

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. 8304

Щука А. А.

Преподаватель

Размочаева Н. В.

Санкт-Петербург

2020

Цель работы.

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

Вариант 1. Поиск в ширину. Поочерёдная обработка вершин текущего фронта, перебор вершин в алфавитном порядке.

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

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 протекающего потока

$V_i \ V_j \ W_{ij}$ – 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).

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

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 (если такой вершины не существует, то она достраивается). К переменной, отвечающей за максимальный поток в графе, прибавляется вес минимального ребра пути.

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

В консоль выводится результат работы алгоритма и промежуточные результаты, такие как текущие вершины поиска в ширину и их соседи с расстоянием до них, найденный путь, преобразованный граф.

Сложность алгоритма по операциям: $O(E * F)$, E – число ребер в графе, F – максимальный поток

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

Описание функций и структур данных.

```
using Graph = std::map<char, std::map<char, int>>>;
```

Структура данных, используемая для хранения направленного графа. Представляет собой ассоциативный контейнер хранения вершин и соответствующего ей контейнера вершина-расстояние. Для каждой вершины хранится ассоциативный массив вершин, до которых можно добраться из текущей и вес пути до них.

```
bool BFS(Graph &graph, char start, char end, std::map<char, char>& path)
```

Функция поиска в графе в ширину. На вход принимает ссылку на граф `graph`, в котором будет осуществляться поиск, стартовую и конечную вершину `start` и `end` соответственно, ассоциативный массив пар `path`, из которого будет получен путь.

Функция возвращает `true`, если при поиске была достигнута финальная вершина, `false` – противном случае.

```
void printCurrentFlows(Graph& flowGraph, int pathFlow, int  
maxCurrentFlow, std::string& pathStr)
```

Функция печати текущего состояния графа `flowGraph`, найденного пути `pathStr` с потоком через него размером `pathFlow`, текущего суммарного потока `maxCurrentFlow`.

```
void printResult(Graph& graph, Graph& flowGraph, int maxFlow)
```

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

```
void FFA(Graph &graph, char start, char finish)
```

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

Тестирование.

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

Результат работы программы:

Start wide search

Current vertex a and his neighbor:

 b with possible flow = 7

 c with possible flow = 6

Current vertex b and his neighbor:

 d with possible flow = 6

Current vertex c and his neighbor:

 f with possible flow = 9

Current vertex d and his neighbor:

 e with possible flow = 3

Current vertex f and his neighbor:

 hasn't not visited neighbor

Current vertex e and his neighbor:

 hasn't not visited neighbor

Find new path with flow = 6: a --> c --> f

Flow graph:

 a b 7

 a c 0

 b d 6

 c a 6

 c f 3

 d e 3

 d f 4

 e c 2

 f c 6

Current flow of graph = 6

Start wide search

Current vertex a and his neighbor:
 b with possible flow = 7

Current vertex b and his neighbor:
 d with possible flow = 6

Current vertex d and his neighbor:
 e with possible flow = 3
 f with possible flow = 4

Current vertex e and his neighbor:
 c with possible flow = 2

Current vertex f and his neighbor:
 hasn't not visited neighbor

Current vertex c and his neighbor:
 hasn't not visited neighbor

Find new path with flow = 4: a --> b --> d --> f
 Flow graph:

```

a b 3
a c 0
b a 4
b d 2
c a 6
c f 3
d b 4
d e 3
d f 0
e c 2
f c 6
f d 4

```

Current flow of graph = 10

 Start wide search

Current vertex a and his neighbor:
 b with possible flow = 3

Current vertex b and his neighbor:
 d with possible flow = 2

Current vertex d and his neighbor:
 e with possible flow = 3

Current vertex e and his neighbor:
 c with possible flow = 2

Current vertex c and his neighbor:
 f with possible flow = 3

Current vertex f and his neighbor:
 hasn't not visited neighbor

Find new path with flow = 2: a --> b --> d --> e --> c --> f

Flow graph:

```
a b 1
a c 0
b a 6
b d 0
c a 6
c e 2
c f 1
d b 6
d e 1
d f 0
e c 0
e d 2
f c 8
f d 4
```

Current flow of graph = 12

Start wide search

Current vertex a and his neighbor:

 b with possible flow = 1

Current vertex b and his neighbor:

 hasn't not visited neighbor

Result of algorithm:

Max flow = 12

```
a b 6
a c 6
b d 6
c f 8
d e 2
d f 4
e c 2
```

Входные данные:

8

a

h

a b 5

a c 4

a d 1

b g 1

c e 2

c f 3

d e 6

e h 4

f h 4

g h 8

Результат работы программы:

Start wide search

Current vertex a and his neighbor:

b with possible flow = 5

c with possible flow = 4

d with possible flow = 1

Current vertex b and his neighbor:

g with possible flow = 1

Current vertex c and his neighbor:

e with possible flow = 2

f with possible flow = 3

Current vertex d and his neighbor:

hasn't not visited neighbor

Current vertex g and his neighbor:

hasn't not visited neighbor

Current vertex e and his neighbor:

h with possible flow = 4

Current vertex f and his neighbor:

hasn't not visited neighbor

Current vertex h and his neighbor:

hasn't not visited neighbor

Find new path with flow = 2: a --> c --> e --> h

Flow graph:

a b 5

a c 2

a d 1

b g 1

c a 2

c e 0

c f 3

d e 6

e c 2

e h 2

h e 2

Current flow of graph = 2

Start wide search

Current vertex a and his neighbor:

b with possible flow = 5

c with possible flow = 2

d with possible flow = 1

Current vertex b and his neighbor:
g with possible flow = 1

Current vertex c and his neighbor:
f with possible flow = 3

Current vertex d and his neighbor:
e with possible flow = 6

Current vertex g and his neighbor:
hasn't not visited neighbor

Current vertex f and his neighbor:
hasn't not visited neighbor

Current vertex e and his neighbor:
h with possible flow = 2

Current vertex h and his neighbor:
hasn't not visited neighbor

Find new path with flow = 1: a --> d --> e --> h
Flow graph:

a b 5
a c 2
a d 0
b g 1
c a 2
c e 0
c f 3
d a 1
d e 5
e c 2
e d 1
e h 1
h e 3

Current flow of graph = 3

Start wide search

Current vertex a and his neighbor:
b with possible flow = 5
c with possible flow = 2

Current vertex b and his neighbor:
g with possible flow = 1

Current vertex c and his neighbor:
f with possible flow = 3

Current vertex g and his neighbor:
hasn't not visited neighbor

Current vertex f and his neighbor:
hasn't not visited neighbor

Result of algorithm:

Max flow = 3

a b 0

a c 2

a d 1

b g 0

c e 2

c f 0

d e 1

e h 3

Выводы.

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.