Основы теории графов

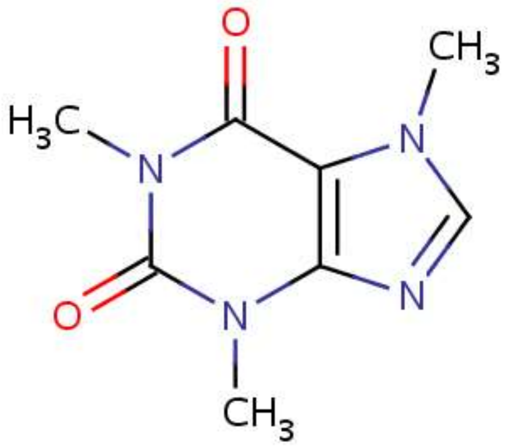
осень 2013

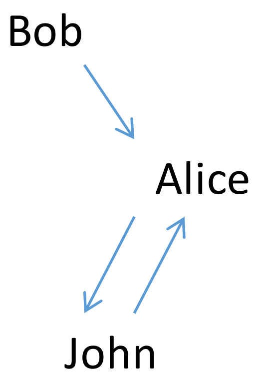
Александр Дайняк

www.dainiak.com

Что такое граф?

- Неформально, граф — набор объектов и связей между парами этих объектов





Что такое граф?

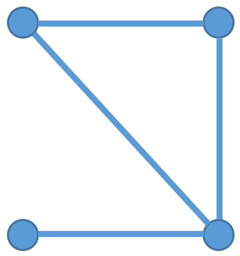
- Формально,
  граф — это **пара множеств** (V, E),
  где E — это множество пар элементов V
- Например,
  $$\mathrm{V} = \{Alice, Bob, John\}$$
  $$\mathrm{E} = \{(Bob, Alice), (John, Alice), (Alice, John)\}$$
- Элементы множеств V и E называют *вершинами* и *рёбрами* графа соответственно
- Если пары в E упорядоченные, то говорят об *ориентированном* графе (*орграфе*), а элементы E называют *дугами*

Немного терминологии

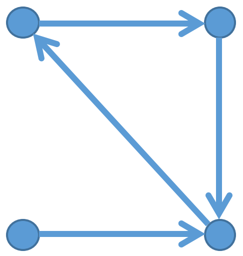
- Множества вершин и рёбер графа G обозначаются $V(G)$ и $E(G)$ соответственно
- Обозначаем $|G| \coloneqq |V(G)|$ и $\|G\| \coloneqq |E(G)|$
- Ребро, соединяющее вершины u и v, обозначается через uv
- Если $u, v \in V(G)$ и $uv \in E(G)$, то вершины u и v называются *смежными* (в графе G)
- Вершина, принадлежащая ребру, называется *концом* этого ребра. При этом говорят, что данная вершина и ребро *инцидентны* друг другу

Какие бывают графы

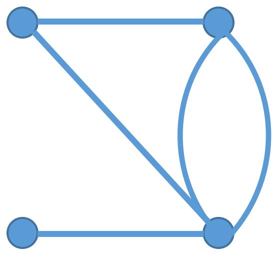
- Если во множестве E есть повторяющиеся пары, то говорят о *мультиграфе*, а сами эти пары называют *кратными* рёбрами/дугами
- Если в E есть пары, оба элемента которых совпадают, то говорят о *псевдографе*, а сами эти пары называют *петлями*



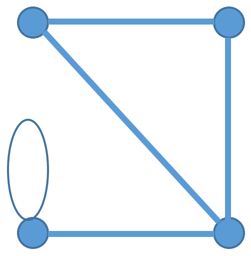
граф



орграф



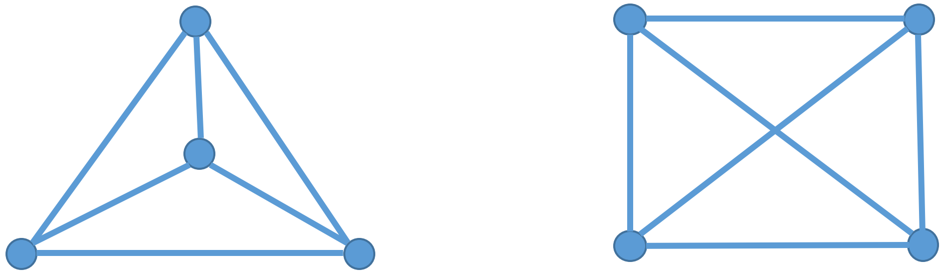
мультиграф



псевдограф

Графы и их представления

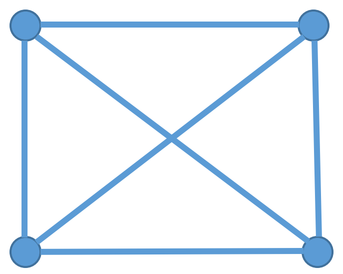
- Следует чётко различать сам граф (абстрактный объект, пара множеств) и его представление (например, изображение на плоскости). Графы существуют вне зависимости от их изображений. Например, граф
$$\big(\{1,2,3,4\},\{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}\big)$$
можно по-разному «изобразить»:



Графы и их представления

- И наоборот, два разных графа могут «структурно» представлять собой одно и то же: графы
$$\left(\{1,2,3,4\},\left\{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\right\}\right)$$
и
$$\left(\{A,B,C,D\},\left\{\{A,B\},\{C,D\},\{A,C\},\{B,D\},\{D,A\},\{B,D\}\right\}\right)$$

имеют идентичную структуру: *четыре вершины, каждые две из которых смежны*

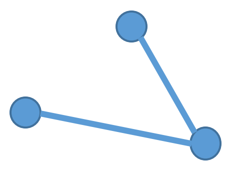


Изоморфизм графов

- Графы, которые неотличимы со структурной точки зрения, называются *изоморфными*
- Формально, графы G' и G'' изоморфны, если существует пара биекций
$$\phi: V(G') \leftrightarrow V(G'')$$
$$\psi: E(G') \leftrightarrow E(G'')$$
такая, что для любого ребра $uv \in E(G')$
$$\psi(uv) = \phi(u)\phi(v)$$
- Сама биекция ϕ называется *изоморфизмом* между G' и G''

Изоморфизм графов

- Иначе говоря, G' и G'' изоморфны, если их вершины можно «занумеровать» так, что вершины с одинаковыми «номерами» либо смежны и в G', и в G'', либо несмежны в обоих этих графах
- Пример:
  - $V(G') = \{1,2,3\},\ E(G') = \{\{1,2\},\{1,3\}\}$
  - $V(G'') = \{x,y,z\},\ E(G'') = \{\{y,z\},\{z,x\}\}$
  - Изоморфизм: $1 \leftrightarrow z, 2 \leftrightarrow x, 3 \leftrightarrow y$



Изоморфизм графов

- Поскольку изоморфные графы с точки зрения структуры идентичны, мы часто будем считать их одним и тем же графом
- Любая характеристика графа (числовая или качественная), зависящая лишь от структуры графа (т.е. равная у любой пары изоморфных графов), называется *инвариантом* графа
- Пример: количества вершин и рёбер графа являются инвариантами

