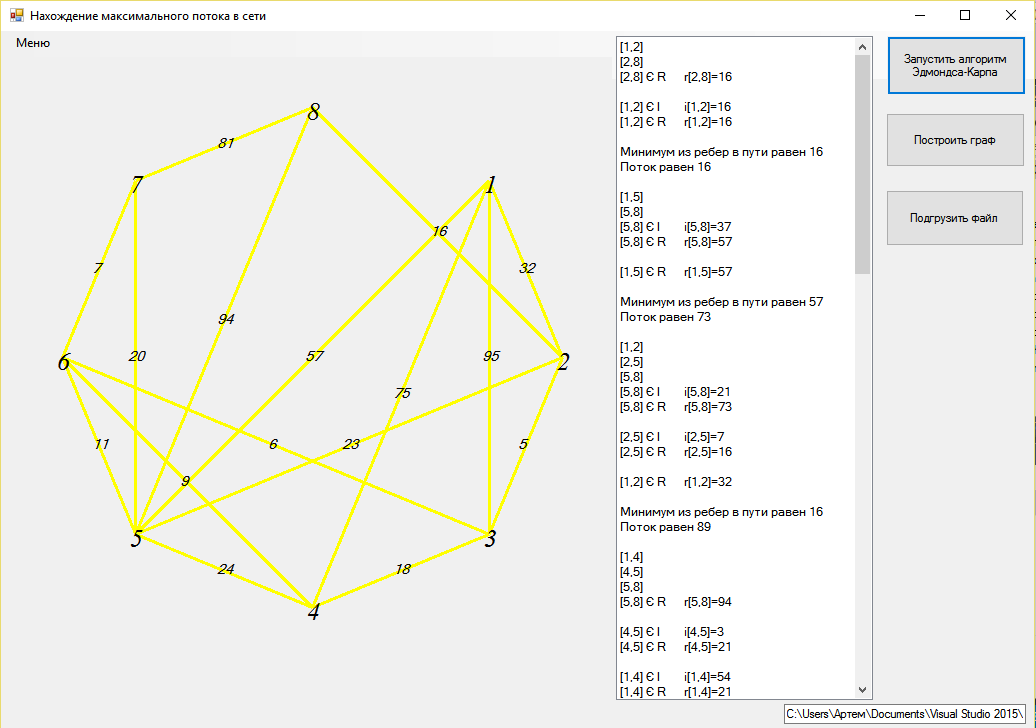


Рис. 2 Визуализация

При нажатии на кнопку “Подгрузить граф” пользователь может выбрать входной файл и в TextBox помещается адрес файла, при нажатии на кнопку “Построить граф” – происходит построение карты графа. Нажав на кнопку “Запустить алгоритм Эдмондса-Карпа” происходит заполнение ListBox данными с решением задачи на конкретном примере.

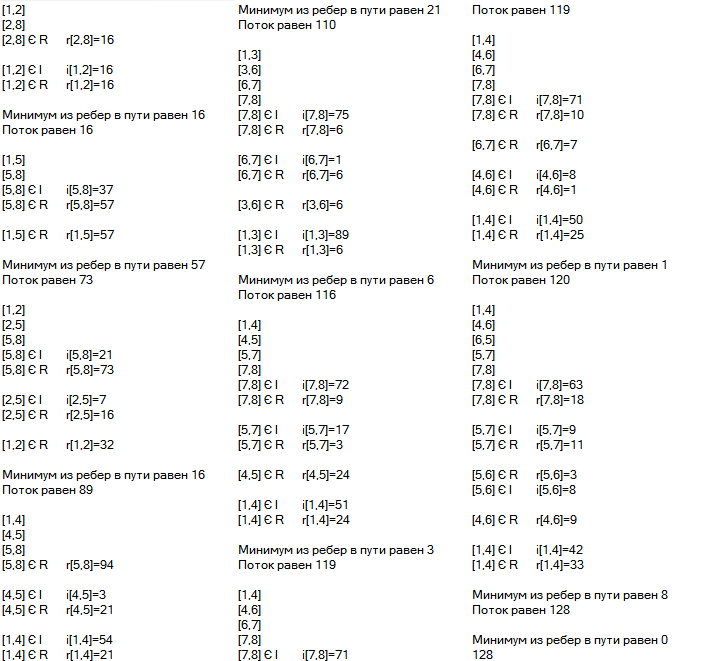


Рис. 3 Решение задачи

Данные, просчитанные при помощи алгоритма, можно сравнить с теоретическим решением задачи, представленным в пункте 3.2.

# ЗАКЛЮЧЕНИЕ

Была рассмотрена задача нахождения максимального потока в сети, ее конкретное решение при помощи алгоритма Эдмондса-Карпа. Также для наглядности работы алгоритма было построено решение задачи о построении графа.

# ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Томас Х. Кормен и др. Алгоритмы: построение и анализ = INTRODUCTION TO ALGORITHMS. — 2-е изд. — М.: «Вильямс», 2006. — С. 1296. — ISBN 0-07-013151-1.
2. http://www-groups.dcs.st-and.ac.uk/~history/Biographies/Dantzig\_George.html
3. http://ivdon.ru/magazine/archive/n2y2013/1666
4. http://professorweb.ru/my/csharp/charp\_theory/level12/12\_10.php
5. https://msdn.microsoft.com/ru-ru/library/xfhwa508(v=vs.110).aspx

# Приложение А РЕАЛИЗАЦИЯ НАХОЖДЕНИЯ МАКСИМАЛЬНОГО ПОТОКА В СЕТИ

Листинг Graph.cs

using System;

using System.IO;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace EdmondsKarp

{

public class Graph:Form

{

public Dictionary<int, Tuple<Tuple<double, double>, List<Tuple<int, int>>>> array1Weighted = new Dictionary<int, Tuple<Tuple<double, double>, List<Tuple<int, int>>>>();

// словрь, хранящий список координат: <номер вершины <<координата x, координата y>,<лист смежных вершин<номер смежной,вес>>>>

public double d = 300;

public double dx = 600 / 2;

public double dy = 600 / 2;

public double r = 250;

public List<int> way = new List<int>();

public List<int> p = new List<int>();//лист предков

public List<int> distance = new List<int>();//лист distance

public List<bool> u = new List<bool>();//лист used

int n;

//Граф представляется списком смежности в виде:

//5 - количество вершин

//1 2(3) - номер вершины смежная вершина(вес)

//....

//....

//....

//....

public Graph(string key)

{

string[] mas = File.ReadAllLines(key);

n = int.Parse(mas[0]);

string[] m;

for (int i = 1; i < n + 1; i++)

{

string[] mas2 = mas[i].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

List<Tuple<int, int>> matr = new List<Tuple<int, int>>();

for (int j = 1; j < mas2.Length; j++)

{

m = mas2[j].Split(new char[] { '(', ')' }, StringSplitOptions.RemoveEmptyEntries);

Tuple<int, int> pair = Tuple.Create(int.Parse(m[0]), int.Parse(m[1]));

matr.Add(pair);

}

//подсчет координат

double x = r \* Math.Cos(2 \* Math.PI \* i / n - Math.PI / 2) + dx;

double y = r \* Math.Sin(2 \* Math.PI \* i / n - Math.PI / 2) + dy;

Tuple<double, double> coordinates = new Tuple<double, double>(x, y);

Tuple<Tuple<double, double>, List<Tuple<int, int>>> vertex = new Tuple<Tuple<double, double>, List<Tuple<int, int>>>(coordinates, matr);

array1Weighted.Add(int.Parse(mas2[0]), vertex);

}

}

public int Flow(int[,] cap, int s, int t,ListBox listBox)

//реализует Алгоритм Эдмондса-Карпа с путем ввиде кратчайшего пути из бфс

//матрица смежности,вершина исток,вершина сток

{

for (int flow = 0; ;)//до тех пор, пока поток не равен нулю

{

List<int> way = new List<int>();

way = bfs(cap, s, t);

int df = FindPath(way, cap, new bool[cap.GetLength(0)], s, t, Int32.MaxValue,listBox);

listBox.Items.Add("Минимум из ребер в пути равен "+ df);

if (df == 0)

return flow;

listBox.Items.Add("Поток равен "+(flow += df));

listBox.Items.Add("\n");

}

}

public int FindPath(List<int> way, int[,] cap, bool[] vis, int u, int t, int f, ListBox listBox)

//(кратчайший путь по бфс,матрица смежности, патрица посещений,действующая вершина в пути от стока к истоку, сток, если сток и сток равны - бесконечный поток

//метод, который подсчитывает поток

{

Form1 number = new Form1();

if (u == t)

return f;

vis[u] = true;

foreach (var item in way)

if (!vis[item])

{

listBox.Items.Add("["+ (u + 1) + "," + (item + 1)+ "]");//вывод на экран дуги из кратчайшего пути

int df = FindPath(way, cap, vis, item, t, Math.Min(f, cap[u, item]),listBox);

//величина потока в между данной вершиной u и последующей вершиной из кратчайшего пути

if (df > 0)

{

cap[u, item] -= df;//увеличивающая дуга

cap[item, u] += df;//уменьшающая дуга

//вывод на экран r и i и принадлежности дуг к I и R(полезно для проверки)

if ((cap[u, item] != 0) && (u < item)) listBox.Items.Add("[" + (u + 1) + "," + (item + 1) + "] Є I i[" + (u + 1) + "," + (item + 1) + "]=" + (cap[u, item]) + " ");

else if (cap[u, item] != 0) listBox.Items.Add("[" + (item + 1) + "," + (u + 1) + "] Є R r[" + (item + 1) + "," + (u + 1) + "]=" + (cap[u, item]) + " ");

if ((cap[item, u] != 0) && (item > u)) listBox.Items.Add("[" + (u + 1) + "," + (item + 1) + "] Є R r[" + (u + 1) + "," + (item + 1) + "]=" + (cap[item, u]) + " ");

else if (cap[item, u] != 0) listBox.Items.Add("[" + (item + 1) + "," + (u + 1) + "] Є I i[" + (item + 1) + "," + (u + 1) + "]=" + (cap[item, u]) + " ");

listBox.Items.Add("\n");

return df;

