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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Построение и анализ алгоритмов»

Тема: Потоки в сети

Студент гр.8303 _____ Рудько Д.Ю.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

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

Задание.

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

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

V_0 – исток

V_N – сток

V_i V_j W_{ij} – ребро графа

V_i V_j W_{ij} – ребро графа

…

Выходные данные:

P_{max} – величина максимального потока

V_i V_j W_{ij} – ребро графа с фактической величиной протекающегопотока

V_i V_j W_{ij} – ребро графа с фактической величиной протекающегопотока

…

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).

Пример входных данных

7

a

f

a b 7

a c 6

b d 6

c f 9

d e 3

d f 4

e c 2

Пример выходных данных

12

a b 6

a c 6

b d 6

c f 8

d e 2

d f 4

e c 2

Вариант 1.

Поиск в ширину. Поочерёдная обработка вершин текущего фронта, перебор вершин в алфавитном порядке.

Описание алгоритма.

В начале работы алгоритму на вход подается граф для поиска максимального потока, вершина-исток и вершина-сток графа. После чего производится поиск в ширину в графе.

На каждом этапе поиска в ширину с помощью очереди находится путь от истока к стоку. Из ребер пути находится ребро с минимальным весом. Из всех ребер пути от истока к стоку вычитается вес минимального ребра пути, а к ребрам пути от стока к истоку вес минимального ребра прибавляется. К переменной, отвечающей за величину максимального потока в графе, прибавляется вес минимального ребра пути.

Цикл поиска в ширину и изменения величины потока ребер в графе осуществляется до тех пор, пока поиск в ширину возможен. Результатом является значение переменной, отвечающей за величину максимального потока в графе. Фактический поток через ребра определяется как разность между первоначальным ребром и ребром, после преобразований.

Сложность алгоритма по операциям: O (E * F),
E – число ребер в графе, F – максимальный поток
На каждом шаге алгоритм увеличивает поток по крайней мере на единицу, следовательно, он сойдётся не более чем за O(F) шагов, где F — максимальный поток в графе. Можно выполнить каждый шаг за время O(*E*), где *E* — число рёбер в графе, тогда общее время работы алгоритма ограничено O(E * F).

Сложность алгоритма по памяти: O (N+E),
N – количество вершин, E – количество ребер

Описание функций.

bool bfs(Graph graph, char source, char sink, map<char, char>& path)

Функция поиска в ширину. Принимает исходный граф, вершину-исток, вершину-сток, ассоциативный массив пар path, из которого будет получен путь.

Функция возвращает true, если была достигнута вершина-исток, false — в обратном случае.

void FordFulkerson(Graph graph, char source, char sink)

Функция, осуществляющая алгоритм Форда-Фалкерсона нахождения максимального потока в сети. На вход принимает исходный граф, в котором будет находиться максимальный поток, вершина-исток и вершина-сток.

void printResult(Graph graph, Graph flow_graph, int max_flow)

Функция печати результата работы алгоритма. С помощью исходного графа graph и графа flow_graph, полученного в результате работы алгоритма, печатаются пары вершин с фактической величиной потока через ребра между ними. Также печатается максимальный поток в сети max_flow.

Тестирование.

```
7
a
f
a b 7
a c 6
b d 6
c f 9
d e 3
d f 4
e c 2

Текущая вешина a, соседи:
        b пропускная способность ребра = 7
        c пропускная способность ребра = 6
Текущая вешина b, соседи:
        d пропускная способность ребра = 6
Текущая вешина c, соседи:
        f пропускная способность ребра = 9

Текущая вешина a, соседи:
        b пропускная способность ребра = 7
Текущая вешина b, соседи:
        d пропускная способность ребра = 6
Текущая вешина d, соседи:
        e пропускная способность ребра = 3
        f пропускная способность ребра = 4

Текущая вешина a, соседи:
        b пропускная способность ребра = 3
Текущая вешина b, соседи:
        d пропускная способность ребра = 2
Текущая вешина d, соседи:
        e пропускная способность ребра = 3
Текущая вешина e, соседи:
        c пропускная способность ребра = 2
Текущая вешина c, соседи:
        f пропускная способность ребра = 3

Текущая вешина a, соседи:
        b пропускная способность ребра = 1
Текущая вешина b, соседи:
------------------------------------------------------------
Максимальный поток = 12

Поток, протекающий через ребра:
a b 6
a c 6
b d 6
c f 8
d e 2
d f 4
e c 2
Для закрытия данного окна нажмите <ВВОД>...
```

```
5
a
d
a c 1
a b 1
b d 1
c d 1
b c 1

Текущая вешина a, соседи:
        b пропускная способность ребра = 1
        c пропускная способность ребра = 1
Текущая вешина b, соседи:
        d пропускная способность ребра = 1

Текущая вешина a, соседи:
        c пропускная способность ребра = 1
Текущая вешина c, соседи:
        d пропускная способность ребра = 1

Текущая вешина a, соседи:
------------------------------------------------------------
Максимальный поток = 2

Поток, протекающий через ребра:
a b 1
a c 1
b c 0
b d 1
c d 1
Для закрытия данного окна нажмите <ВВОД>...
```

```
3
a c
a b 7
b c 12
b a 5

Текущая вешина a, соседи:
        b пропускная способность ребра = 7
Текущая вешина b, соседи:
        c пропускная способность ребра = 12

Текущая вешина a, соседи:
------------------------------------------------
Максимальный поток = 7

Поток, протекающий через ребра:
a b 7
b a 0
b c 7
Для закрытия данного окна нажмите <ВВОД>...
```

