Сети. Потоки в сетях

Толстов Роберт Сергеевич,
Рудяк Артём Станиславович
3 курс, «Программная инженерия»

Саратов, 2025

Содержание

Введение

Графы — это прекрасный способ описать многие процессы, которые представляют собой связи между какими-то объектами. Однако в реальных ситуациях эти самые связи между узлами могут быть ограниченными с точки зрения физики.

Представим небольшую, но очень гордую африканскую деревню. Там есть водонапорная башня — источник воды, а также дома жителей. Между ними проложены трубы, каждая из которых имеет ограниченную пропускную способность — максимальный объём воды, который может пройти по ней за час. Нужно доставить воду в каждый дом.

Если бы вычисления проводились с использованием классической модели графа, то учёт физических ограничений бы не проводился. Произошла бы авария, а множество африканских детишек погибло бы от обезвоживания. Именно для таких ситуаций, когда нам **важна пропускная способность рёбер** и была описана теория потоковых сетей.

В рамках данной лекции мы немного погрузимся (скорее, погуляем по мелководью) теории потоков, а также рассмотрим алгоритмы, которые используются при решении прикладных задач.

Теоретические основы

Введём некоторые определения для формализации нашего разговора.

Сеть $G = \langle V, E \rangle$ — это ориентированный граф, такой что:

$$\forall (u, v) \in E \quad c(u, v) > 0;$$

$$\forall (u, v) \notin E \quad c(u, v) = 0.$$

Функцию $c(u, v)$ называют **пропускной способностью** ребра (u, v). Мы будем рассматривать транспортные сети, в которых выделяются две вершины: исток s и сток t.

Поток f в сети G — это функция $f : V \times V \to \mathbb{R}_{\geqslant 0}$, такая что $\forall u, v \in V$:

1. $f(u, v) = -f(v, u)$ (антисимметричность);
2. $f(u, v) \leq c(u, v)$;

Вообще говоря, есть другое определение потока в сети, но рассматривать его в рамках лекции мы не будем. Ввёл его Асанов. Оно не вводит антисимметричность в графе, из-за чего работать с ним труднее.

Величину потока определяют как $|f| = \sum\limits_{u \in V} f(s, v)$. Это общий объём, выходящий из истока (равно как и входящий в сток).

Закон сохранения потока

Также выделяют третье свойство потока, называемое «законом сохранения потока»:

$$\forall v \in V \setminus \{s, t\} \quad \sum_{\substack{u \in V \\ (u,v) \in E}} f(u, v) = \sum_{\substack{w \in V \\ (v,w) \in E}} f(v, w).$$

Проще говоря, сумма входящих потоков равна сумме выходящих. Очевидно, что выполняется для всех вершин кроме истока и стока.

