Сети. Потоки в сетях

Толстов Роберт Сергеевич,
Рудяк Артём Станиславович.

Для ситуаций, когда **важна пропускная способность рёбер** была описана теория потоковых сетей.

Для ситуаций, когда **важна пропускная способность рёбер** была описана теория потоковых сетей.

В рамках данной лекции мы успеем:

Для ситуаций, когда **важна пропускная способность рёбер** была описана теория потоковых сетей.

В рамках данной лекции мы успеем:

- познакомиться с теоретическими основами;

Для ситуаций, когда **важна пропускная способность рёбер** была описана теория потоковых сетей.

В рамках данной лекции мы успеем:

- познакомиться с теоретическими основами;
- рассмотреть ключевые практические задачи теории потоков;

Для ситуаций, когда **важна пропускная способность рёбер** была описана теория потоковых сетей.

В рамках данной лекции мы успеем:

- познакомиться с теоретическими основами;
- рассмотреть ключевые практические задачи теории потоков;
- изучить алгоритмы для их решения.

Теоретические основы

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).

Теоретические основы

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).

В сетях выделяют две вершины: исток s и сток t.

Теоретические основы

Поток f в сети G — это функция $f : V \times V \to \mathbb{R}_{\geqslant 0}$

Теоретические основы

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);

Теоретические основы

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. $f(u, v) \leq c(u, v)$;

Теоретические основы

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. $f(u, v) \leq c(u, v)$;

Величина потока — это объём потока, выходящий из истока. $|f| = \sum\limits_{u \in V} f(s, v)$.

Теоретические основы

Закон сохранения потока. Для сети $G = \langle V, E \rangle$ с потоком f верно, что:

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

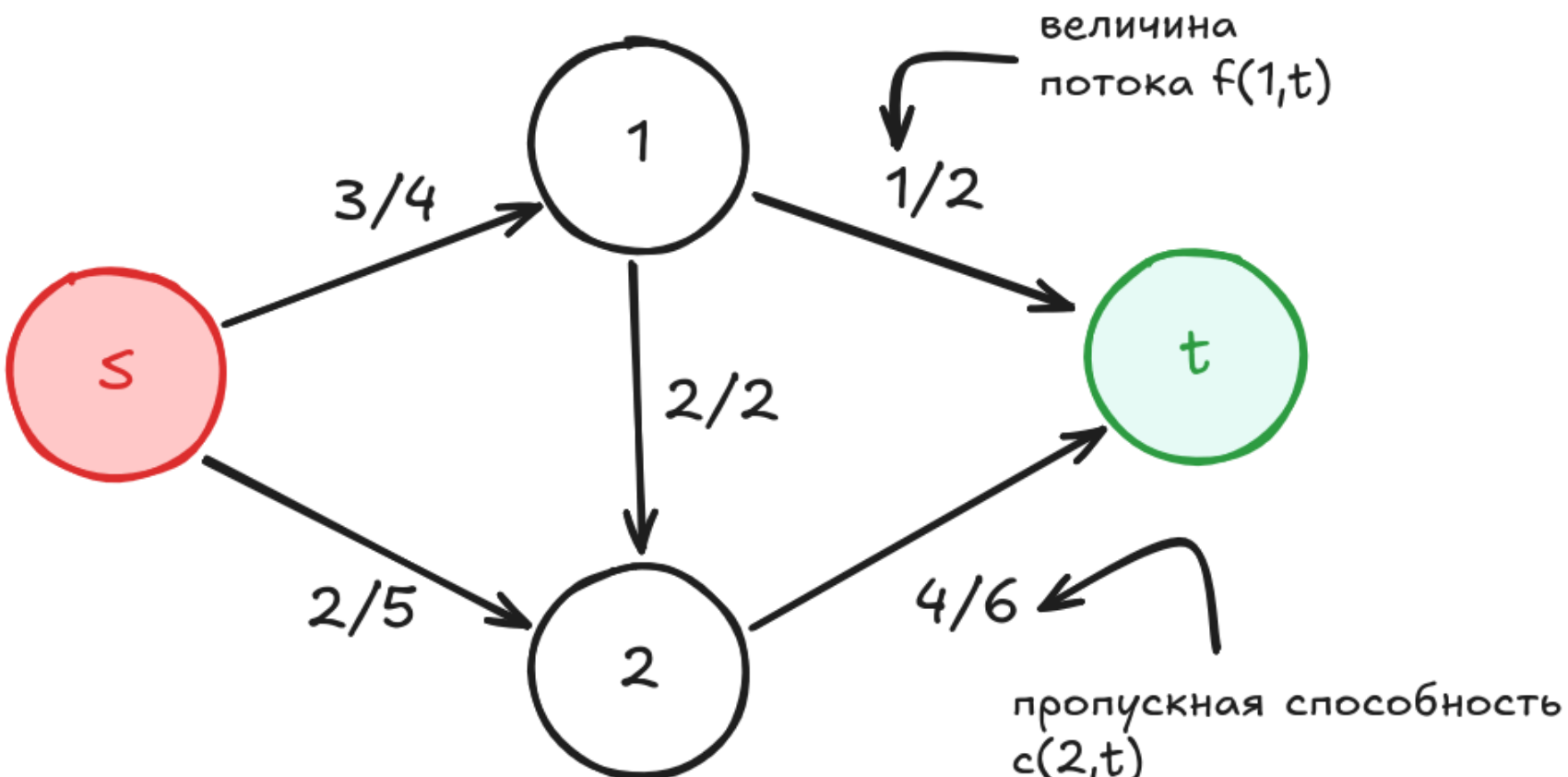
Теоретические основы

Закон сохранения потока. Для сети $G = \langle V, E \rangle$ с потоком f верно, что:

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

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

Теоретические основы



Интерактив на крутой суперприз

Вопросы

1. **Что такое источник?**

Вопросы

1. **Что такое источник?** Вершина, из которой исходит поток

Вопросы

1. **Что такое источник?** Вершина, из которой исходит поток
2. **Что такое сток?**

Вопросы

1. **Что такое источник?** Вершина, из которой исходит поток
2. **Что такое сток?** Вершина, в которую поток должен поступить

Вопросы

1. **Что такое источник?** Вершина, из которой исходит поток
2. **Что такое сток?** Вершина, в которую поток должен поступить
3. **Что такое пропускная способность?**

Вопросы

1. **Что такое источник?** Вершина, из которой исходит поток
2. **Что такое сток?** Вершина, в которую поток должен поступить
3. **Что такое пропускная способность?** Максимальный объём потока, разрешённый на дуге

Вопросы

1. **Что такое источник?** Вершина, из которой исходит поток
2. **Что такое сток?** Вершина, в которую поток должен поступить
3. **Что такое пропускная способность?** Максимальный объём потока, разрешённый на дуге
4. **Что такое поток?**

Вопросы

1. **Что такое источник?** Вершина, из которой исходит поток
2. **Что такое сток?** Вершина, в которую поток должен поступить
3. **Что такое пропускная способность?** Максимальный объём потока, разрешённый на дуге
4. **Что такое поток?** Реальное значение ресурса, которое протекает по дуге.

Задача максимального потока

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.

Задача максимального потока

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.

Далее рассмотрим различные алгоритмы для решения этой задачи.

Алгоритм Форда-Фалкерсона

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.

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.

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 $c_f(u, v) = c(u, v) - f(u, v)$

Алгоритм Форда-Фалкерсона

Алгоритм Форда-Фалкерсона

1. Все потоки выставляем в 0.

Алгоритм Форда-Фалкерсона

1. Все потоки выставляем в 0.
2. В остаточной сети находим любой путь из источника в сток. Если нет, то останавливаемся

Алгоритм Форда-Фалкерсона

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

Алгоритм Форда-Фалкерсона

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 $c_{\min}$

Алгоритм Форда-Фалкерсона

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.

Алгоритм Форда-Фалкерсона

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.

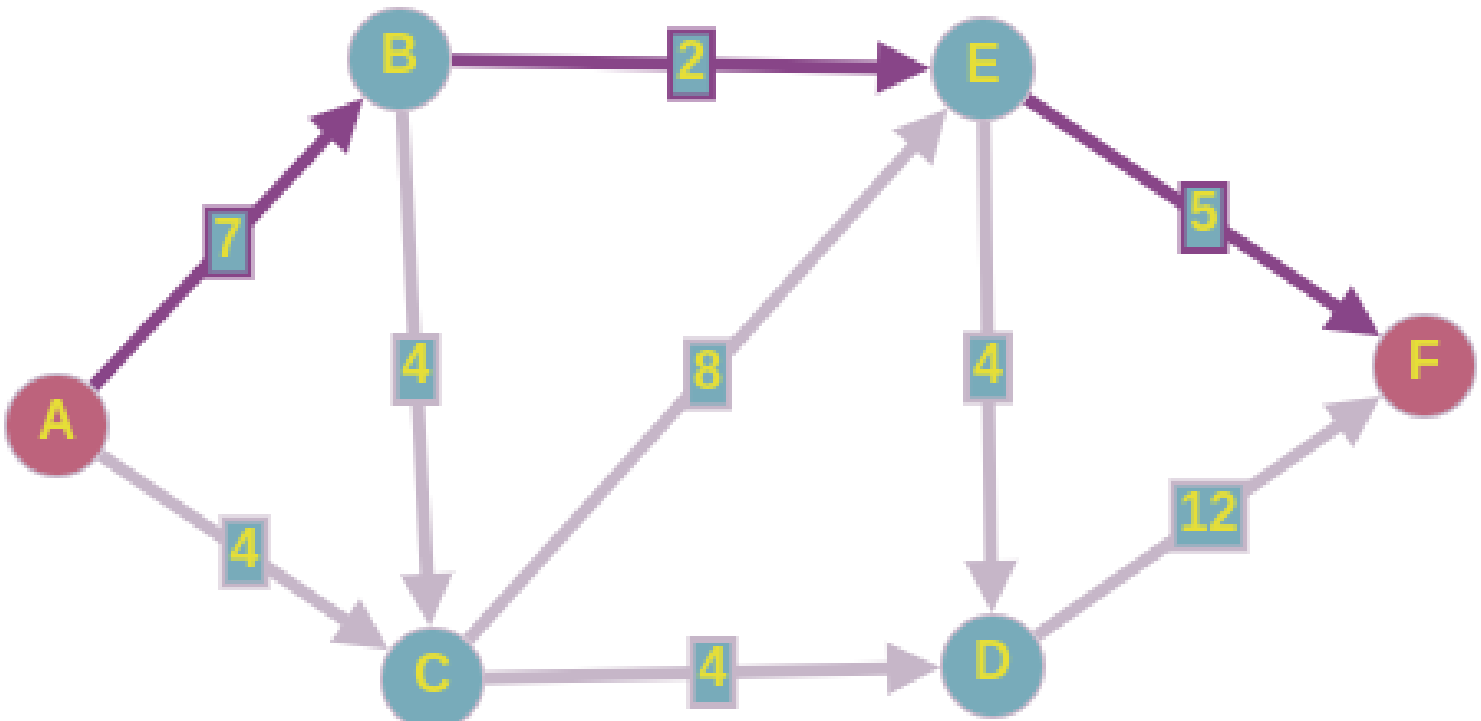
Алгоритм Форда-Фалкерсона

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.

