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

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и Структуры Данных»

Тема: «Потоки в сетях. Алгоритм Форда-Фалкерсона»

Студент гр. 8301 _____ Урбан М.Ф.
Преподаватель _____ Тутуева А. В.

Санкт-Петербург
2020

Цель

Реализовать алгоритм, находящий максимальный поток в сети с помощью алгоритма Форда-Фалкерсона.

Входные данные: текстовый файлы со строками в формате V1, V1, P, где V1, V2 направленная дуга транспортной сети, а P – ее пропускная способность. Исток всегда обозначен как S, сток – как T

Описание программы

Сперва находится вершина, доступная из истока. Выбирается путь с наибольшей пропускной способностью, если таких несколько – выбирается первый найденный. Если такой путь найден, то пытаемся попасть в сток.

Если через выбранный путь нельзя дойти до стока, путь ищется заново, при этом прошлый путь отмечается как посещённый, для того чтобы по нему нельзя было пройти ещё раз. Посещенные пункты сохраняются для последующей обработки.

После попадания в сток производится обработка посещённых пунктов. Вначале находится наименьшая пропускная способность в пути и вычитается из остальных. Затем накапливается максимальный поток на основе пройденного пути.

Алгоритм завершается после того как были перепробованы все способы добраться до стока или если пропускная способность путей в сток становится равной 0.

В программе используются следующие структуры данных:

- Класс **List** (из прошлых работ)
- Класс **Map** (из прошлых работ)
- Класс **Queue** (из прошлых работ)
- Класс **flowStruct:**

Используется в нескольких методах класса Graph. Имеет три поля: **checked** (проверка на то, посещался ли данный пункт), **capacity** (мап, хранящий пропускные способности между предыдущими пунктами), **bands** (мап пунктов, до которых можно добраться из данного пункта).

- Структура **Graph**:

Данная структура реализует поиск максимального потока в сети с помощью алгоритма Форда-Фалкерсона.

Эта структура хранит очередь **fLines**, список **pathToEnd**, и мэп **items** (хранит названия пунктов и их индекс).

Данная структура имеет два публичных метода. Конструктор получает текстовый файл и передаёт его в метод, считывающий его. Также запускает функцию создания

графа. Второй метод – **getFlowSize**, который возвращает максимальный поток, считающийся в методе **countFlow**.

Этот класс также имеет несколько приватных методов:

_readFile

Метод построчно считывает файл и сохраняет строки в очередь **fLines**. Также с помощью метода **readWord** из строк выделяются пункты и сохраняются в мэп **items.** В качестве ключа сохраняются их названия, в качестве значения сохраняется их индивидуальный индекс, при этом индекс истока всегда 0, а стока – максимальный индекс.

Метод **readLine** получает на вход строку из очереди **fLines** и делит её на два пункта и пропускную способность, которая переводится в целочисленный тип данных.

Метод **makeGraph** создает массив **flow**, затем метод получает строки о пунктах и их пропускных способностях из очереди **fLines** и заполняет данными массив **flow**

Метод **countFlow** ищет максимальный поток.

Сначала ищется пункт, в который можно пойти из истока. Выбирается путь с наибольшей пропускной способностью (если таких несколько – выбирается первый найденный). Если такой путь найден, то вызывается рекурсивный метод **makeWay**.

Данный метод работает по аналогии с **countFlow**, но при этом рекурсивно пытается попасть в сток. Если оказывается, что через выбранный путь нельзя дойти до стока, то путь ищется заново (при этом по старому пути пойти нельзя, поскольку он отмечается как посещенный). Посещенные пункты помещаются в список **pathToEnd** для последующей его обработки.

После попадания в сток начинается обработка списка **pathToEnd**. Сперва находится наименьшая пропускная способность и вычитается из остальных (если пропускная способность путей в сток становится равной 0, то поиск дальнейших путей прекращается). Затем накапливается максимальный поток на основе пройденного пути.

Таким образом, методы работают до тех пор, пока можно добраться от истока до стока, а затем возвращают максимальный поток.

Метод **updateFlow** используется в **countFlow.** Возвращает **true**, если нет пути в сток, и **false** если путь есть. Также в этом методе увеличивается пропускная способность.

Временная сложность

Graph(file) – O(1);(содержит **_readFile** и **makeGraph**)

_readFile – O($n*\log$);

readWord – O(n);

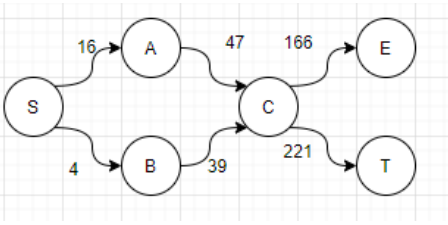
makeGraph – O($n*\log$);

readLine – O(1);

getFlowSize – O(1) (содержит **countFlow**);

countFlow – O($n^2*m*\log$).

Примеры работы




```
S A 16
S B 4
A C 47
B C 39
C E 166
C T 221
```

```
The widest flow: 20
```

```
S A 200
A B 100
A C 30
B D 160
C D 50
D K 30
B M 45
C K 33
M E 23
E T 30
K T 11
```

```
The widest flow: 34
```

```
S A 20
S D 30
S G 10
S F 50
A B 40
B C 5
C F 25
C H 15
D B 30
D E 20
E G 5
E F 30
F H 10
F T 30
G T 20
H T 20
```

```
The widest flow: 60
```

Код на GitHub