Пример потоковой сети

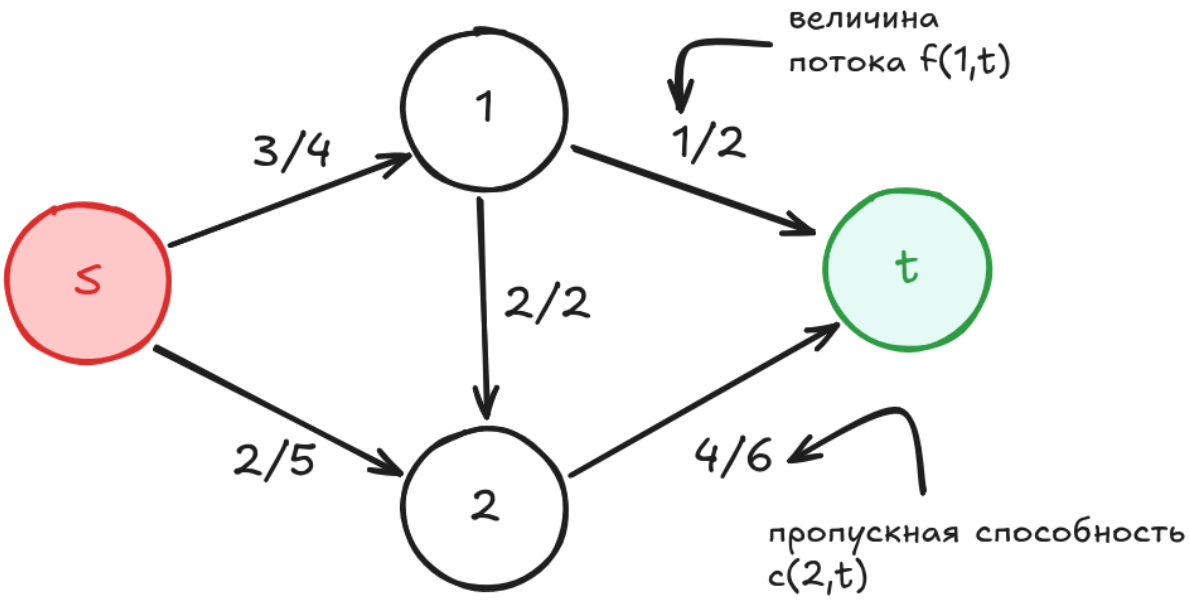


Рисунок 1: Пример потоковой сети

Так, на иллюстрации выше можно увидеть пример потоковой сети.

Первое число означает величину потока, второе — пропускную способность ребра. Отрицательные величины потока не указаны (так как они мгновенно получаются из антисимметричности). Сумма входящих рёбер везде (кроме источника и стока) равна сумме исходящих. Кроме

того, величина потока на ребре никогда не превышает пропускную способность этого ребра.

Величина потока в этом примере равна $3 + 2 = 5$ (считаем от s).

Подведём итоги:

- **Источник** — вершина, из которой исходит поток;
- **Сток** — вершина, в которую поток должен поступить;
- **Дуга** — ориентированное ребро графа, по которому движется поток;
- **Пропускная способность** — максимальный объём потока, разрешённый на дуге;
- **Поток** — реальное значение ресурса, которое протекает по дуге.

Задача максимального потока

Ключевой в теории потоков является **задача о максимальном потоке**. Сформулируем её.

Задача. Пусть дана сеть $G = \langle V, E \rangle$. Требуется найти функцию потока:

$$f_{\max} = \max_f |f|,$$

которая, очевидно, удовлетворяет всем свойствам функции потока. Такую функцию и называют функцией максимального потока.

Для решения этой задачи используют алгоритмы, которые будут рассмотрены далее.

Алгоритм Форда-Фалкерсона

Данный алгоритм основан на итеративном поиске увеличивающих путей — путей от источника к стоку в остаточной сети, по которым можно дополнительно провести поток.

- **Увеличивающим путём** называют путь из s в t остаточной сети, по которому каждая дуга имеет положительную остаточную пропускную способность. Таким образом, по данному пути можно увеличить поток.
- **Остаточной сетью** называется вспомогательный граф, показывающий, сколько еще можно накачать по ребру. Для каждого ребра определяют остаточную пропускную способность $c_f(u, v) = c(u, v) - f(u, v)$