Вывод.

В ходе выполнения лабораторной работы был реализован алгоритм Форда-Фалкерсона, который находит максимальный поток в сети, а также фактическую величину потока, протекающего через каждое ребро.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

```
#include <iostream>
#include <algorithm>
#include <map>
#include <deque>
#include <queue>
#include <climits>

using namespace std;
typedef map<char, map<char, int>> Graph; //Граф для алгоритма
Graph graph;

bool bfs(Graph graph, char source, char sink, map<char, char>& path){
    cout << endl;
    map<char,bool> visited; //Посещенные вершины
    queue<char> q_vertex;  //Очередь вершин
    q_vertex.push(source);
    visited[source] = true;
    while (!q_vertex.empty())  //Пока очередь не пустая
    {
        char vertex = q_vertex.front();
        q_vertex.pop();
        cout << "Текущая вешина " << vertex << ", соседи: " << endl;
        for (auto i : graph[vertex]){                       //Просматриваются
соседи и кладутся в очередь
            if (i.second > 0 && !(visited[i.first])) {
                q_vertex.push(i.first);
                visited[i.first] = true;
                path[i.first] = vertex;
                cout << "\t" << i.first << " пропускная способность ребра = " <<
i.second << endl;
                if (i.first == sink) {
                    q_vertex = *(new queue<char>());
                    break;
                }
            }

        }
    }

    return visited[sink]; //Была ли достигнута финиш
}

void printResult(Graph graph, Graph flow_graph, int max_flow){
//Функция печати результата
    cout << "-----------------------------------------------------------" <<
endl;
    int flow;
    cout <<"Максимальный поток = "<< max_flow << endl; //Печать максимального
потока
    cout << "\nПоток, протекающий через ребра:" << endl;
    for (auto vertex: graph)
```

```
        for (auto i: graph[vertex.first]) {
            if (i.second - flow_graph[vertex.first][i.first] < 0)
                flow = 0;
            else
                flow = i.second - flow_graph[vertex.first][i.first];

            cout << vertex.first << " " << i.first << " " << flow << endl;   //
Печать потока через ребро
        }
}

void FordFulkerson(Graph graph, char source, char sink){
    Graph flow_graph = graph;                                   //Граф с потоками
    char from, to;
    map<char,char> path;                                  //Пары, составляющие путь
    int max_flow = 0;
    while (bfs(flow_graph, source, sink, path))        //Пока возможен поиск в
ширину
    {
        int current_flow = INT_MAX;
        for (to = sink; to != source; to = path[to])
//Восстанавливается путь от финиша к началу
        {
            from = path[to];
            current_flow = min(current_flow, flow_graph[from][to]);  //Находится
поток пути
        }
        for (to = sink; to != source; to = path[to]) //Восстанавливается путь от
финиша к началу
        {
            from = path[to];
            flow_graph[from][to] -= current_flow;  //Изменяется граф с потоком
            flow_graph[to][from] += current_flow;
        }

        max_flow += current_flow;  //Изменяется число максимального потока
    }

    printResult(graph, flow_graph, max_flow);
}

int main() {

    char source, sink;
    int n; //число ребер

    cin >> n >> source >> sink;

    for (int i = 0; i < n; i++) { //Считывание вершин графа
        char from, to;
        int capacity;
        cin >> from >> to >> capacity;
        graph[from][to] = capacity;
    }

    FordFulkerson(graph, source, sink); //Запуск алгоритма Форда-Фалкерсона

    return 0;
}
```