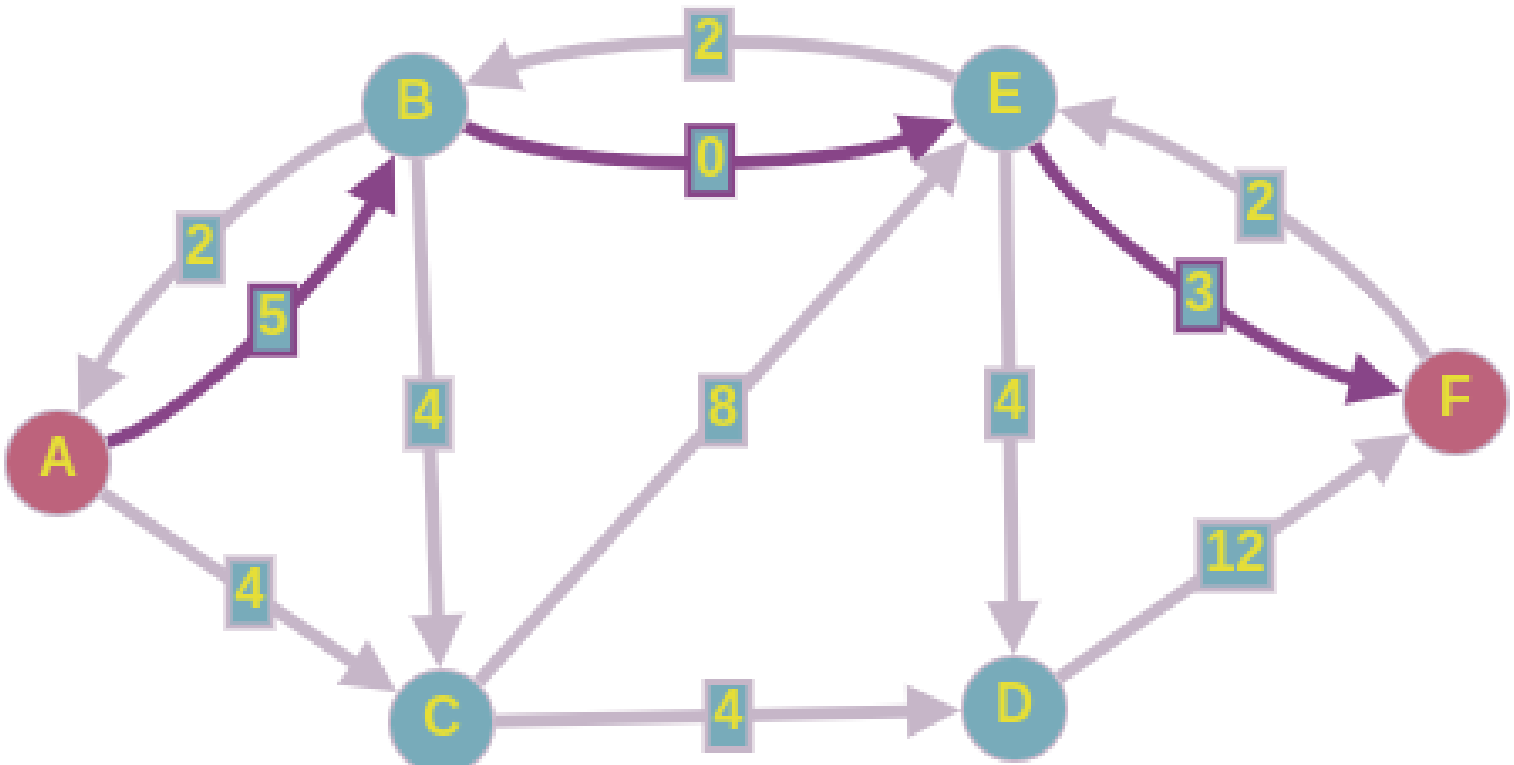
Алгоритм Форда-Фалкерсона

```
1  func FordFulkerson(G, s, t):
2    для всех рёбер (u, v) в G:
3      f(u, v) := 0
4
5    пока существует увеличивающий путь P из s в t в остаточной сети G_f:
6      delta := min(c_f(u, v), (u, v)) // бутылочное горлышко
7      для (u, v) в P:
8        f(u, v) := f(u, v) + delta // увеличиваем по прямому
9        f(v, u) := f(v, u) - delta // уменьшаем по обратному
10   вернуть сумму потоков, выходящих из s
```

