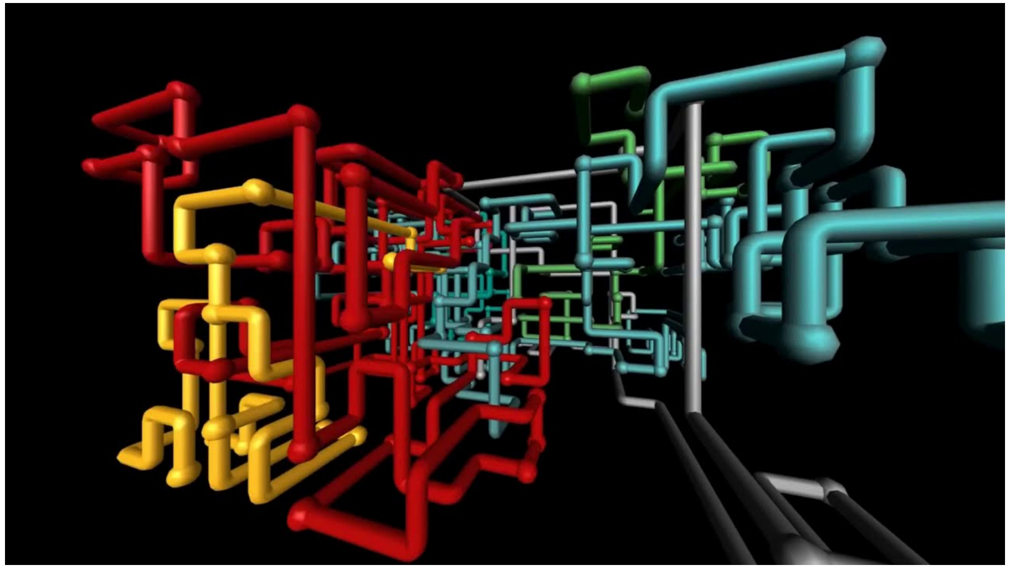
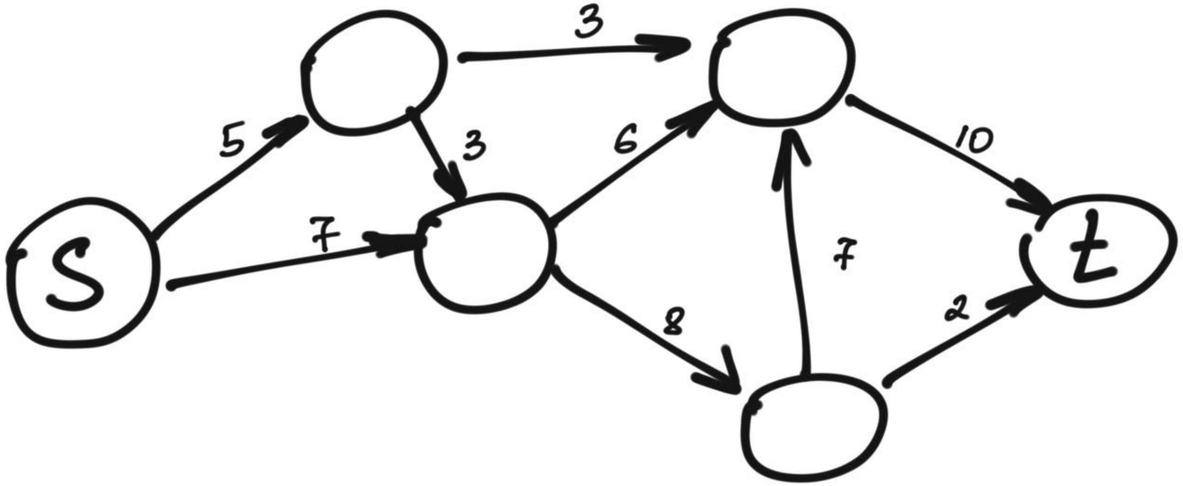
Потоки

Задача

Даны узлы и соединяющие их ребра (трубы/дороги/кабели). Каждому ребру ставится в соответствие некоторое число - пропускная способность. Требуется определить максимальную величину пропускной способности сети из заданного источника (source) в сток (sink).



