МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Построение и анализ алгоритмов»
Тема: Максимальный поток сети

Студент гр. 0382 _____ Корсунов А.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург
2022

Цель работы.

Применить на практике знания о построение алгоритма Форда-Фалкерсона. Реализовать алгоритм Форда-Фалкерсона для поиска максимального потока в сети.

Задание.

Найти максимальный поток в сети, а также фактическую величину потока, протекающего через каждое ребро, используя алгоритм Форда-Фалкерсона.

Сеть (ориентированный взвешенный граф) представляется в виде триплета из имён вершин и целого неотрицательного числа - пропускной способности (веса).

Входные данные:

N- количество ориентированных рёбер графа

v0 - исток

vn - сток

vi vj ωij- ребро графа

vi vj ωij- ребро графа

...

Выходные данные:

Pmax - величина максимального потока

vi vj ωij - ребро графа с фактической величиной протекающего потока

vi vj ωij- ребро графа с фактической величиной протекающего потока

...

В ответе выходные рёбра отсортируйте в лексикографическом порядке по первой вершине, потом по второй (в ответе должны присутствовать все указанные входные рёбра, даже если поток в них равен 0).

Ход работы.

1. Был произведен анализ задания.

2. Был реализован алгоритм Форда-Фалкерсона:

1) Алгоритм:

Обнуляются все потоки, остаточная сеть изначально совпадает с исходной сетью.

а) В остаточной сети ищется путь из источника в сток (путь ищется помощью dfs), если такого пути нет, производится выход из алгоритма, если есть, через найденный путь пускается максимально возможный поток, а именно: на найденном пути в остаточной сети производится поиск ребра с минимальной пропускной способностью tmp. Для каждого ребра на найденном пути поток увеличивается на tmp, а в противоположном ему ребру («обратному) - уменьшается на tmp.

б) Для всех рёбер на найденном пути, а также для противоположных им рёбер, вычисляется новая пропускная способность. Если она стала ненулевой, ребро добавляется к остаточной сети, а если обнулилась, удаляется. Алгоритм работает, пока есть путь из истока в сток.

2) Структура данных, которая представляет переданный граф: двумерных массив типа int, который представляет из себя матрицу смежности пропускных способностей ребер, двумерный массив типа flow, который хранит дуги и их значения (в том числе обратные);

3) Функции и структуры данных:

*class ford_fulkerson

*int bandwidth_matrix[300][300] = { 0 } - матрица смежности пропускных способностей ребер вершин

*int prevs[300] = { 0 } - пути

*bool visited[300] = { 0 } - список просмотренных вершин

*int flow[300][300] = { 0 } - фактическая величина потока на ребре

*int istock - исток

*int stock сток

*void initPrevs() обнуление путей

*void dfs(int) - поиск путей в глубину

*bool getPath(int) - получение пути

*int alg() - максимальный поток

4) Тестирование:

a) Входные данные:

7

a

f

a b 7

a c 6

b d 6

c f 9

d e 3

d f 4

e c 2

Выходные данные:

12

a b 6

a c 6

b d 6

c f 8

d e 2

d f 4

e c 2

б) Входные данные:

16

a

e

a b 20

b a 20

a d 10

d a 10

a c 30

c a 30

b c 40

c b 40

c d 10

d c 10

c e 20

e c 20

b e 30

e b 30

d e 10

e d 10

Выходные данные:

60

a b 20

a c 30

a d 10

b a 0

b c 0

b e 30

c a 0

c b 10

c d 0

c e 20

d a 0

d c 0

d e 10

e b 0

e c 0

e d 0

в) Входные данные:

5

a d

a b 20

a c 1

b c 20

b d 1

c d 20

Выходные данные:

21

a b 20

a c 1

b c 19

b d 1

c d 20

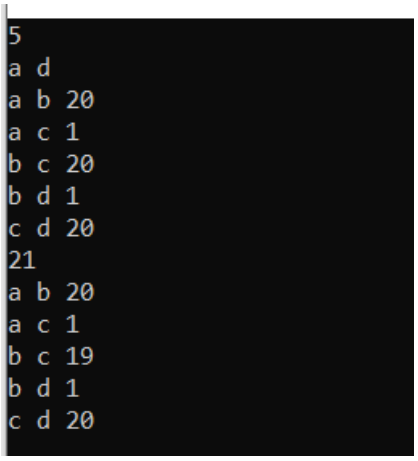

Рисунок 1 - Пример работы первой программы

Выводы.