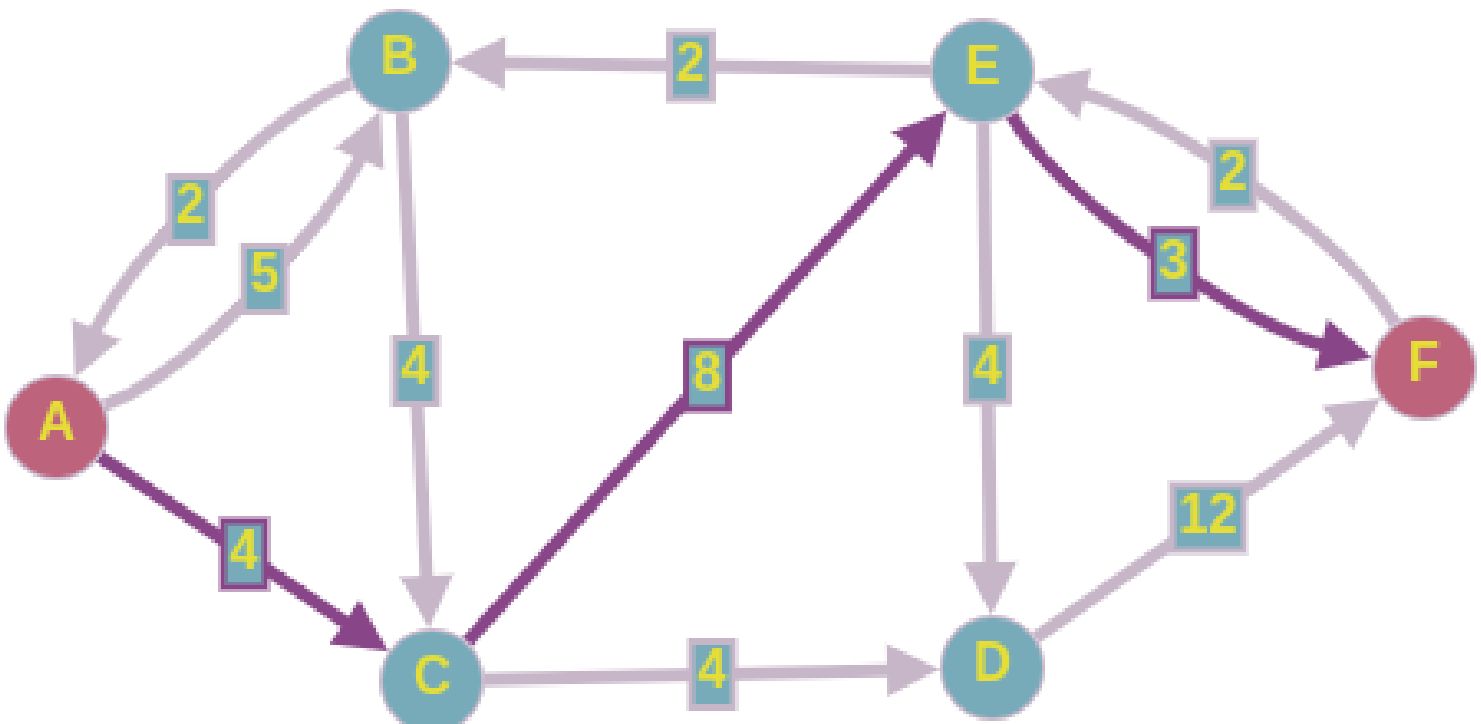
Алгоритм Форда-Фалкерсона



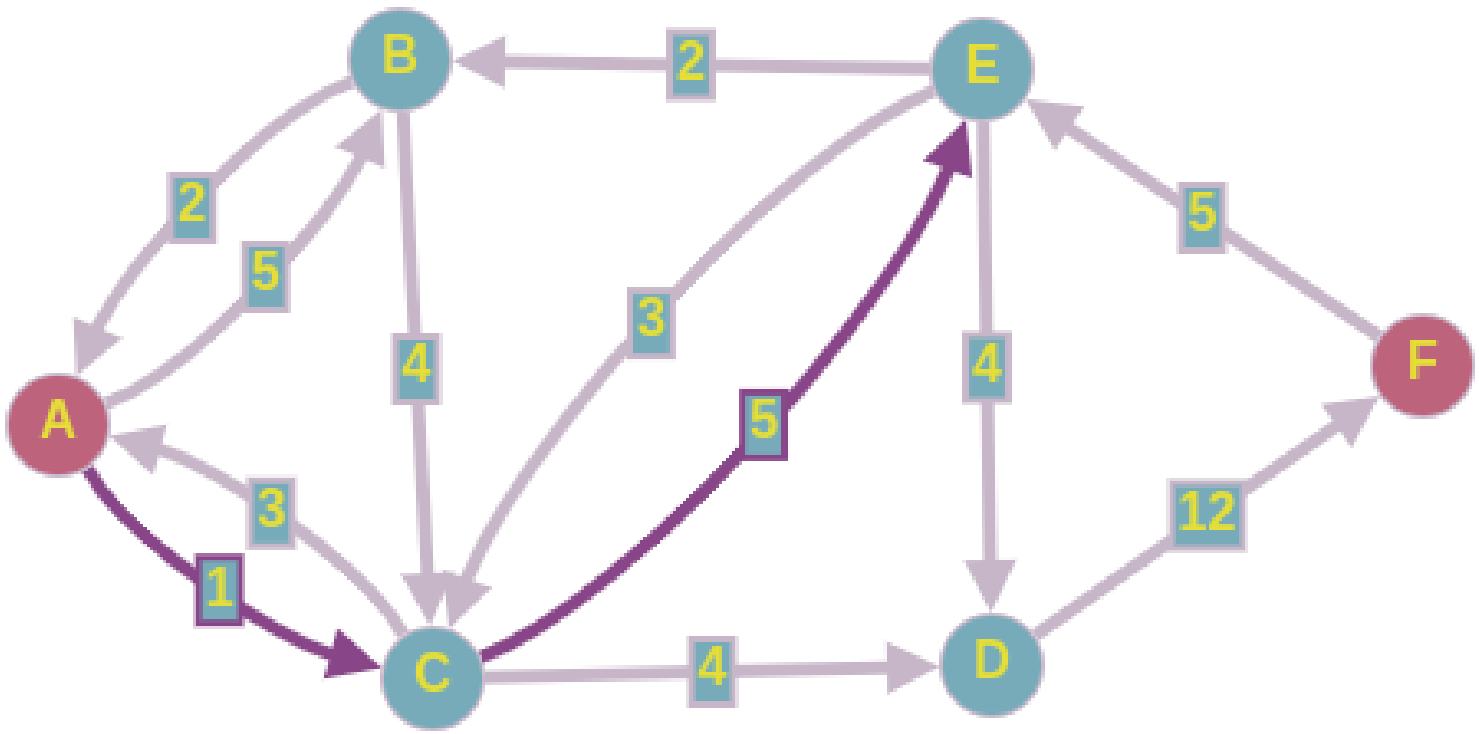
Алгоритм Форда-Фалкерсона



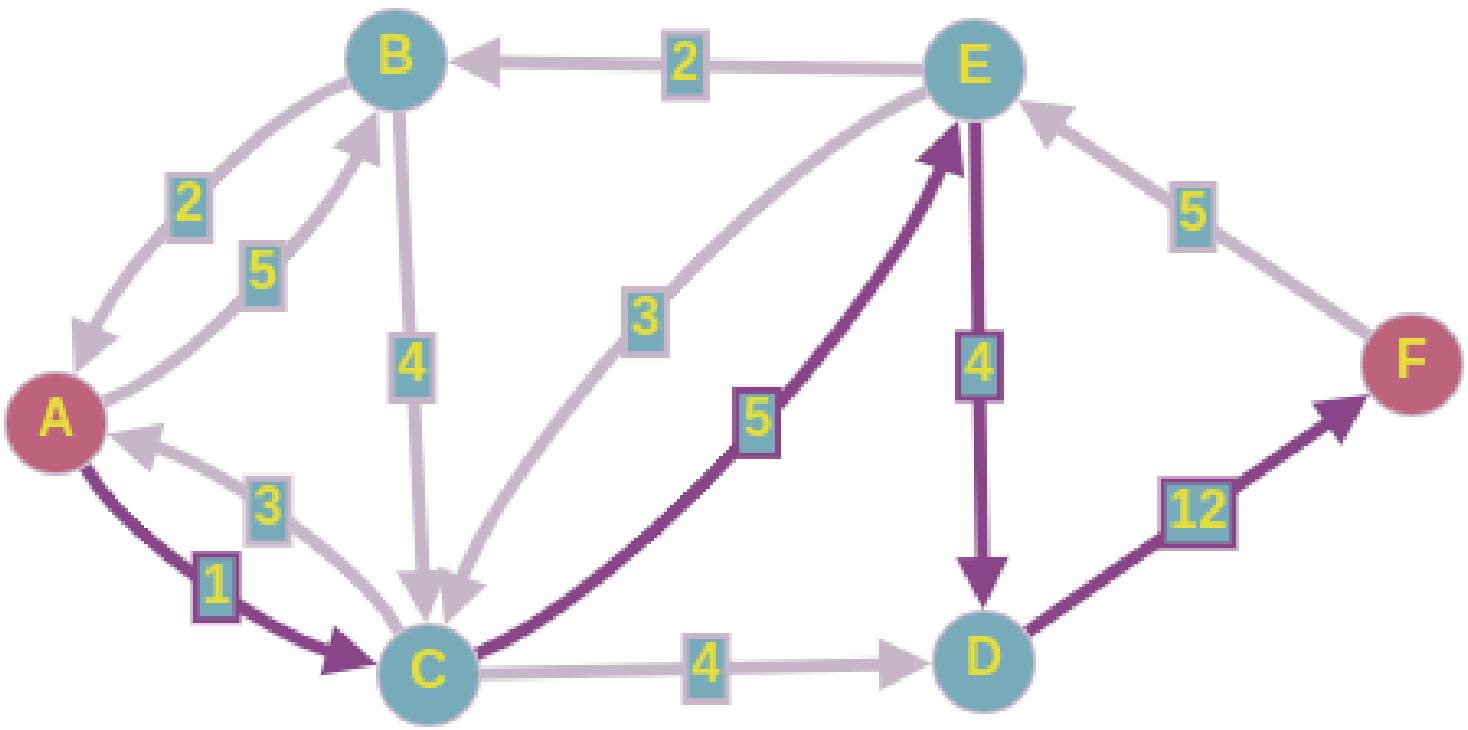
Алгоритм Форда-Фалкерсона