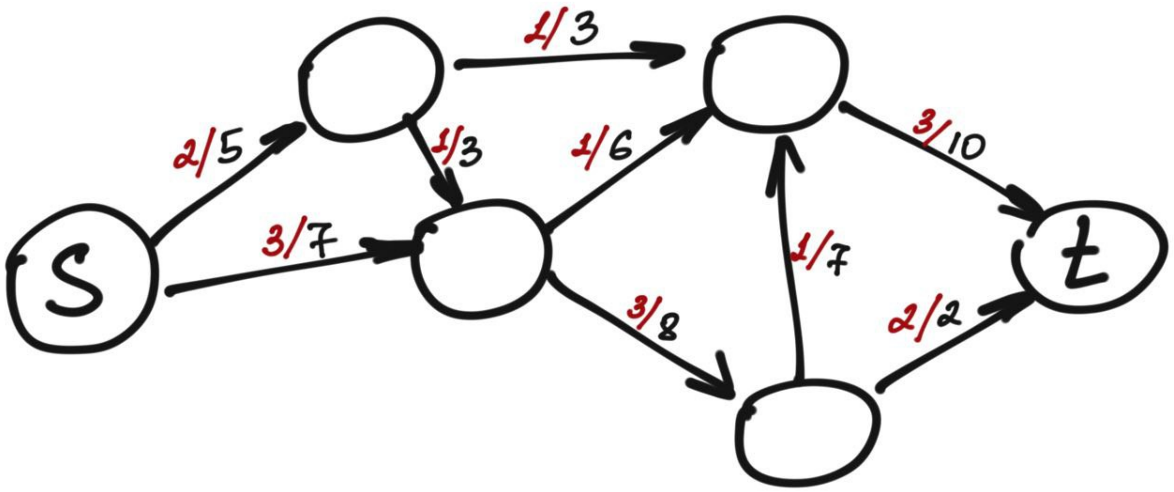


Определения (естественные, но неудобные)

Опр. *Транспортной сетью* называется граф $G = (V, E)$ с выделенной парой вершин s (исток) и t (сток) и введенной на нем функцией пропускных способностей $c : V \times V \to \mathbb{R}_+$, причем $(v, u) \notin E \Leftrightarrow c(v, u) = 0$.

Опр. *Потоком* в транспортной сети называется функция $f : V \times V \to \mathbb{R}_+$:

1. $\forall v, u \in V : 0 \leq f(v, u) \leq c(v, u)$
2. $\forall v \in V \setminus \{s, t\} : \sum\limits_{u \in V} f(v, u) = \sum\limits_{u \in V} f(u, v)$

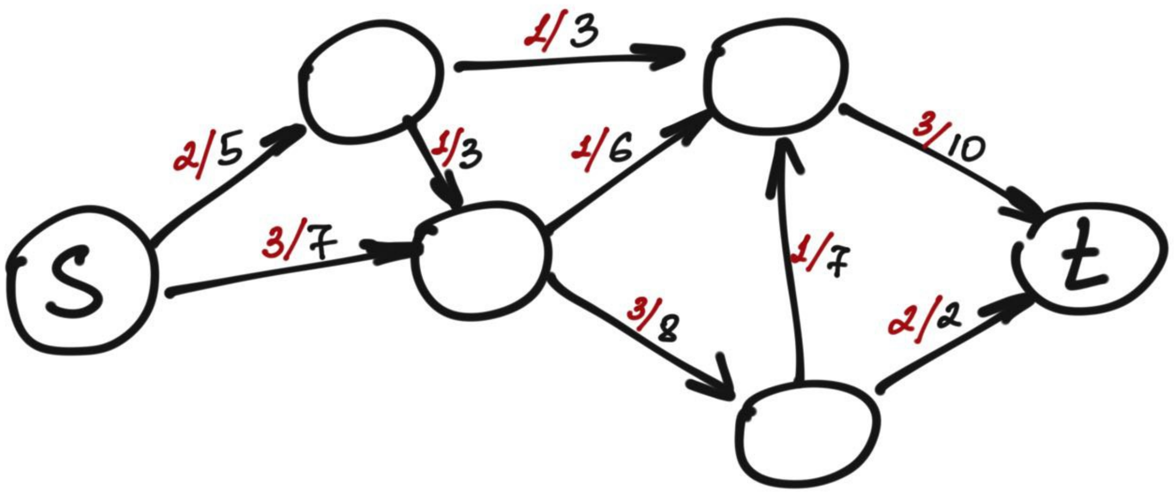


Определения (естественные, но неудобные)