Были применены на практике знания о построение алгоритма Форда-Фалкерсона. Реализован алгоритм Форда-Фалкерсона для поиска максимального потока в сети.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл ford_fulkerson.cpp:

```
#include <iostream>
#include <algorithm>

class ford_fulkerson
{
public:
	int bandwidth_matrix[300][300] = { 0 }; //матрица смежности пропускных способностей ребер вершин
	int prevs[300] = { 0 }; //пути
	bool visited[300] = { 0 }; //список просмотренных вершин
	int flow[300][300] = { 0 }; //фактическая величина потока на ребре
	int istock; //исток
	int stock; //сток

	void initPrevs(); //обнуление путей
	void dfs(int); //поиск путей в глубину
	bool getPath(int); //получение пути
	int alg(); //максимальный поток
};


void ford_fulkerson::initPrevs()
{
	for (int i = 0; i < 300; i++)
	{
		prevs[i] = -1;
	}
}

void ford_fulkerson::dfs(int v)
{
	visited[v] = true; //пометка, что пройденная вершина просмотренна
	for (int i = 0; i < 300; i++)
	{
		if (!visited[i] && (bandwidth_matrix[v][i] - flow[v][i] > 0 && bandwidth_matrix[v][i] != 0 || flow[v][i] < 0 && bandwidth_matrix[i][v] != 0)) //(если вершина не просмотрена + остаточный поток больше нуля на ребре + пропускная способность на ребре не равна нулю) или (фактическая величина
```

```
потока на ребре меньше нуля + пропускная способность не равна нулю)
                {
                        prevs[i] = v; //запоминаем путь
                        dfs(i);
                }
        }
}

bool ford_fulkerson::getPath(int v) //ищем путь до стока по dfs
{
        dfs(v);
        for (int i = 0; i < 300; i++)
        {
                visited[i] = false; //обнуление посещенных вершин
        }
        return (prevs[stock] != -1);
}

int ford_fulkerson::alg()
{
        int maxFlow = 0; //максимальный поток
        while (getPath(istock)) //пока не дошли до конца сток
        {
                int tmp = 10000000; //текущая минимальная пропускная
способность
                for (int v = stock; 0 <= prevs[v]; v = prevs[v]) //проход по найденному
пути
                {
                        tmp = std::min(tmp, bandwidth_matrix[prevs[v]][v] -
flow[prevs[v]][v]); //минимальная пропускная способность
                }
                for (int v = stock; 0 <= prevs[v]; v = prevs[v])
                {
                        flow[prevs[v]][v] += tmp; //увеличение "обратного" пути
ребра на tmp
                        flow[v][prevs[v]] -= tmp;; //уменьшение "прямого" пути ребра
на tmp
                }
                maxFlow += tmp; //увеличение максимального потока на tmp
                initPrevs();
        }
        return maxFlow;
}
```

```
int main()
{
	int N; //число поданных на вход дуг
	int cost = 0;
	char left; //левая вершина
	char rigth; //правая вершина

	std::cin >> N;
	std::cin >> left >> rigth;

	ford_fulkerson test;
	test.initPrevs();
	test.istock = left - '0';
	test.stock = rigth - '0';

	for (int k = 0; k < N; k++)
	{
		std::cin >> left >> rigth >> cost;
		int i = left - '0'; //перевод в int
		int j = rigth - '0'; //перевод в int
		test.bandwidth_matrix[i][j] = cost; //обнуление всех потоков сети (в
матрице смежности все пути будут равны 0
	}

	std::cout << test.alg() << '\n';

	for (int i = 0; i < 300; i++) //вывод фактических величин в потоке
	{
		for (int j = 0; j < 300; j++) {
			if (test.flow[i][j] != 0 && test.flow[i][j] < 0)
			{
				test.flow[i][j] = 0;
			}
			if (test.bandwidth_matrix[i][j] > 0)
			{
				std::cout << (char)(i + '0') << " " << (char)(j + '0') << "
" << test.flow[i][j] << '\n';
			}
		}
	}
	return 0;
}
```