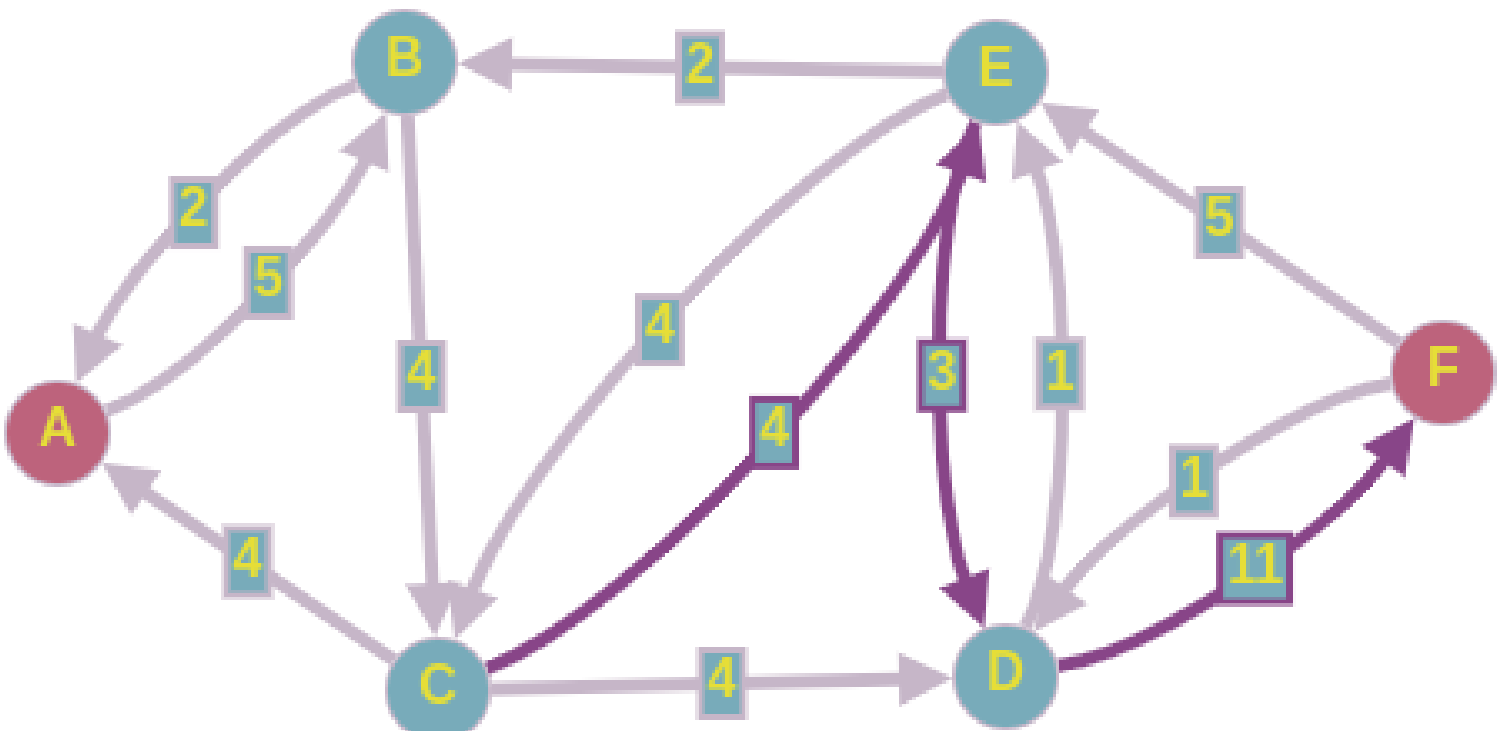
Алгоритм Форда-Фалкерсона



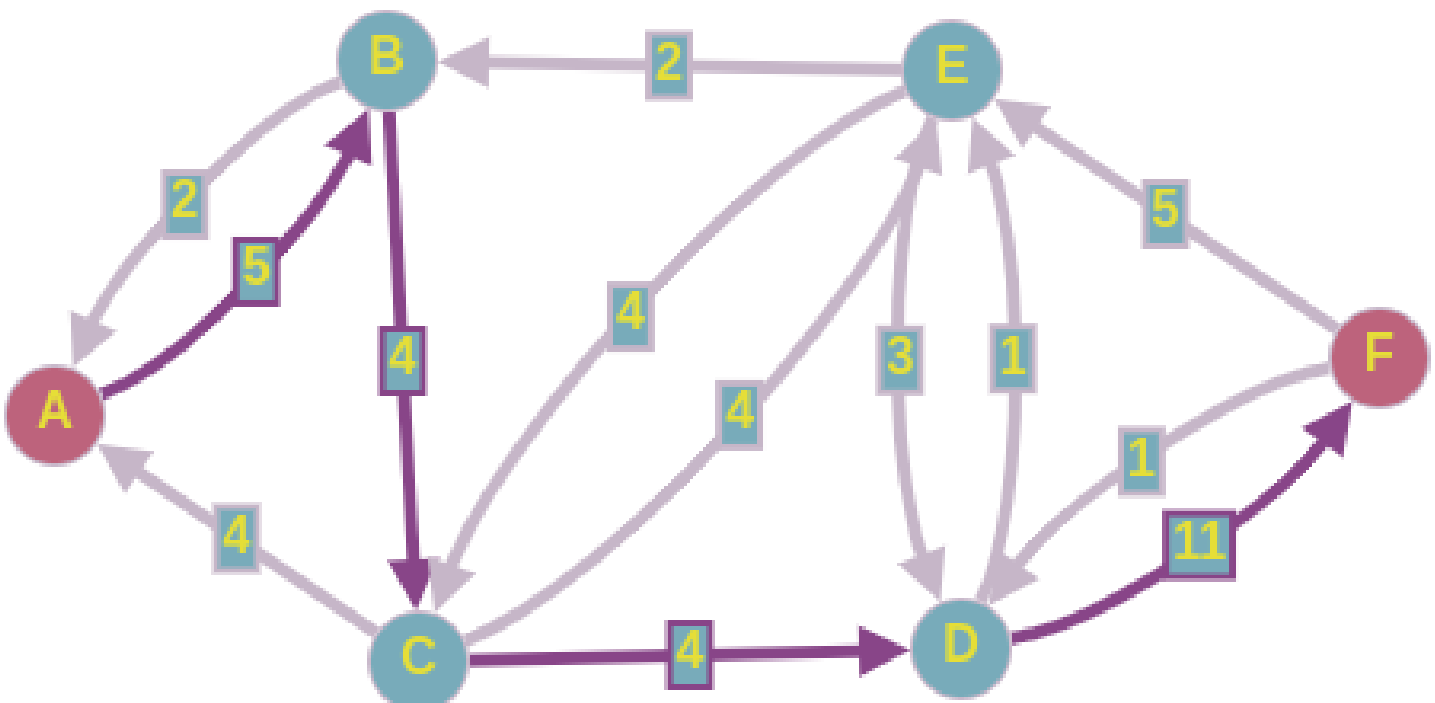
Алгоритм Форда-Фалкерсона



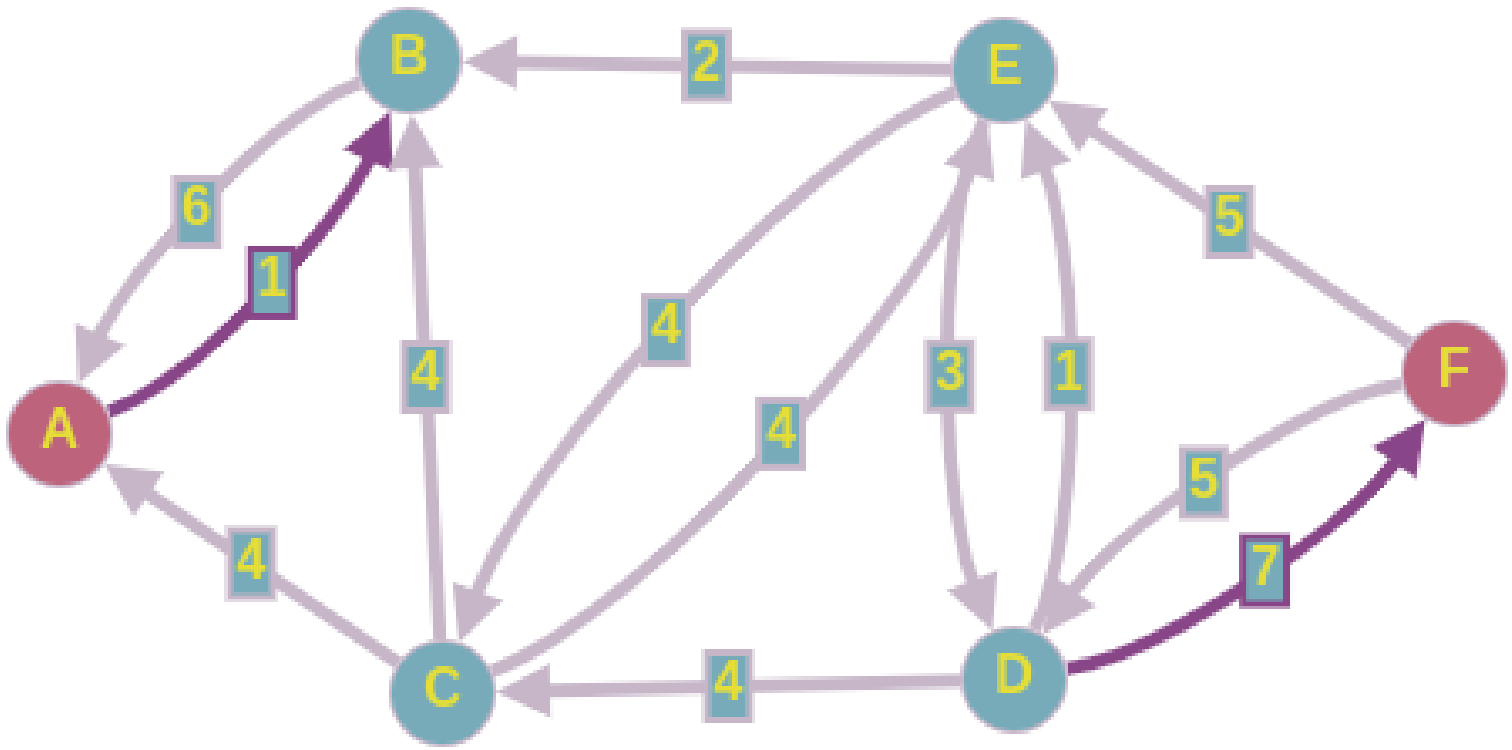