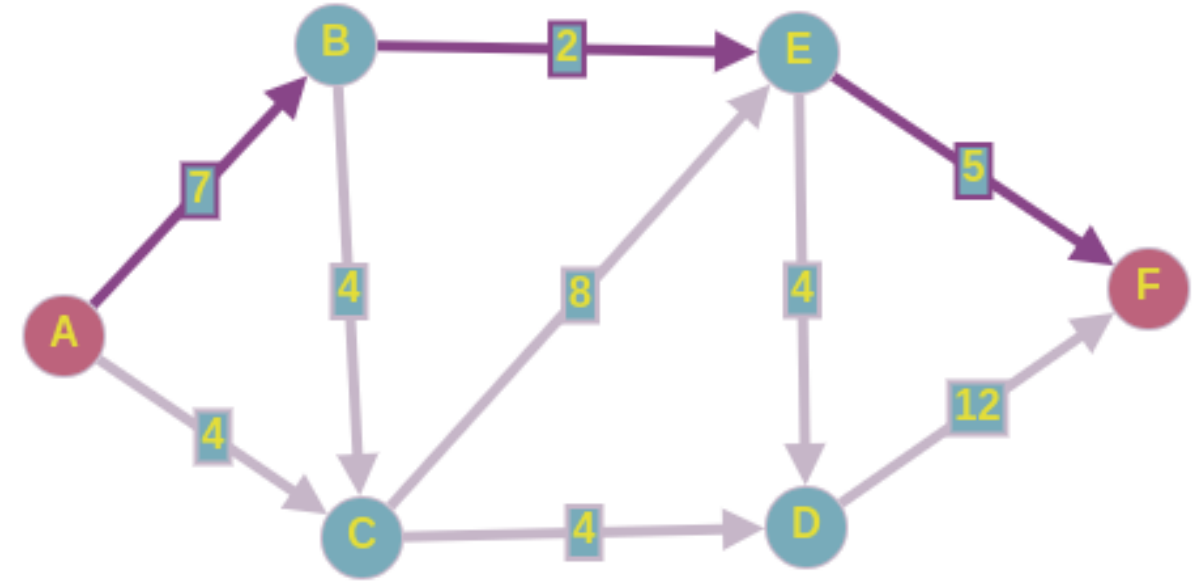
Описание алгоритма

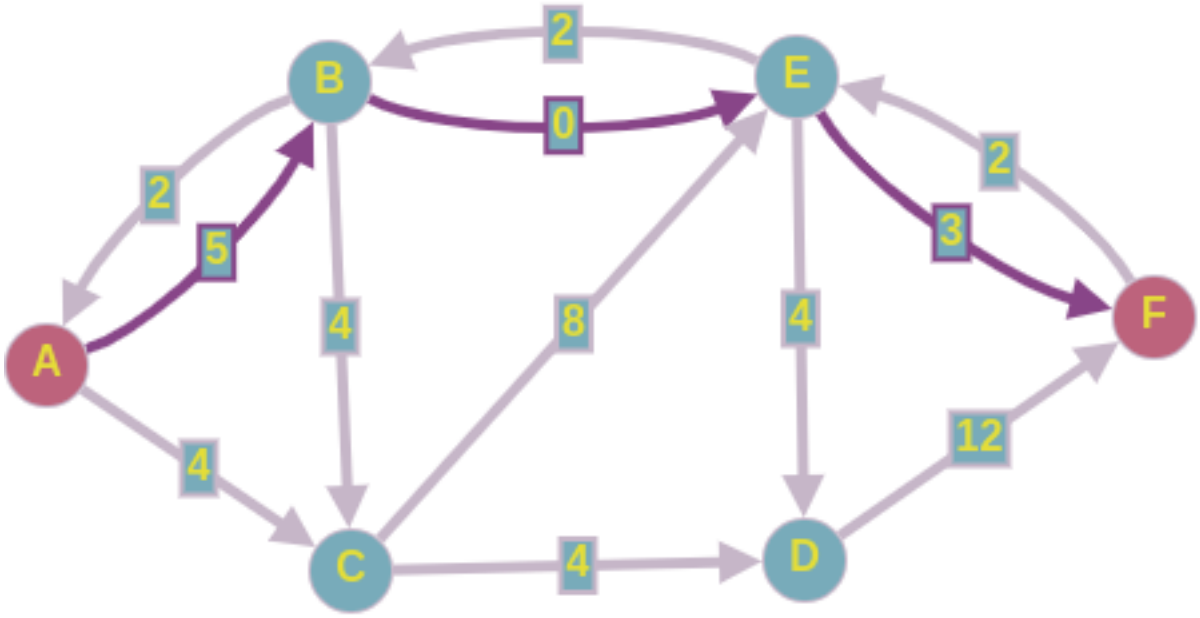
1. Все потоки выставляем в 0.
2. В остаточной сети находим любой путь из источника в сток. Если нет, то останавливаемся
3. На найденном **увеличивающем пути** пускаем максимально возможный поток
    1. На найденном пути ищем ребро с минимальной пропускной способностью $c_{\min}$
    2. Для каждого ребра на пути увеличиваем поток на $c_{\min}$, а противоположный ему уменьшаем на эту величину.
    3. Для всех рёбер на найденном пути, а также для противоположных им, вычисляем новую пропускную способность. Если она ненулевая, то добавляем ребро к остаточной сети, а если обнулилась, то стираем.
4. Возвращаемся на шаг 2.