}

}

return 0;

}

public List<int> bfs(int[,] cap, int s, int t)//поиск в ширину кратчайшего пути

{

Queue<int> q = new Queue<int>(); ;

q.Enqueue(s);

for (int i = 0; i < cap.GetLength(0); i++) { u.Add(false); }

for (int i = 0; i < cap.GetLength(0); i++) { p.Add(-1); }

for (int i = 0; i < cap.GetLength(0); i++) { distance.Add(0); }

u[s] = true;

while (q.Count != 0)

{

int v = q.Peek();

q.Dequeue();

if (v != -1)

for (int i = 0; i < cap.GetLength(0); i++)

{

if (cap[v, i] != 0)

{

int to = i;

if (!u[to])

{

u[to] = true;

distance[to] = distance[v] + 1;

q.Enqueue(to);

p[to] = v;

}

}

}

}

List<int> path = new List<int>();

if (!u[t])

Console.WriteLine("No path!");

else

{

for (int j = t; j != -1; j = p[j])

path.Add(j);

path.Reverse();

u.Clear();

return path;

}

u.Clear();

return path;

}

public void MaxFlow(int first, int second,ListBox listBox)//преобразуем список смежности с весами в матрицу смежности и запускаем Flow

{

int[,] MyArray = new int[n, n];

int[] R = new int[n];

int[] I = new int[n];

foreach (var item in array1Weighted)

{

foreach (var i in item.Value.Item2)

{

MyArray[item.Key - 1, i.Item1 - 1] = i.Item2;

}

}

Console.WriteLine();

int strange = Flow(MyArray, first - 1, second - 1,listBox);

listBox.Items.Add(strange);

}

}

}

Листинг Form1.cs

using System;

using System.Threading;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing.Drawing2D;

using System.IO;

namespace EdmondsKarp

{

public partial class Form1 : Form

{

public string k;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

Graph graph = new Graph(k);

graph.MaxFlow(1, 8,listBox1);

}

private void CMap\_Paint(object sender, PaintEventArgs e)

{

Graph alg = new Graph(k);

Graphics g = CMap.CreateGraphics();

Graph graph = new Graph(k);

int vertices = graph.array1Weighted.Count;

foreach (var item in graph.array1Weighted)

{

label(item.Key, item.Value.Item1.Item1, item.Value.Item1.Item2);

foreach (var i in item.Value.Item2)

{

Pen pen = new Pen(Color.Yellow, 3);

e.Graphics.DrawLine(pen, (float)item.Value.Item1.Item1, (float)item.Value.Item1.Item2, (float)graph.array1Weighted[i.Item1].Item1.Item1, (float)graph.array1Weighted[i.Item1].Item1.Item2);

weight(i.Item2, item.Value.Item1.Item1, item.Value.Item1.Item2, graph.array1Weighted[i.Item1].Item1.Item1, graph.array1Weighted[i.Item1].Item1.Item2);

}

}

}

private void label(int vertex,double x,double y)

{

Color myColor = Color.FromArgb(0, Color.White);

Label label = new Label();

label.BackColor = myColor;

label.Text = vertex.ToString();

label.Width = 40;

label.Height = 23;

label.Font = new Font("Times New Roman", 19, FontStyle.Italic);

label.Location = new Point(Convert.ToInt32(x - 10), Convert.ToInt32(y - 10));

CMap.Controls.Add(label);

}

private void weight(int vertex, double x1, double y1, double x2, double y2)

{

Color myColor = Color.FromArgb(0, Color.White);

Label label1 = new Label();

label1.BackColor = myColor;

label1.Location = new Point(Convert.ToInt32(Math.Abs(x1-10+x2-10)/2), Convert.ToInt32(Math.Abs(y1-10 + y2-10)/2));

label1.Text = vertex.ToString();

label1.AutoSize = false;

label1.Width=17;

label1.Height = 17;

label1.Font = new Font("Aerial", 10, FontStyle.Italic);

CMap.Controls.Add(label1);

}

private void button2\_Click(object sender, EventArgs e)

{

CMap.Visible = true;

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void инструкцияToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("Количество вершин не более 20");

}

private void авторToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("Акимов A.A., КНиИТ, гр.341, 2015г.");

}

private void openFileDialog1\_FileOk(object sender, CancelEventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void button3\_Click(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

textBox1.Text = openFileDialog1.FileName;

k = textBox1.Text;

}

}

private void примерВходногоФайлаToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("8\n1 2(32) 3(95) 4(75) 5(57)\n2 3(5) 5(23) 8(16)\n3 4(18) 6(6)\n4 5(24) 6(9)\n5 7(20) 8(94)\n6 5(11) 7(7)\n7 8(81)\n8");

}

public void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

private void listBox1\_SelectedIndexChanged\_1(object sender, EventArgs e)

{

}

}

}

Листинг Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace EdmondsKarp

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}