Алгоритм Форда-Фалкерсона



Алгоритм Форда-Фалкерсона



Алгоритм Форда-Фалкерсона



Недостатки

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.

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.

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.

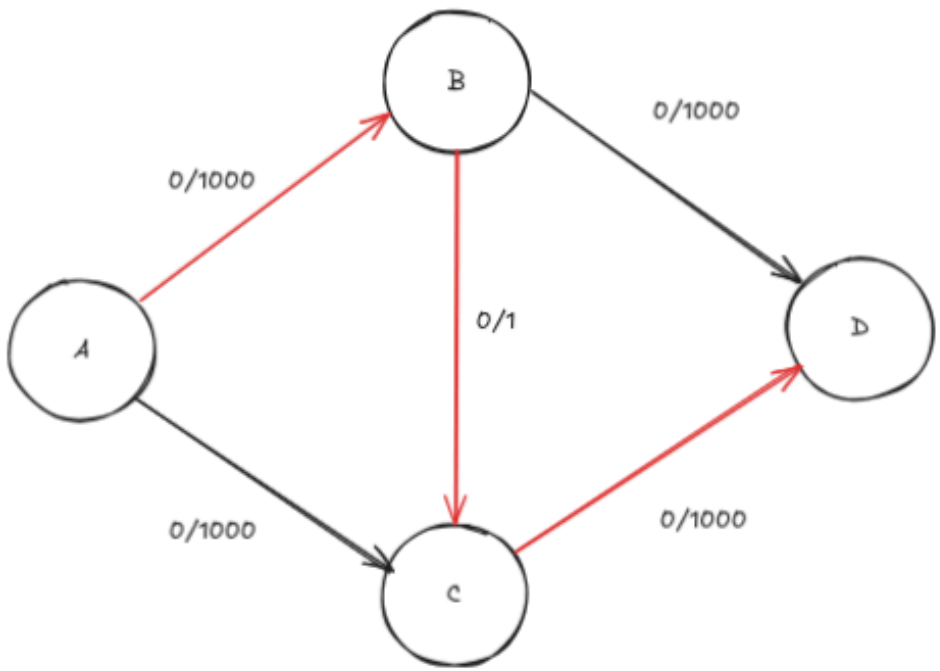
Все эти недостатки привели к появлению новых алгоритмов, среди которых алгоритм Эдмондса-Карпа.