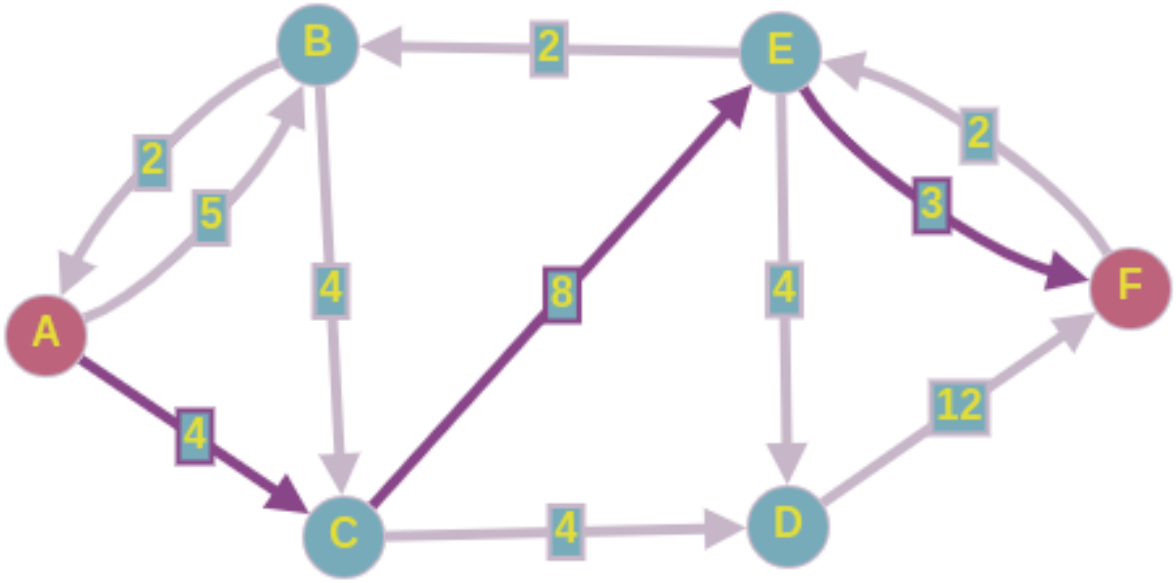
Псевдокод

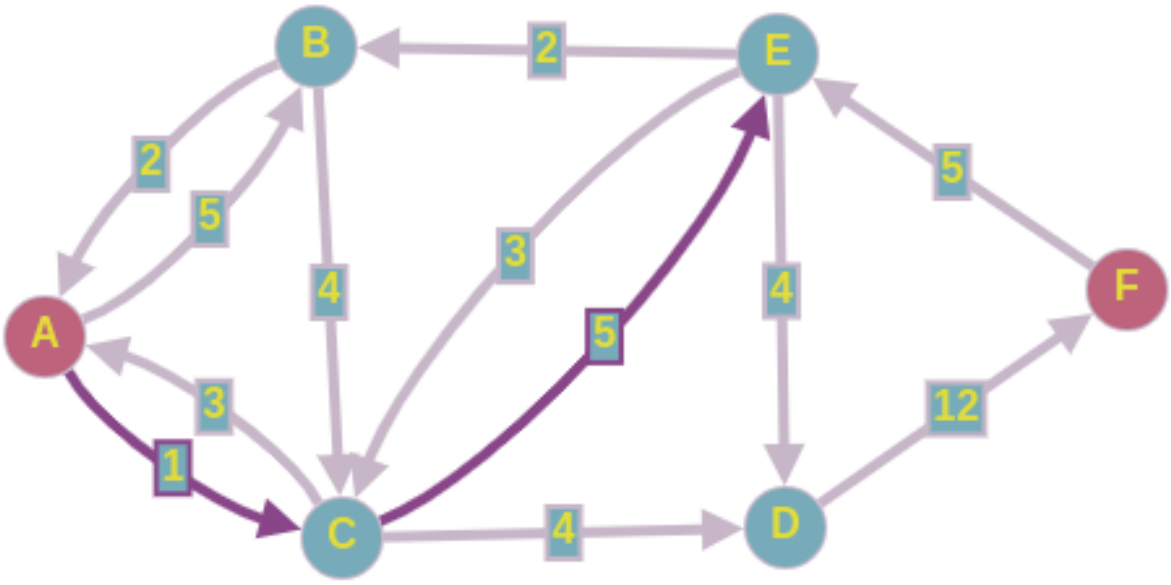
```
func FordFulkerson(G, s, t):
  для всех рёбер (u, v) в G:
    f(u, v) := 0

  пока существует увеличивающий путь P из s в t в остаточной сети G_f:
    delta := min(c_f(u, v), (u, v)) // бутылочное горлышко
    для (u, v) в P:
      f(u, v) := f(u, v) + delta // увеличиваем по прямому
      f(v, u) := f(v, u) - delta // уменьшаем по обратному
 вернуть сумму потоков, выходящих из s
```