Изоморфизм графов

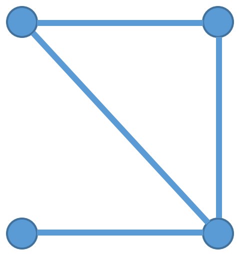
Если требуется определить, являются ли два данных графа изоморфными, то

- для доказательства неизоморфности графов надо указать инвариант, значение которого различается у этих графов
- для доказательства изоморфности нужно указать, какая вершина первого графа соответствует какой вершине второго

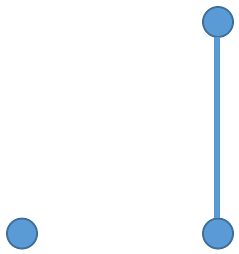
Подграфы

- Пусть $G = (V, E)$ и $G' = (V', E')$ — графы.
- G' является *подграфом* графа G, если $V' \subseteq V$ и $E' \subseteq E$
- G' — *остовный* подграф, если $V' = V$
- G' — *порождённый* подграф графа G, если с каждой парой вершин он содержит и все инцидентные им рёбра:

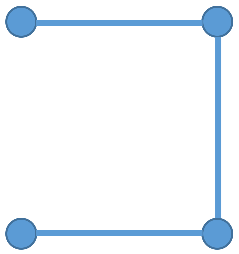
$$\forall u, v\ (uv \in E \text{ и } u, v \in V') \Rightarrow uv \in E'$$



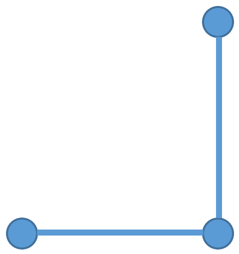
граф



подграф
(не являющийся
порождённым)



остовный
подграф



порождённый
подграф

Удаление/добавление вершин и рёбер

Пусть $G = (V, E)$.

- Для множества $V' \subseteq V$ через $(G - V')$ обозначаем подграф графа G, порождённый множеством $V \setminus V'$
- Для множества $E' \subseteq E$ через $(G - E')$ обозначаем граф $(V, E \setminus E')$
- Если G' — подграф G, то через $(G - G')$ обозначаем граф
$$\big(G - V(G')\big)$$
- Если e — пара вершин графа G, то $(G + e)$ обозначает граф
$$(V, E \cup \{e\})$$

Соседи. Степени вершин

- *Любая* вершина, смежная с вершиной v, называется *соседом* v
- Множество соседей v обозначают $N(v)$
- Для множества вершин A считаем $N(A) = \left(\bigcup_{v \in A} N(v)\right) \setminus A$
- *Степень* вершины — это количество рёбер, инцидентных этой вершине (в простом графе это равно $|N(v)|$)
- Степень вершины v обозначается $d(v)$ или $\deg v$

Степени вершин

В ориентированном графе:

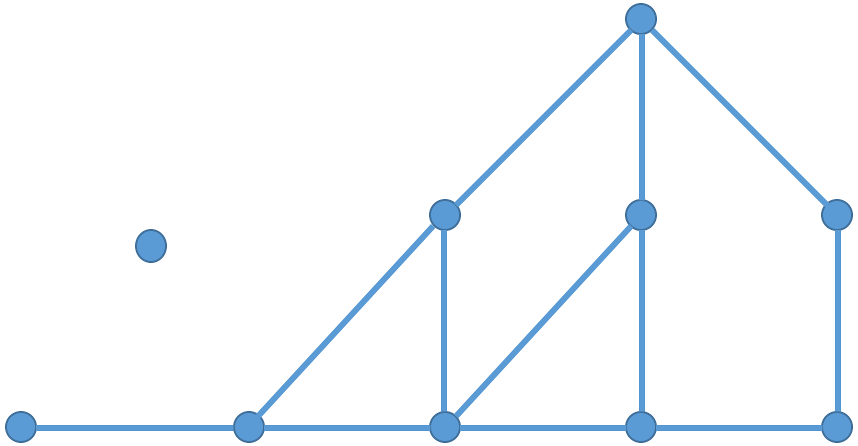
- Число входящих в v дуг обозначается через $d^{-}(v)$, называется *полустепенью захода*
- Число выходящих из v дуг обозначается через $d^{+}(v)$, называется *полустепенью исхода*

При этом

$$d(v) = d^{-}(v) + d^{+}(v)$$

Степени вершин

- *Степень вершины* — это количество инцидентных ей рёбер
- Вершина степени 0 — *изолированная*
- Вершина степени 1 — *висячая* или *лист*
- Вершина степени 2 — *проходная*



Инварианты, основанные на степенях вершин графа

- Максимальная степень вершины в графе G:
$$\Delta(G) = \max_{v \in V(G)} d(v)$$

- Минимальная степень вершины в графе G:
$$\delta(G) = \min_{v \in V(G)} d(v)$$

- Средняя степень вершин в графе G:
$$d(G) = \frac{1}{|G|} \cdot \sum_{v \in V(G)} d(v)$$

Теорема «о рукопожатиях»

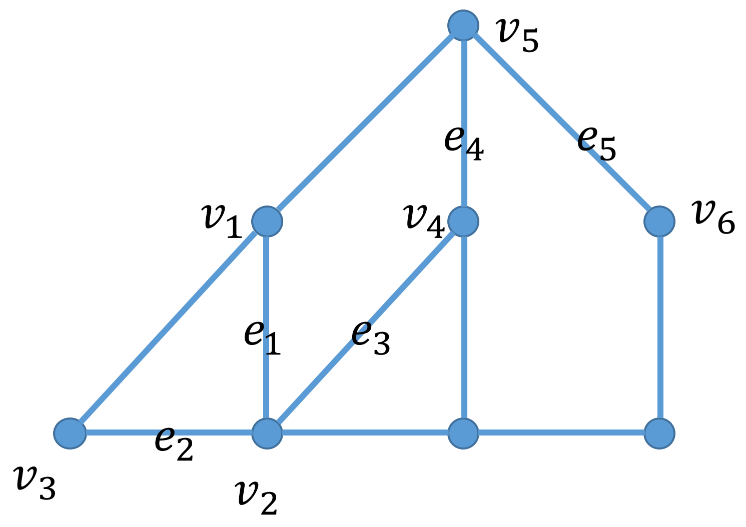
- В любом графе сумма степеней всех вершин равна удвоенному количеству рёбер:

$$\sum_{v \in V(G)} d(v) = 2 \cdot \|G\|$$

Маршруты

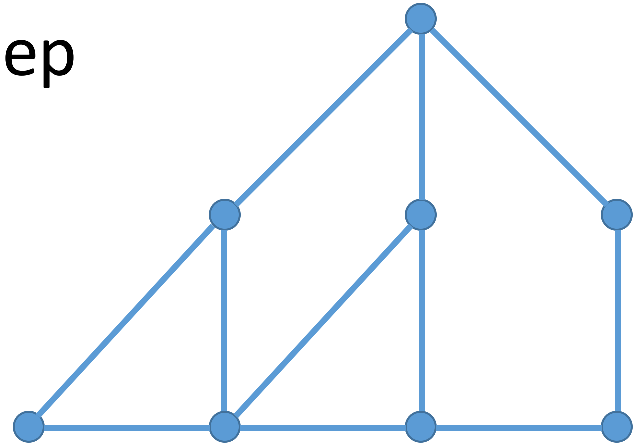
- *Маршрут* — последовательность вершин и рёбер графа (начало и конец — в вершинах),
в которой последовательные элементы инцидентны друг другу