Алгоритм Эдмондса-Карпа

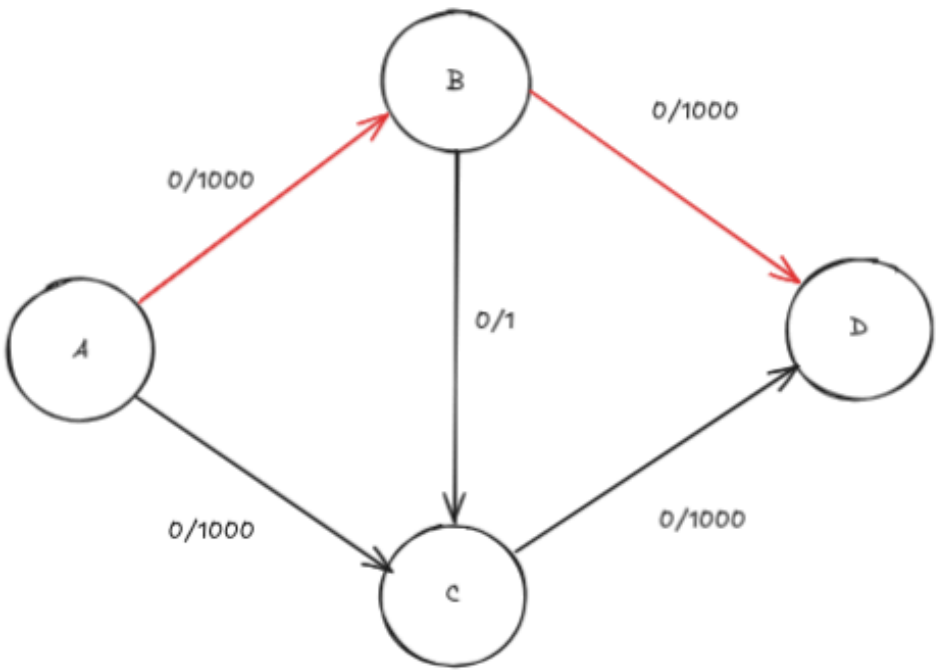
Алгоритм Эдмондса-Карпа

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 соста вляет $O(VE^2)$

Алгоритм Эдмондса-Карпа



Насыщение за 2000 итераций



Насыщение за 2 итерации

Алгоритм масштабирования потока

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.

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.

Алгоритм масштабирования потока

```
1  func FordFulkerson(G, s, t):
2    для всех рёбер (u, v) в G:
3      f(u, v) := 0
4
5    пока существует увеличивающий путь P из s в t в остаточной сети G_f:
6      delta := min(c_f(u, v), (u, v)) // бутылочное горлышко
7      для (u, v) в P:
8        f(u, v) := f(u, v) + delta // увеличиваем по прямому
9        f(v, u) := f(v, u) - delta // уменьшаем по обратному
10   вернуть сумму потоков, выходящих из s
```

Алгоритм масштабирования потока

```
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
```

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

А зачем всё это?

Применение алгоритмов

Области применения алгоритмов поиска максимального потока довольно очевидны:

Применение алгоритмов

Области применения алгоритмов поиска максимального потока довольно очевидны:

- Транспортные сети. Например, при перевозке товаров от поставщика между распределительными пунктами до заказчика.

Применение алгоритмов

Области применения алгоритмов поиска максимального потока довольно очевидны:

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

Применение алгоритмов

Области применения алгоритмов поиска максимального потока довольно очевидны:

- 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.

Немного о мультипотоках

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.

Немного о мультипотоках

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.

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

Немного о мультипотоках

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

Немного о мультипотоках

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

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

Немного о мультипотоках

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

- $\sum\limits_k f_k(u, v) \leq c(u, v)$
- $\sum\limits_u f_k(u, v) = \sum\limits_w f_k(v, w)$ для вершин, кроме источника и стока товара k

Немного о мультипотоках

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

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

А это зачем?

Применение мультипотоков

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.

Итоги

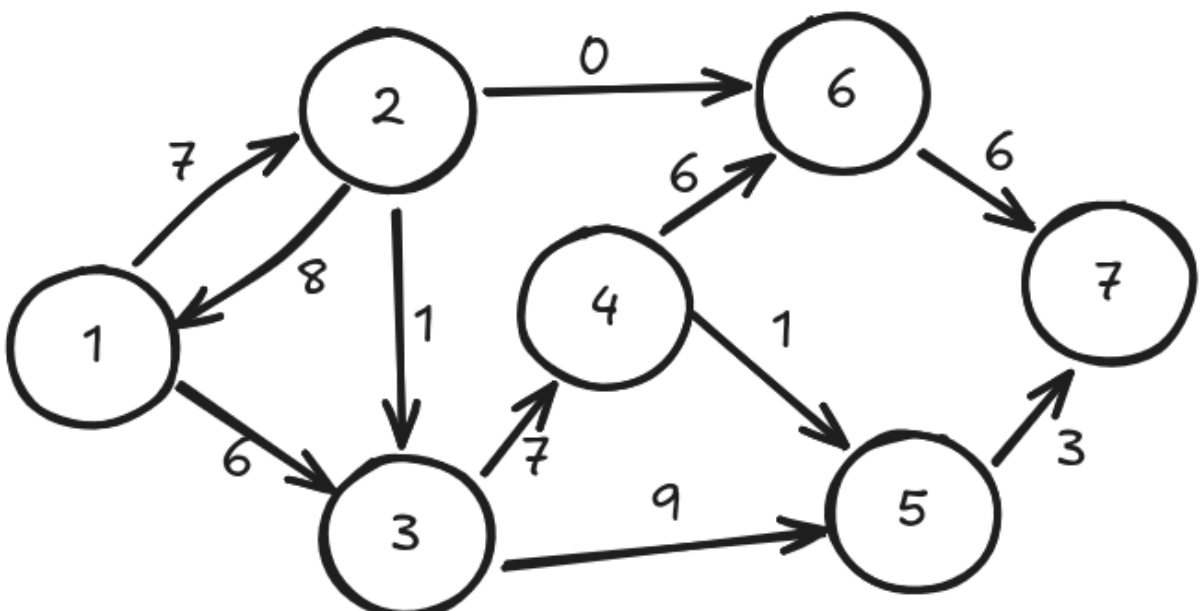
Домашнее задание

Домашнее задание

1. Реализовать на любом языке программирование один из алгоритмов в данной лекции

Домашнее задание

1. Реализовать на любом языке программирование один из алгоритмов в данной лекции
2. Найти максимальный поток в графе:



Домашнее задание по графам:

По два примера гетерогенного и гомогенного графа. Взять по одному гетерогенному и гомогенному графу и привести пример задачи на уровне всего графа, вершин, ребер

Спасибо за внимание. Вопросы?

Приз BMW x5 за лучший вопрос