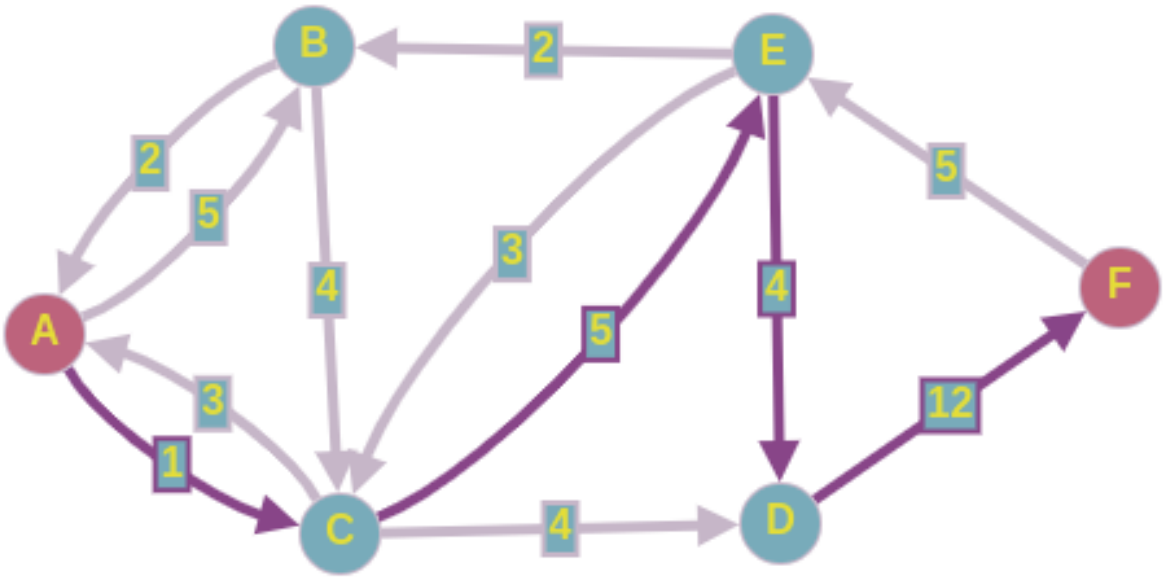
Работа алгоритма

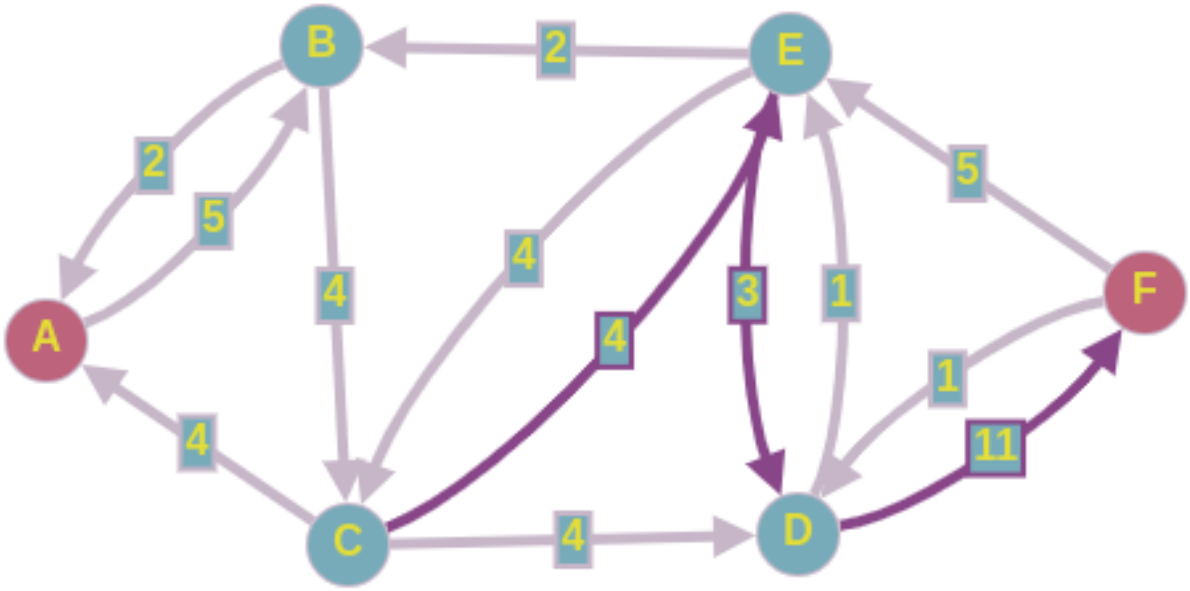


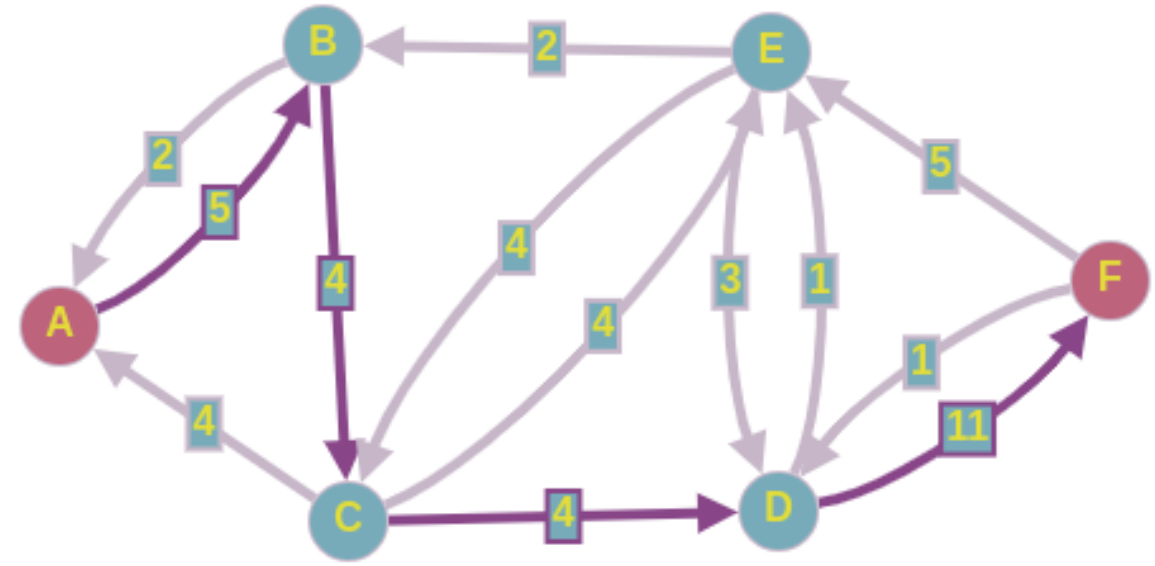


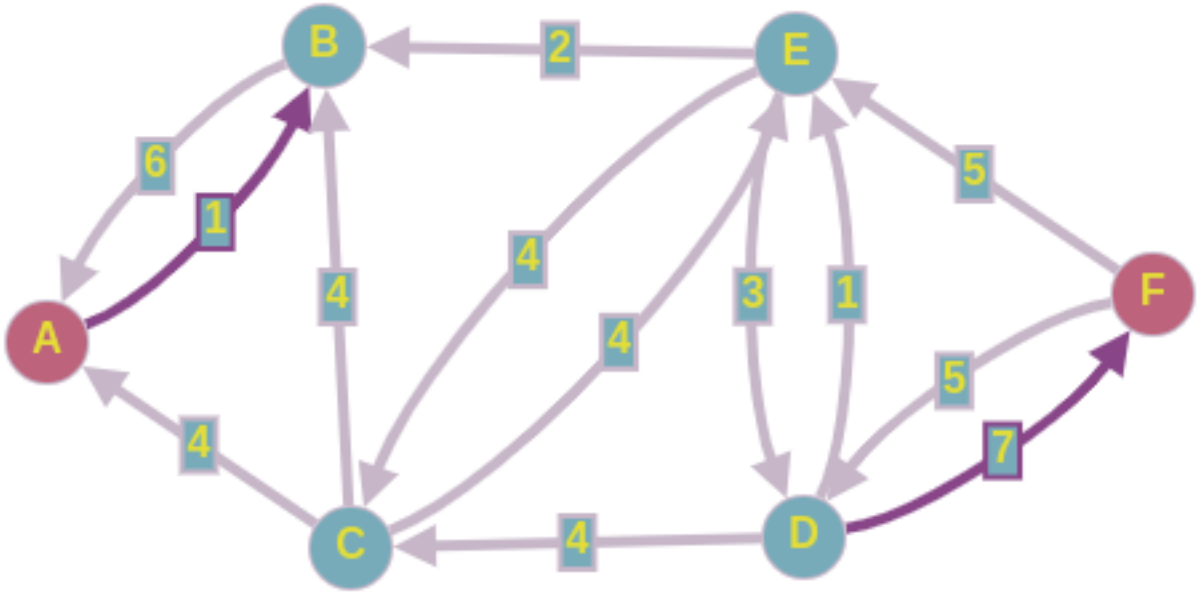

B
2
E
2
2
5
3
F
4
8
4
A
4
12
C
4
D


B
2
E
2
5
5
3
4
4
F
5
A
3
12
1
C
4
D


B
2
E
2
5
3
4
5
4
5
F
A
3
12
1
C
4