Пример маршрута:
$v_1 e_1 v_2 e_2 v_3 e_2 v_2 e_3 v_4 e_4 v_5 e_5 v_6$



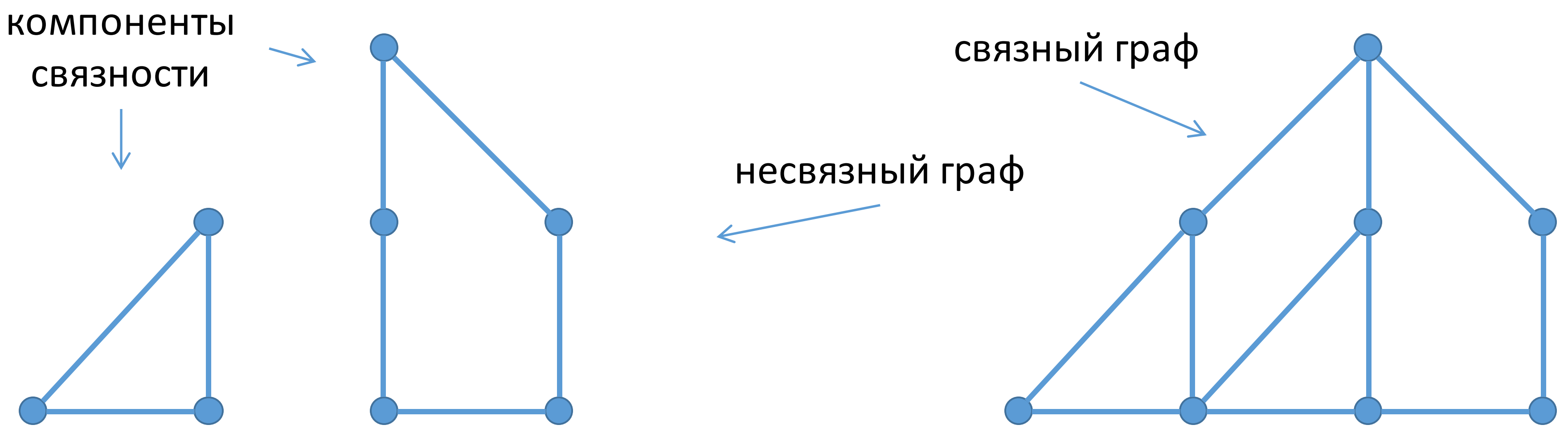
Пути, цепи и циклы

- *Цикл* — это замкнутый маршрут (т.е. начало совпадает с концом) без повторяющихся рёбер
- *Простой цикл* — это цикл без повторяющихся вершин
- *Путь* — это незамкнутый маршрут без повторений рёбер
- *Цепь* — путь без повторяющихся вершин
- *Длина* цикла/цепи — это количество рёбер



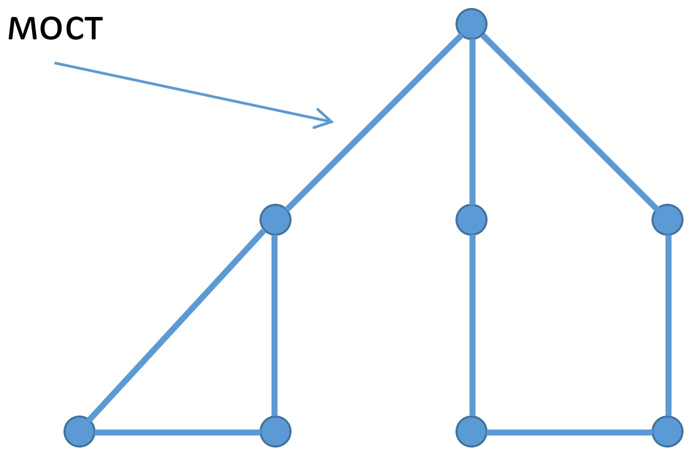
Связность

- *Связный граф* — это граф, в котором между любыми двумя вершинами существует путь
- *Компонента связности* графа — это его максимальный связный подграф



Связность

- *Мост* — это ребро, удаление которого приводит к графу с бо́льшим числом компонент связности
- *Точка сочленения* — вершина, удаление которой приводит к графу с бо́льшим числом компонент связности

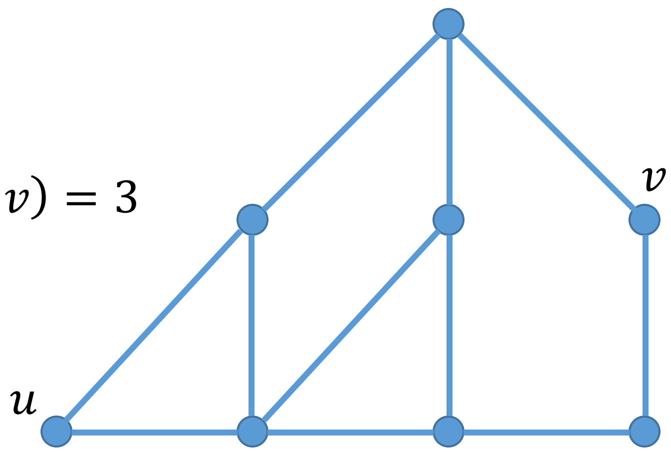


Расстояния в графе

- *Расстояние между парой вершин* u, v — это длина кратчайшей цепи, соединяющей эти вершины. Обозначение: $d(u, v)$
- *Диаметр* графа — это максимальное из расстояний между парами вершин. Обозначение:

$$\operatorname{diam}(G) = \max_{u,v} d(u, v)$$

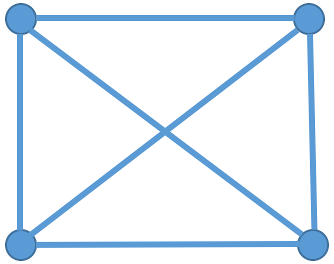
$d(G) = d(u, v) = 3$



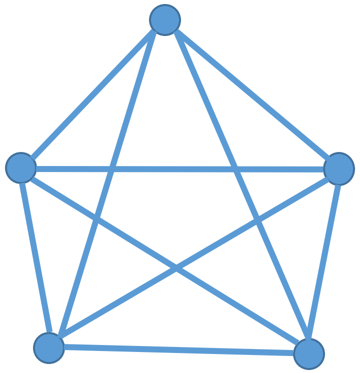
Полные и пустые графы

- *Полный граф* на n вершинах K_n — это граф, в котором есть все возможные рёбра (сколько?)

