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

Студентка гр. 7383 _____ Ханова Ю.А.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2019

Цель работы

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

Вар. 4с. Списки смежности. Поиск пути по правилу: каждый раз выполняется переход по ребру, соединяющему вершины, имена которых в алфавите ближе всего друг к другу.

Реализация задачи

В ходе выполнения данной работы был реализован алгоритм Форда-Фалкерсона.

Изначально все остаточные потоки равны фактическим и на истоке стоит соответствующая метка. Затем из истока в сток ищется путь, указанный в варианте (т.е. для каждой последующей вершины ищется смежная вершина, такая что, на ней нет метки, остаточная пропускная способность не равно 0 и ее название ближе всего по алфавиту к предыдущей вершине). После нахождения каждого пути подсчитывается его минимальная пропускная способность (ребро с минимальной остаточный пропускной способностью), все метки стираются, все остаточные пропускные способности уменьшаются на значение минимальной, а фактические увеличиваются на то же значение.

Алгоритм продолжает работу, до тех пор пока существуют пути. Максимальный поток в сети в итоге будет равен сумме минимальных потоков, найденных на каждой итерации.

Тестирование

Программа собрана в операционной системе Ubuntu 17.04 с использованием компилятора g++. В других ОС и компиляторах тестирование не проводилось. Результаты тестирования показали, что поставленная цель выполнена. Результаты тестирования представлены в Приложении Б.

Так же было проведено исследование алгоритма. Сложность поиска максимального потока оказывается равной O(F), где F — максимальный поток в сети. Каждая итерация занимает O(E), где E — количество ребер.

Общее время работы алгоритма O(F*E).

Выводы

В ходе выполнения лабораторной работы была решена задача поиска максимального потока в сети, а также фактическую величину потока, протекающего через каждое ребро на языке C++, и исследован Форда-Фалкерсона. Полученный алгоритм имеет сложность O(F*E). Код программ представлен в Приложении А.

ПРИЛОЖЕНИЕ А.
КОД ПРОГРАММЫ

```
#include <iostream>
#include <climits>
#include <cstring>
#include <queue>
#include <map>
#include <cmath>
#define A 'z'

using namespace std;
using List = map<char, map <char, int>>;

class Graph{
private:
  char distance[A];
  List orig_graph;
  List real_graph;
  array <bool, A> check;
public:
  Graph() = default;
  ~Graph() = default;
  void Print();
  void Insert(char from, char to, int way);
  int Residual_Capacity(char from, char to);
  bool Find_way(char source, char sink);
  int Ford_Fulkerson(char source, char sink);

};


bool Graph::Find_way(char source, char sink){
  check.fill (false);
  check[source] = true;
  distance[source] = -1;
  queue <char> queue;
  queue.push(source);
  while (!queue.empty()) {
      char from = queue.front();
      for (char to = 0; to < A; to++) {
          if (!check[to] && Residual_Capacity(from, to) > 0) {
              queue.push(to);
              check[to] = true;
```

```
                distance[to] = from;
                if (to == sink) {
                    queue.pop();
                    return (true);
                }
            }
        }
        queue.pop();
    }
    return (false);
}

int Graph::Residual_Capacity(char from, char to){
    return (orig_graph[from][to] - real_graph[from][to]);
}

void Graph::Insert(char from, char to, int way){
    orig_graph[from][to] = way;
    real_graph[to][from]= 0;
}

void Graph::Print(){
    for (char i = 0; i < A; i++)
        for (char j = 0; j < A; j++)
            if (orig_graph[i][j])
                cout << char(i) << " " << char(j) << " " <<
max(real_graph[i][j], 0) <<endl;
}

int Graph::Ford_Fulkerson(char source, char sink) {
    int maxFlow = 0;
    while (Find_way(source, sink)) {
        int minFlow = INT_MAX;
        char to = sink;
        for(int i = sink; 0 <= distance[i]; i = distance[i])
            minFlow = min(minFlow, Residual_Capacity(distance[i], i));
        for(int i = sink; 0 <= distance[i]; i = distance[i]) {
            real_graph[distance[i]][i] += minFlow;
            real_graph[i][distance[i]] -= minFlow;
        }
        maxFlow += minFlow;
        memset(distance, -1, A * sizeof(char));
        check.fill (false);
    }
    return (maxFlow);
}
```

```
int main(){
  int N, w;
  char source, sink, v, u;
  cin >> N >> source >> sink;
  Graph new_graph;
  for(int i = 0; i < N; i++){
    cin >> v >> u >> w;
    new_graph.Insert(v, u, w);
  }
  cout << new_graph.Ford_Fulkerson(source, sink) <<endl;
  new_graph.Print();

  return 0;
}
```

ПРИЛОЖЕНИЕ Б.

ТЕСТОВЫЕ СЛУЧАИ

Результаты тестов представлены в табл. 1.

Таблица 1 – запуск lr3.cpp.

Входные данные	Выходные 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	12 a b 6 a c 6 b d 6 c f 8 d e 2 d f 4 e c 2