D


B
2
E
2
5
4
4
3
1
5
F
A
4
1
4
11
C
4
D





Недостатки

Так как алгоритм не предписывает, как именно искать путь в остаточной сети, то в худших случаях это может приводить к выбору “плохих” увеличивающих путей, по одному единичному потоку за итерацию. Ну и время работы алгоритма напрямую зависит от величины максимального потока. Если максимальный поток очень большой, то алгоритм может работать долго.

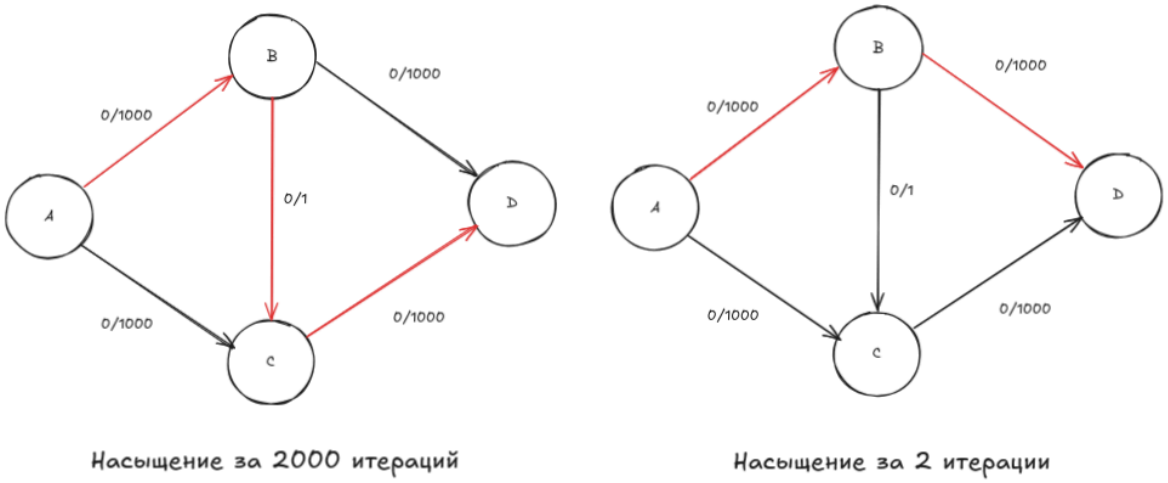
Что более важно, так это то, что алгоритм корректно работает разве что на целочисленных значениях потока, а на иррациональных и вовсе не завершиться.