K_4



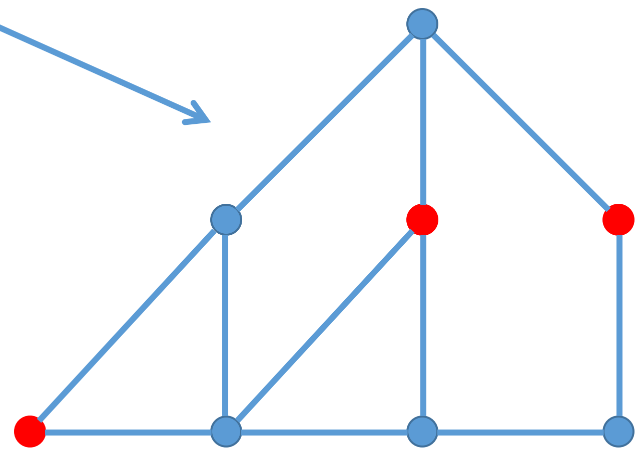
K_5



- *Пустой граф* — в котором нет ни одного ребра

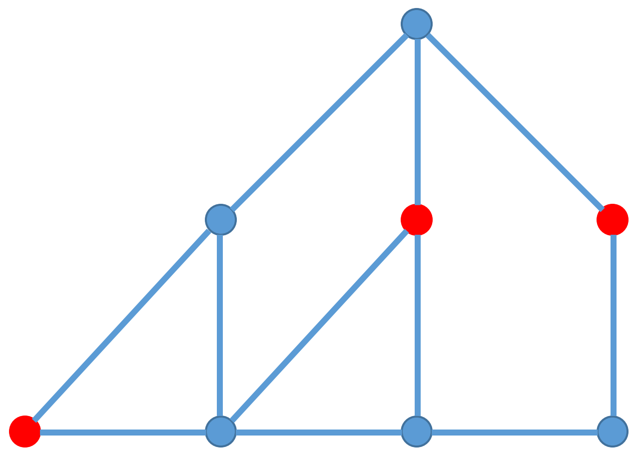
Независимые множества и клики

- *Клика* в графе — это полный подграф
- *Независимое множество* — это подмножество вершин, порождающее пустой подграф



Независимые множества и клики

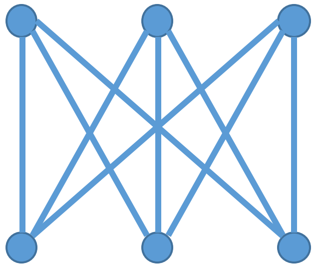
- *Число независимости* $\alpha(G)$ — это размер максимального независимого множества
- *Кликовое число* $\omega(G)$ — это максимальный размер клики в графе



Двудольные графы

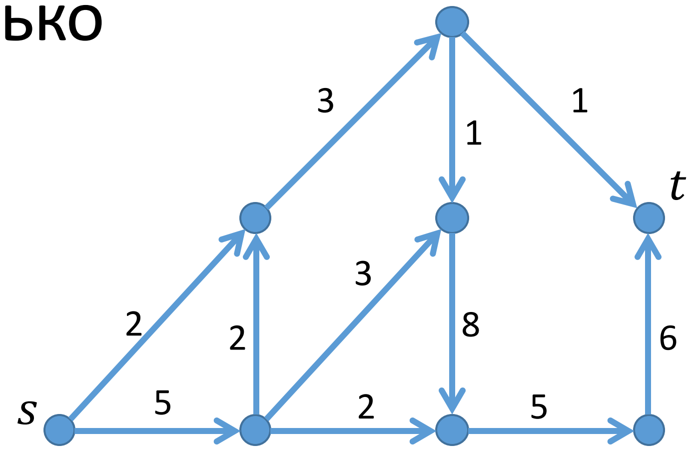
- *Двудольный* граф — это граф, вершины которого можно разбить на два независимых множества
- *Полный двудольный* граф $K_{m,n}$ — это двудольный граф со всеми возможными рёбрами между долями (m и n — мощности долей)

$K_{3,3}$



Потоки в сетях

- *Сеть* — это орграф, каждой дуге e которого приписано некоторое число $c(e) \geq 0$ (*пропускная способность*), и выделены две вершины: источник s и сток t
- Из s в t по дугам орграфа транспортируется некоторый «продукт»
- Задача: указать, по какой дуге сколько единиц продукта пустить



Потоки в сетях

Поток в сети — это функция f, которая каждой дуге e орграфа ставит в соответствие некоторое число $f(e)$, такое, что выполнены условия:

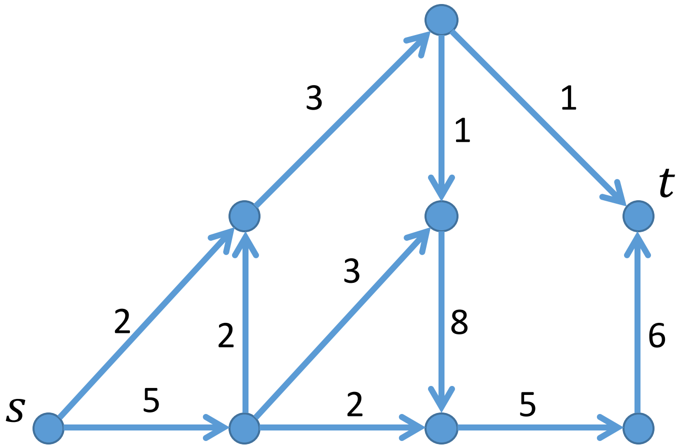
- $0 \le f(e) \le c(e)$ для любого e (считаем $c(e) = 0$ при $e \notin E$)
- $\sum_{e \text{ входит в } v} f(e) = \sum_{e \text{ выходит из } v} f(e)$
  для любой вершины v, не являющейся источником и стоком

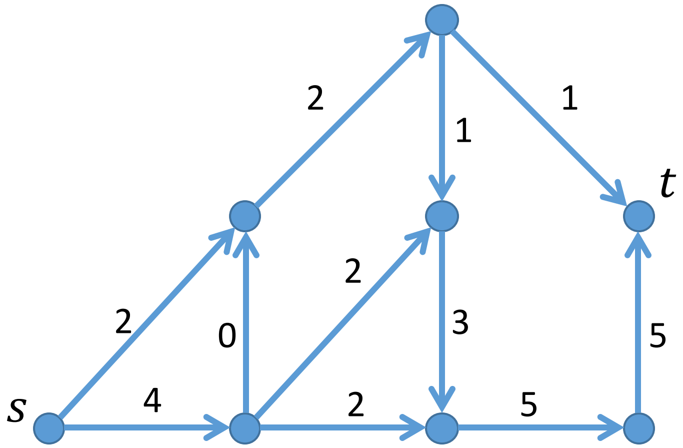
Задача: подобрать f, так, чтобы максимизировать *величину потока*:

$$\sum_{e \text{ выходит из } s} f(e) = \sum_{e \text{ входит в } t} f(e) \to \max$$

Потоки в сетях

• Пример потока в сети:





Верхние границы величины потока

- Величина потока не может быть больше суммарной пропускной способности дуг, выходящих из источника:

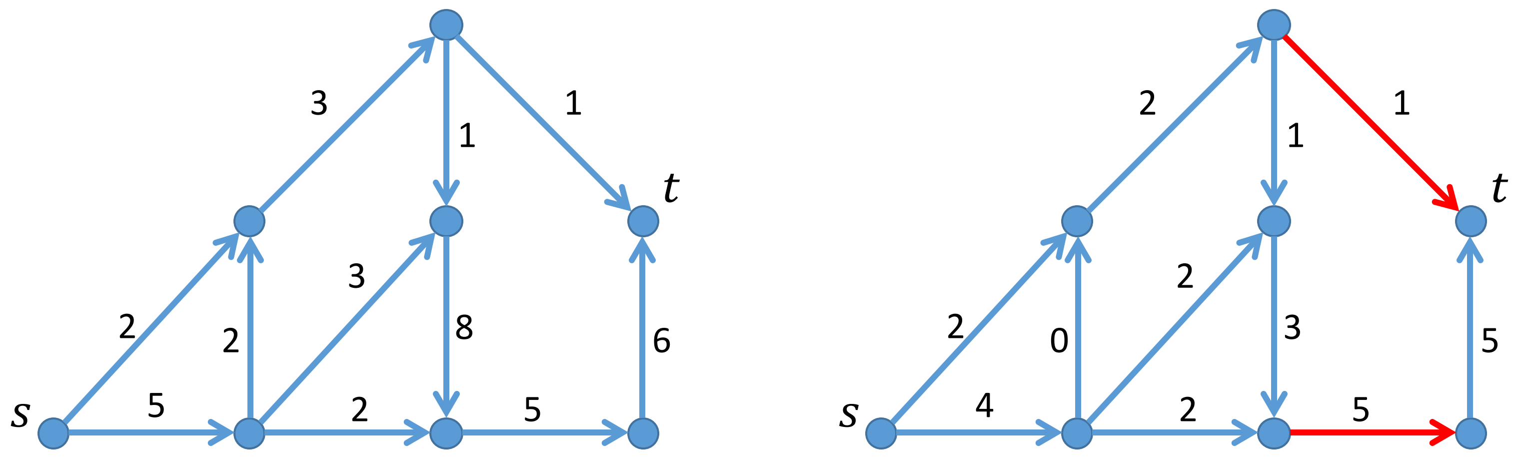
$$f_{\max} \leq \sum_{e \text{ выходит из } s} c(e)$$

- Величина потока не может быть больше суммарной пропускной способности дуг, входящих в сток:

$$f_{\max} \leq \sum_{e \text{ входит в } t} c(e)$$

Верхние границы величины потока

- Ни одна из указанных оценок не точна:

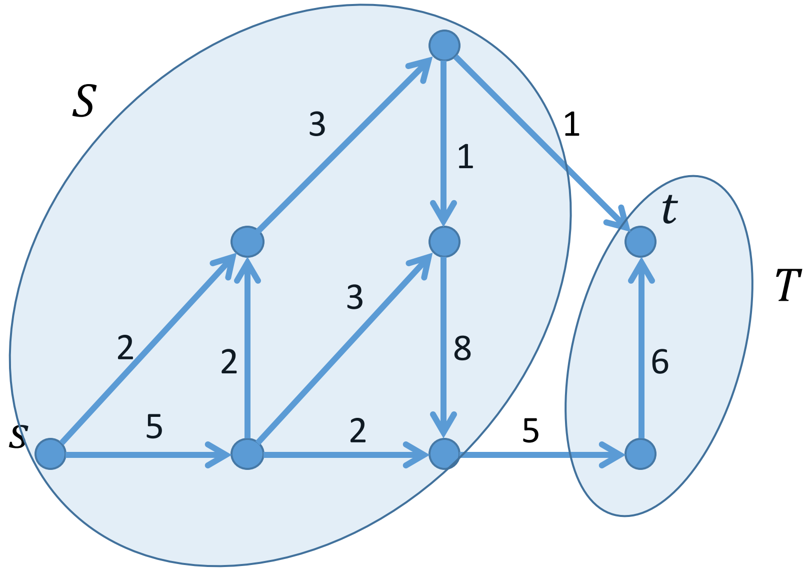


- Но их можно обобщить, получив точную…

Верхние границы величины потока

- Разрез в сети — это разбиение вершин сети на две части: $V = S \sqcup T$, где $S \ni s$ и $T \ni t$
- Пропускная способность разреза — это сумма пропускных способностей дуг, ведущих из S в T:

$$\sum_{e \text{ ведёт из } S \text{ в } T} c(e)$$



Верхние границы величины потока

- Итак, величина любого потока не больше пропускной способности любого разреза, а значит и
$$f_{\max} \leq c_{\min},$$
где $c_{\min}$ — пропускная способность минимального разреза в сети

Теорема Форда—Фалкерсона

Теорема. Имеет место равенство

$$f_{\max} = c_{\min},$$

то есть всегда можно указать поток в сети, достигающий верхней границы.

Алгоритм Форда—Фалкерсона

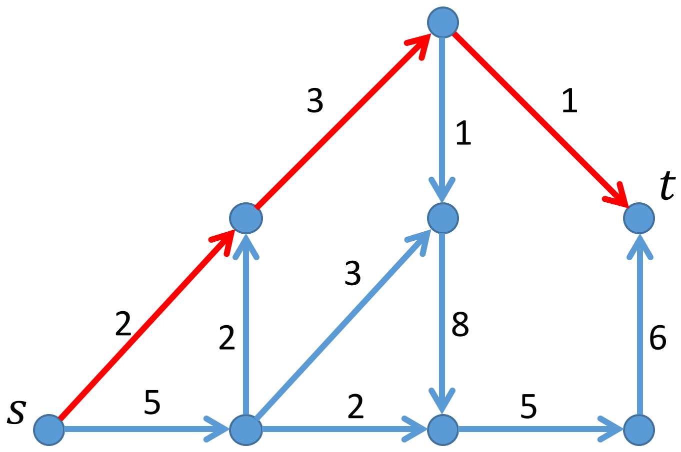
Алгоритм Форда—Фалкерсона строит максимальный поток в сети с помощью последовательных улучшений:

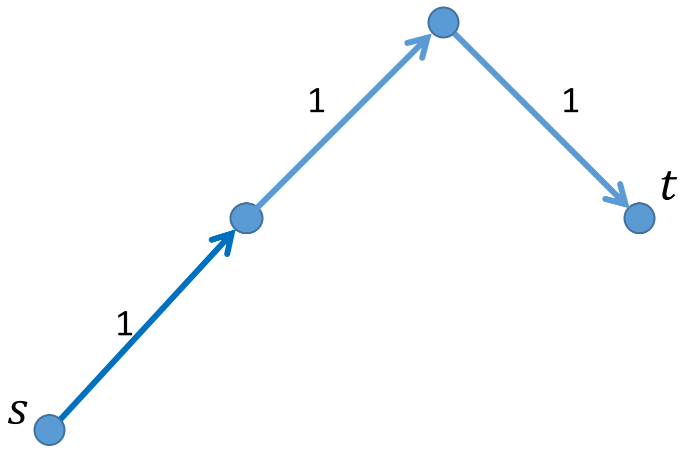
- Начинаем с произвольного потока
- Пока можно улучшить — улучшаем
- Если нельзя улучшить — останавливаемся

Алгоритм Форда—Фалкерсона

- Начинаем с произвольного потока

С помощью поиска в ширину по сети можно проверить, существует ли в ней ненулевой поток:

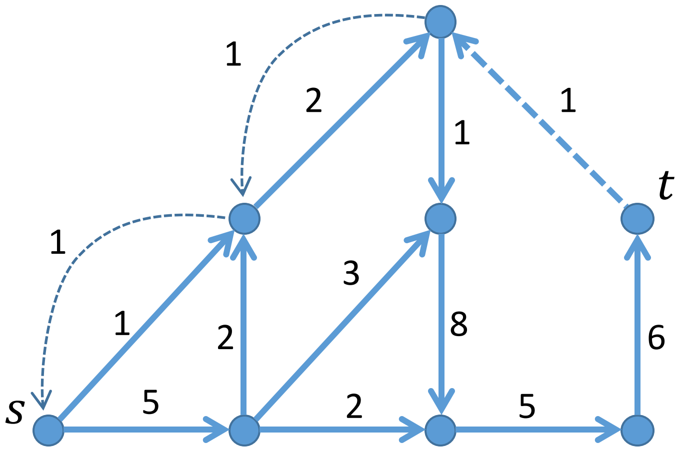


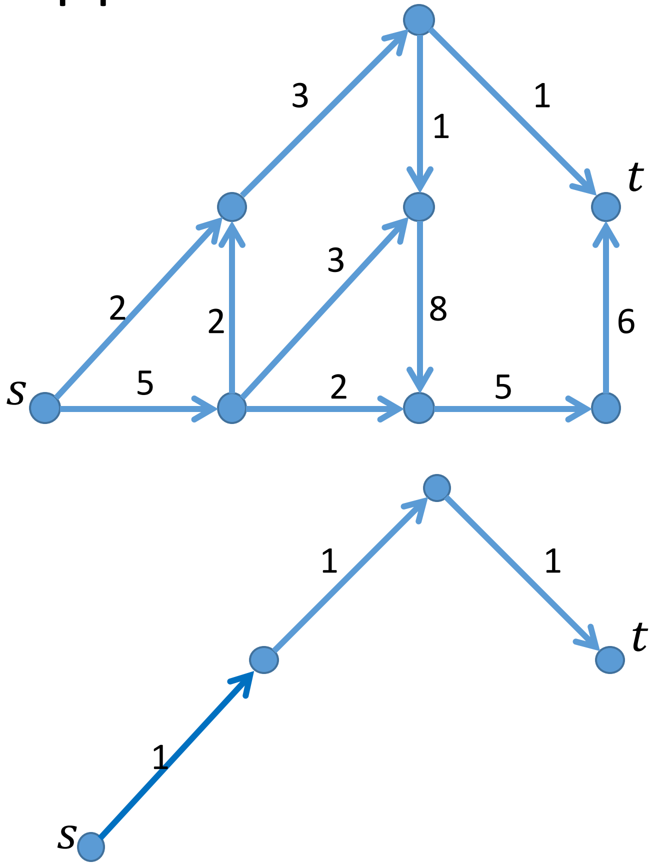


Алгоритм Форда—Фалкерсона

- Пока поток можно улучшить — улучшаем

Пусть в сети уже построен некоторый поток. Для этой сети и этого потока строим *остаточную сеть*:





Алгоритм Форда—Фалкерсона

Пусть в сети уже построен некоторый поток. Для этой сети и этого потока строим *остаточную сеть*. Если в исходной сети пропускная способность дуги e была c_0, и по этой дуге шёл поток f_0, то в остаточной сети считаем

