**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. 6381 |  | Поляков Н. С. |
| Преподаватель |  | Филатов А. Ю. |

Санкт-Петербург

2018

**Цель работы.**

Разработать программу, коротая находит максимальный поток в сети по алгоритму Форда-Фалкурсона.

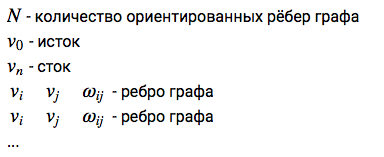
**Условия задания.**

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

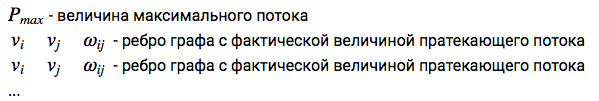
Сеть (ориентированный взвешенный граф) представляется в виде триплета из имён вершин и целого неотрицательного числа - пропускной способности (веса).

В ответе выходные рёбра отсортируйте в лексикографическом порядке по первой вершине, потом по второй

**Входные данные.**

****

**Выходные данные.**



**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

**Соответствующие выходные данных.**

12

a b 6

a c 6

b d 6

c f 8

d e 2

d f 4

e c 2

**Описание структур данных.**

|  |  |  |
| --- | --- | --- |
| Класс | Имя | Назначение |
| Edge | unsigned int weight | Вес ребра |
| unsigned int stream | Поток |
| Vertex \*end | Вершина, в которую ребро входит |
| Edge \*next | Следующее ребро в списке |
| void setWeight(unsugned int) | Устанавливает поле weight |
| void setEnd(Vertex\*) | Устанавливает поле end |
| void setNext(Edge\*) | Устанавливает поле next |
| unsigned int getWeight() | Возвращает значение поля weight |
| Vertex\* getEnd() | Возвращает значение поля end |
| Edge(double,Vertex\*) | Конструктор |
| Edge\*& getNext() | Возвращает значение поля next |
| void addToList(Priority\_list&,Vertex\*) | Добавляет вершину end в список приоритетов |
| Graph | std::vector<Vertex\*> list\_of\_Vertexes | Массив вершин графа |
| Vertex \*start | Исток |
| Vertex \* end | Сток |
| Vertex\* addVertex(char) | Добавляет вершину к графу |
| void addEdge(char,char,double) | Добавляет ребро в граф |
| unsigned int findMaxStream(char,char) | Находит максимальный поток в сети |
| void printStream() | Вывод ребер с соответствующими им потоками |
| Vertex\* findVertex(char) | Находит вершину по ее имени |
| Graph() | Конструктор |
| Vertex | char name | Имя вершины |
| bool visited | Индикатор посещенности |
| std::vector<Edge> edges | Массив инцидентных ребер |
| Vertex \*prevVisited | Предыдущая посещенная вершина |
| char getName() | Возвращает значение поля name |
| bool getVisited() | Возвращает значение поля visited |
| void addToVisited(Priority\_list&) | Добавляет вершину к посещенным |
| void addEdge(double,Vertex\*) | Добавляет инцидентное ребро |
| Vertex(char) | Конструктор |
| void setPrevVisited(Vertex\*) | Устанавливает поле prevVisited |
| void setVisited(bool) | Устанавливает поле visited |
| void setName(char) | Устанавливает поле name |
| Vertex\* getPrevVisited() | Возвращает значение поля prevVisited |

**Описание алгоритма.**

Для поиска максимального потока в сети, производится поиск путей от истока к стоку. Поиск путей производится с помощью жадного алгоритма, т. е. на каждом шаге выбирается ребро с максимальной остаточной пропускной способностью. В каждом пути находится ребро с наименьшим весом и на это число увеличивается величина потока для каждого ребра этого пути. После того, как построить еще один путь от истока к стоку становится невозможно, значение потока между смежными вершинами, имеющими ребра в обе стороны, пересчитывается (из каждого вычитается значение наименьшего из этих двух потоков).

**Тестирование.**

|  |  |  |
| --- | --- | --- |
| Ввод | Вывод | Рисунок |
| 9  K  P  K L 5  K O 15  O L 25  L N 35  N O 10  L M 35  M N 20  M P 20  N P 10 | 20  K L 5  K O 15  L M 10  L N 10  M N 0  M P 10  N O 0  N P 10  O L 15 | ../../../Screen%20Shot%202018-04-24%20at%2019.28.53.png |
| 13  K  T  K L 5  K O 15  K Q 10  O S 10  L R 10  L T 25  N T 5  N O 10  S N 20  S T 20  S Q 20  Q O 40  R S 5 | 15  K L 5  K O 10  K Q 0  L R 0  L T 5  N O 0  N T 5  O S 10  Q O 0  R S 0  S N 5  S Q 0  S T 5 | ../../../Screen%20Shot%202018-04-24%20at%2019.31.57.png |

**Вывод.**

В ходе выполнения данной лабораторной работы был изучен алгоритм Форда-Фалкерсона и разработана программа, реализующая его.