Все эти недостатки привели к появлению новых алгоритмов, среди которых алгоритм Эдмондса-Карпа.

Алгоритм Эдмондса-Карпа

Это по сути реализация метода Форда-Фалкерсона, в которой в качестве увеличивающего пути выбирается кратчайший по рёбрам путь в остаточной сети. Алгоритмическая сложность такого решения составляет $O(VE^2)$

Сравнение с классическим алгоритмом



Алгоритм масштабирования потока

Ещё одна доработка метода Форда-Фалкерсона. Она заключается в том, чтобы сначала пускать поток по более пропускным путям. Т.е. выдавая им более высокий приоритет, нежели путям с низкой пропускной способностью. Ключевым нововведением этого алгоритма является параметр масштаба пути Δ, который инициализируется максимальной степенью двойки, не превосходящей максимальную пропускную способность $U = \max\limits_{(u,v)\in E} c(u,v)$ сети.

Псевдокод

```
1 func maxFlowByScaling(G, s, t):
2   для всех рёбер (u, v) в G:
3     f(u, v) := 0
4
5   scale = 2 ** floor (log_2(U))
6   пока scale >= 1
```

```
7     пока существует увеличивающий путь P из s в t в остаточной
сети G_f:
8       delta := min(c_f(u, v), (u, v)) // бутылочное горлышко
9       для (u, v) в P:
10        f(u, v) := f(u, v) + delta // увеличиваем по прямому
11        f(v, u) := f(v, u) - delta // уменьшаем по обратному
12    scale = scale / 2
13
14  вернуть сумму потоков, выходящих из s
15
```

Очевидно, что при $\text{scale} = 1$ алгоритм превращается в классический метод Форда-Фалкерсона.

Применение алгоритмов

Области применения алгоритмов поиска максимального потока довольно очнвидны:

- Транспортные сети. Например, при перевозке товаров от поставщика между распределительными пунктами до заказчика.
- Сетевые задачи. Определение максимальной пропускной способности компьютерной сети.

В целом любые сети, в которых требуется перегон какого-то ресурса из одной точки в другую через лимитированные по пропускной способности пути, подходят под алгоритм поиска максимального потока

Мультипотоки

Кроме классических потоках в графах можно рассматривать и мультипотоки — это расширение классической теории потоков в сетях, где одновременно рассматривается несколько типов потоков (товаров, ресурсов), каждый со своим источником и стоком.

Формулировка

Как вместить несколько потоков разного рода в одну сеть так, чтобы суммарный поток через каждое ребро не превышал его пропускную способность?

Для графа $G = (V, E)$ с пропускными способностями $c(u, v)$ определяется поток для каждого товара k как $f_k(u, v)$ удовлетворяющий:

- $\sum\limits_k f_k(u, v) \leq c(u, v)$

- Закон сохранения: $\sum\limits_{u} f_k(u,v) = \sum\limits_{w} f_k(v,w)$ для вершин, кроме источника и стока товара k.
- Поток $f_k(u,v)$

Применение

Все те же задачи оптимизации распределения разных ресурсов. Например, распределения товаров по одной логичестической сети или производственные процессы с несколькими потоками сырья и продуктов.

В таких задачах часто используется принцип разделения потоков, когда мультипоток разбивается на несколько сетей, в которых требуется определить максимальный поток.