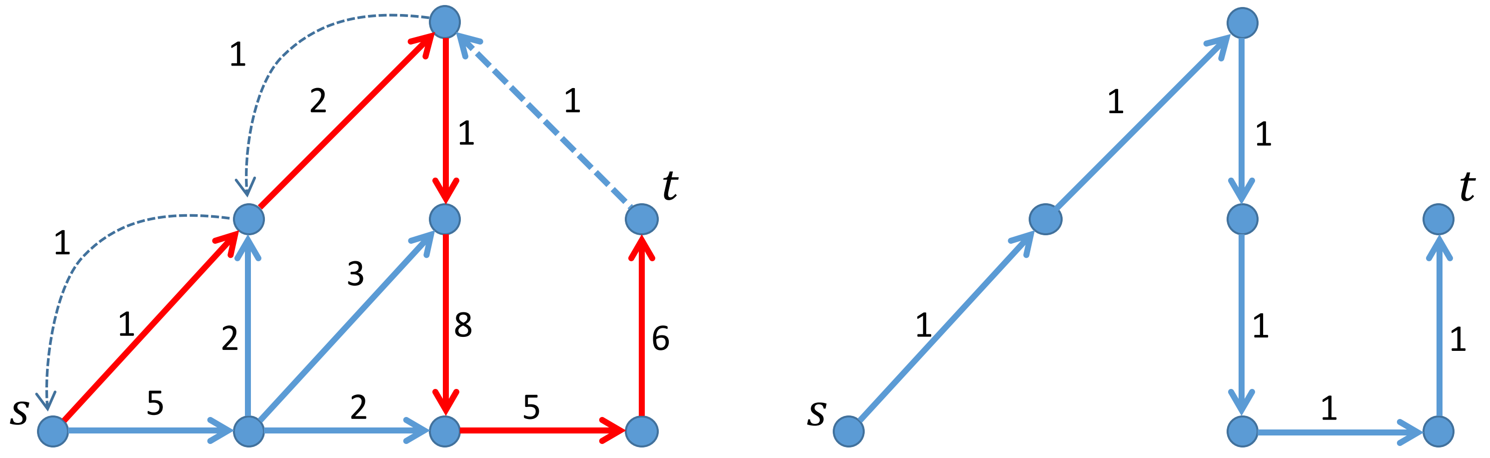
$$c(e) = c_0 - f_0$$

и добавляем вспомогательную дугу e' в обратном направлении, считая при этом

$$c(e') = f_0$$

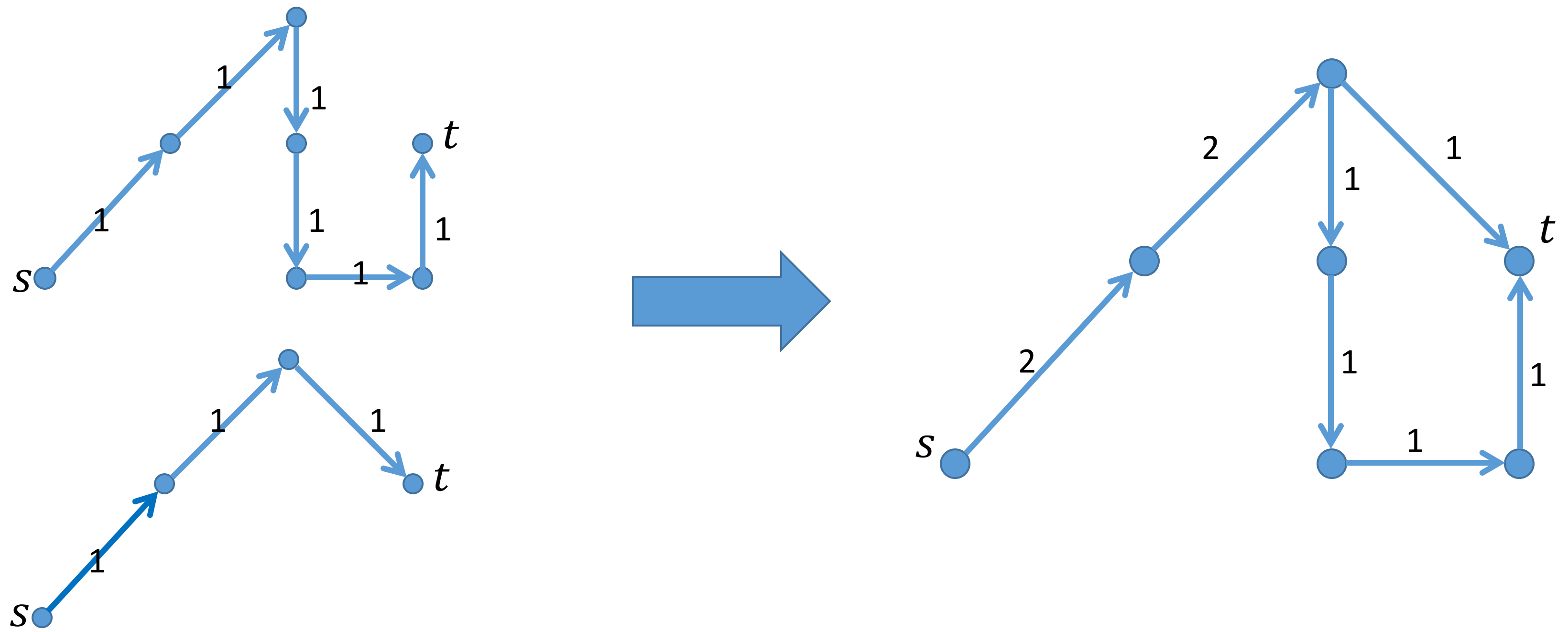
Алгоритм Форда—Фалкерсона

Теперь пытаемся найти какой-нибудь ненулевой поток в остаточной сети. Если он есть, значит поток в исходной сети можно увеличить:



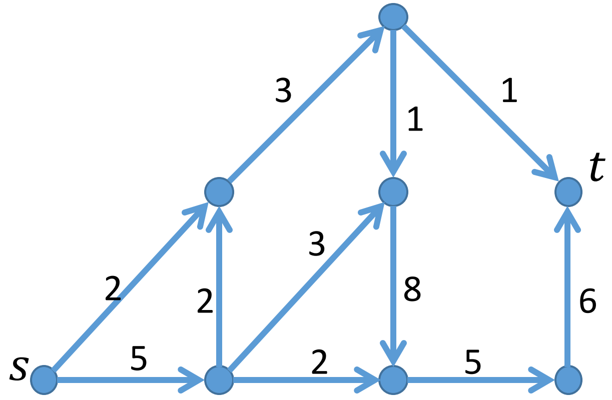
Алгоритм Форда—Фалкерсона

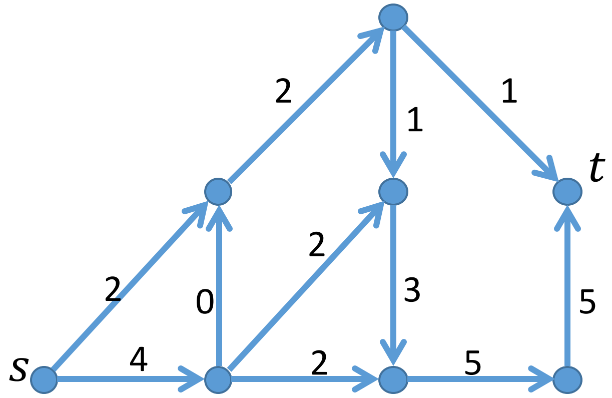
Теперь пытаемся найти какой-нибудь ненулевой поток в остаточной сети. Если он есть, значит поток в исходной сети можно увеличить:

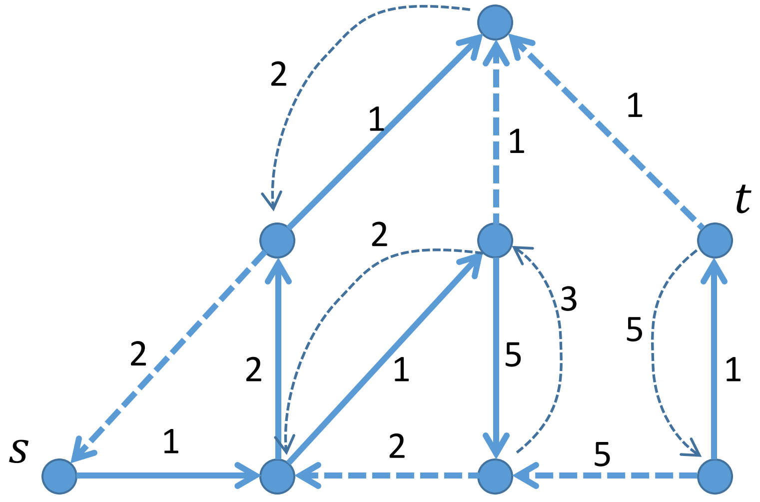


Алгоритм Форда—Фалкерсона

Если в остаточной сети не оказалось ненулевого потока, то это в ней нельзя дойти от s до t по дугам с положительными пропускными способностями.

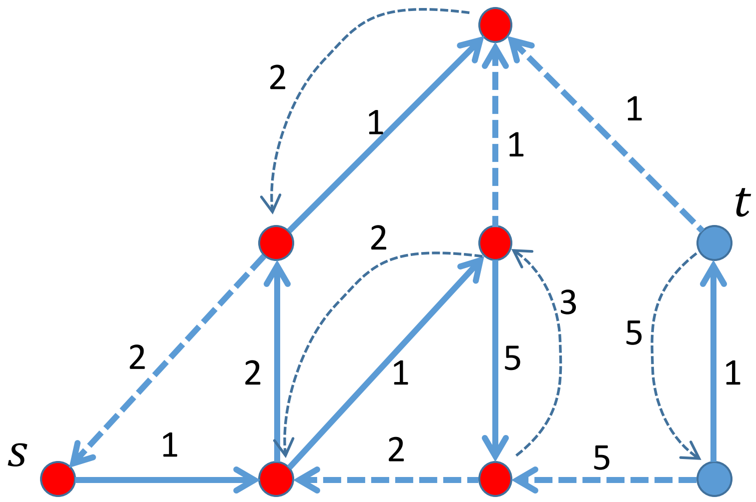






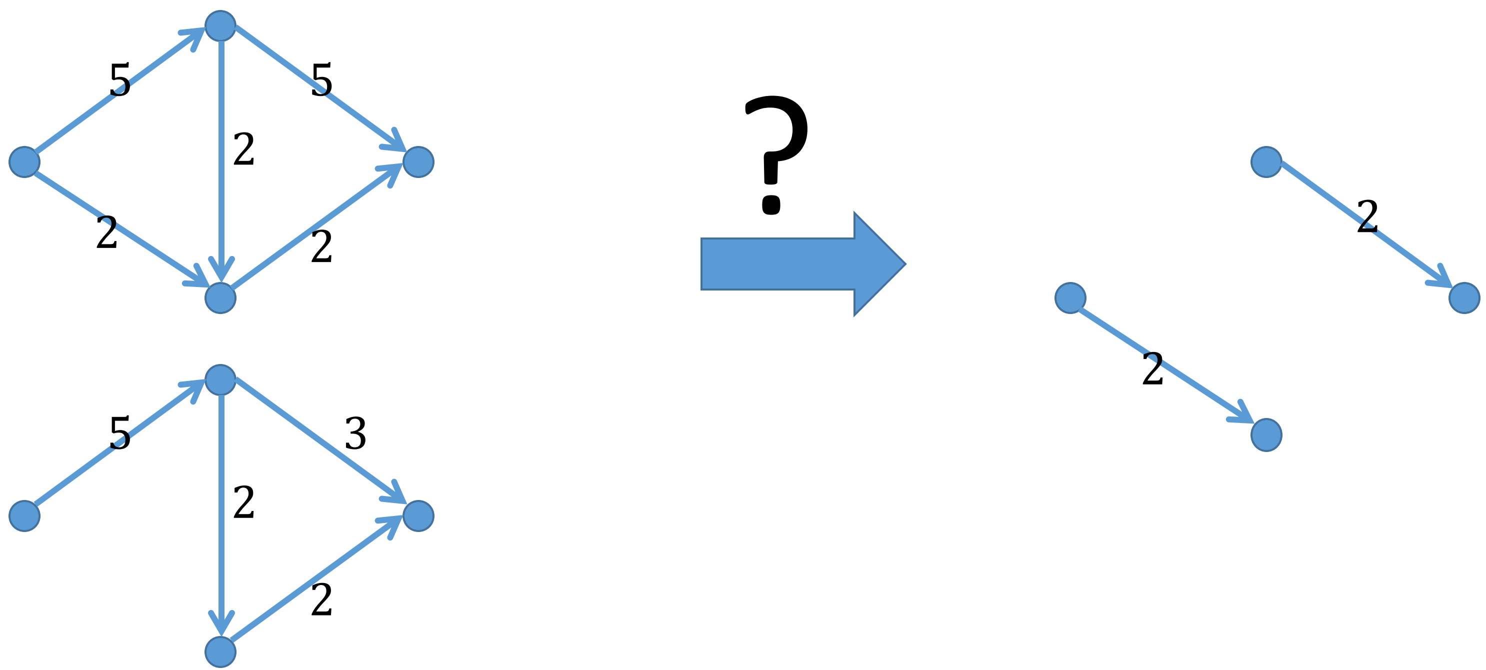
Алгоритм Форда—Фалкерсона

Если в остаточной сети не оказалось ненулевого потока, то это в ней нельзя дойти от s до t по дугам с положительными пропускными способностями. Беря в качестве S вершины, до которых можно дойти по ненулевым дугам из s, а в качестве T все остальные вершины, получаем разрез. Его пропускная способность равна величине потока, по которому строилась остаточная сеть.



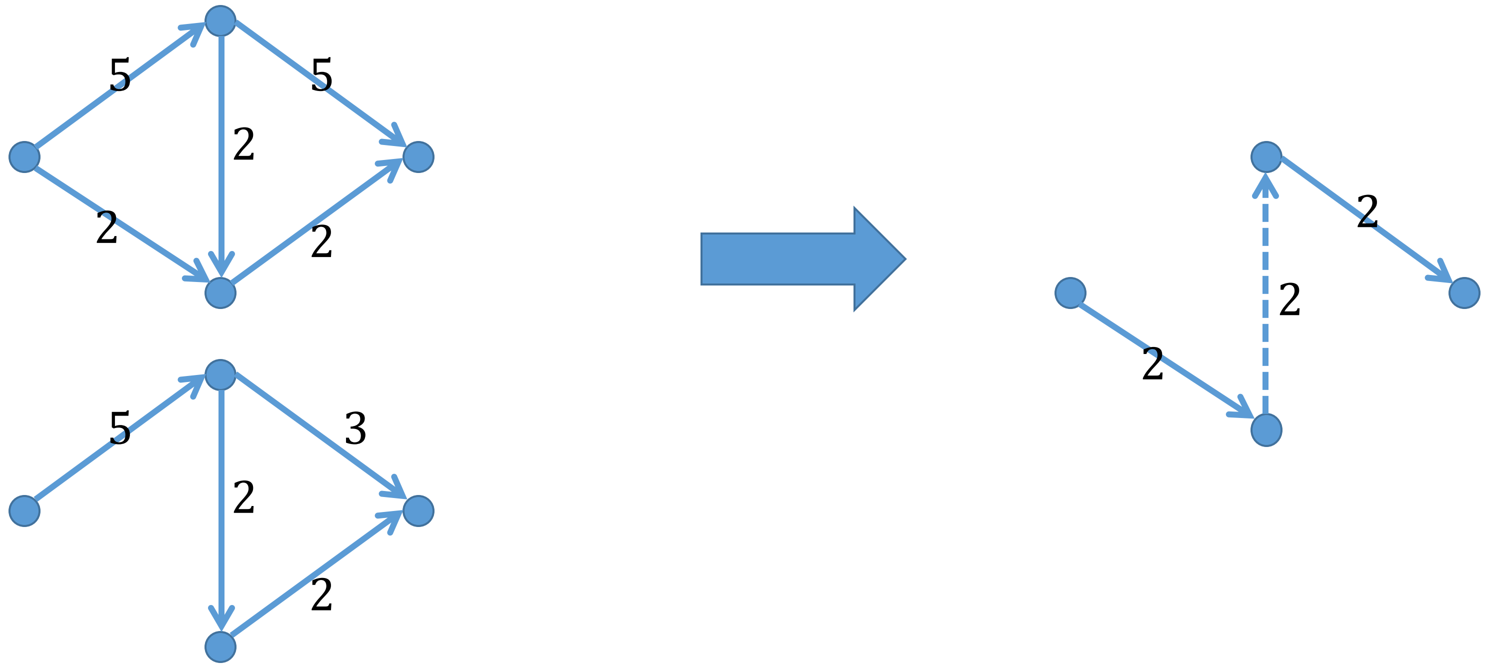
Алгоритм Форда—Фалкерсона

Замечание. Проводить «обратные» дуги при построении остаточной сети важно, иначе максимальность получаемого потока не гарантируется.



Алгоритм Форда—Фалкерсона

Замечание. Проводить «обратные» дуги при построении остаточной сети важно, иначе максимальность получаемого потока не гарантируется.



Теорема Форда—Фалкерсона

- Если все пропускные способности в сети целочисленные, то и построенный алгоритмом Форда—Фалкерсона поток оказывается целочисленным (то есть $f(e) \in \mathbb{Z}$ для каждой дуги e).
- Если все пропускные способности в сети — рациональные дроби, то, домножив их на наименьшее общее кратное знаменателей, можно перейти к целым числам.
- Если пропускные способности в сети иррациональные, то *не гарантируется, что алгоритм Форда—Фалкерсона завершится*.