Опр. *Величиной исходящего потока* называется число $|f|_+ = \sum_{u \in V} f(s,u)$

Опр. *Величиной входящего потока* называется число $|f|_- = \sum_{u \in V} f(u,t)$

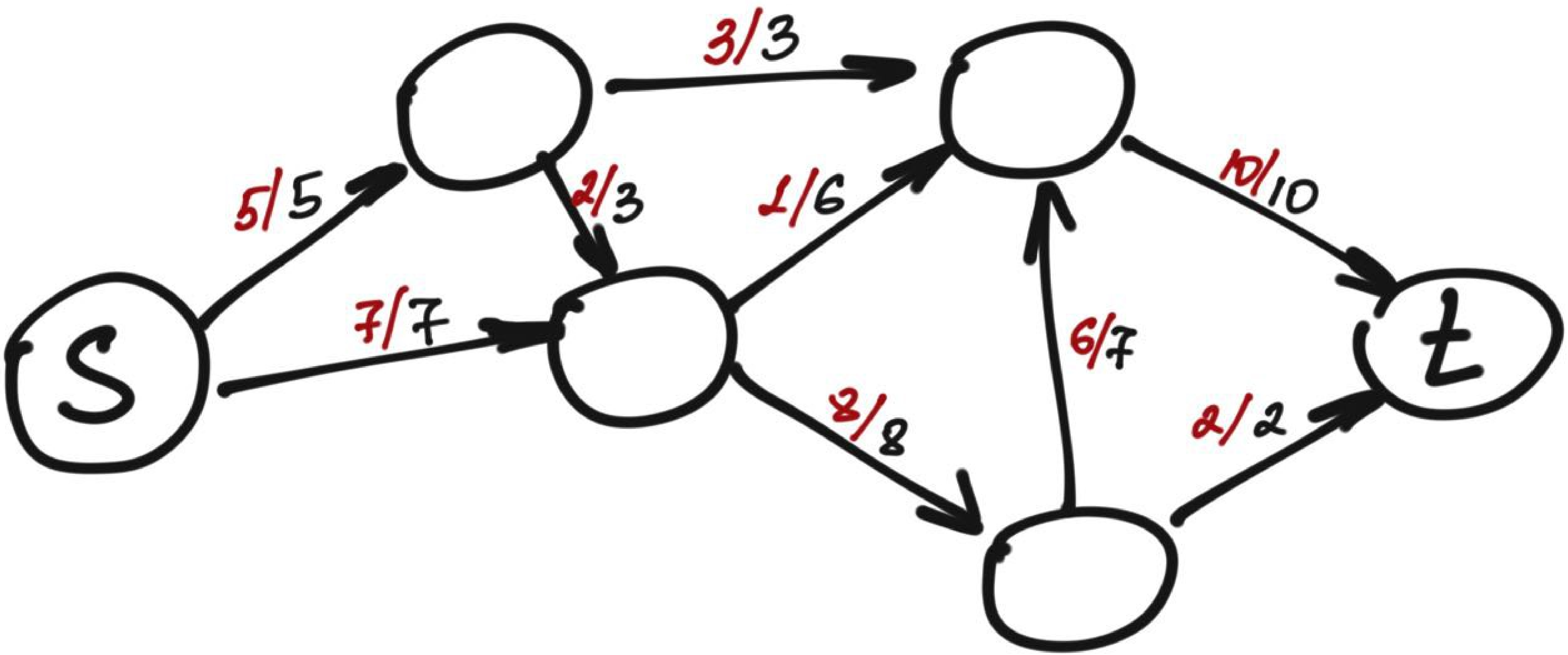
Очевидным утверждением (которое, тем не менее, докажем позже) является равенство исходящего и входящего потоков. Поэтому для упрощения будем просто говорить *величина потока* ($|f|$).



Задача поиска потока максимальной величины

Математически задача формуллируется так:

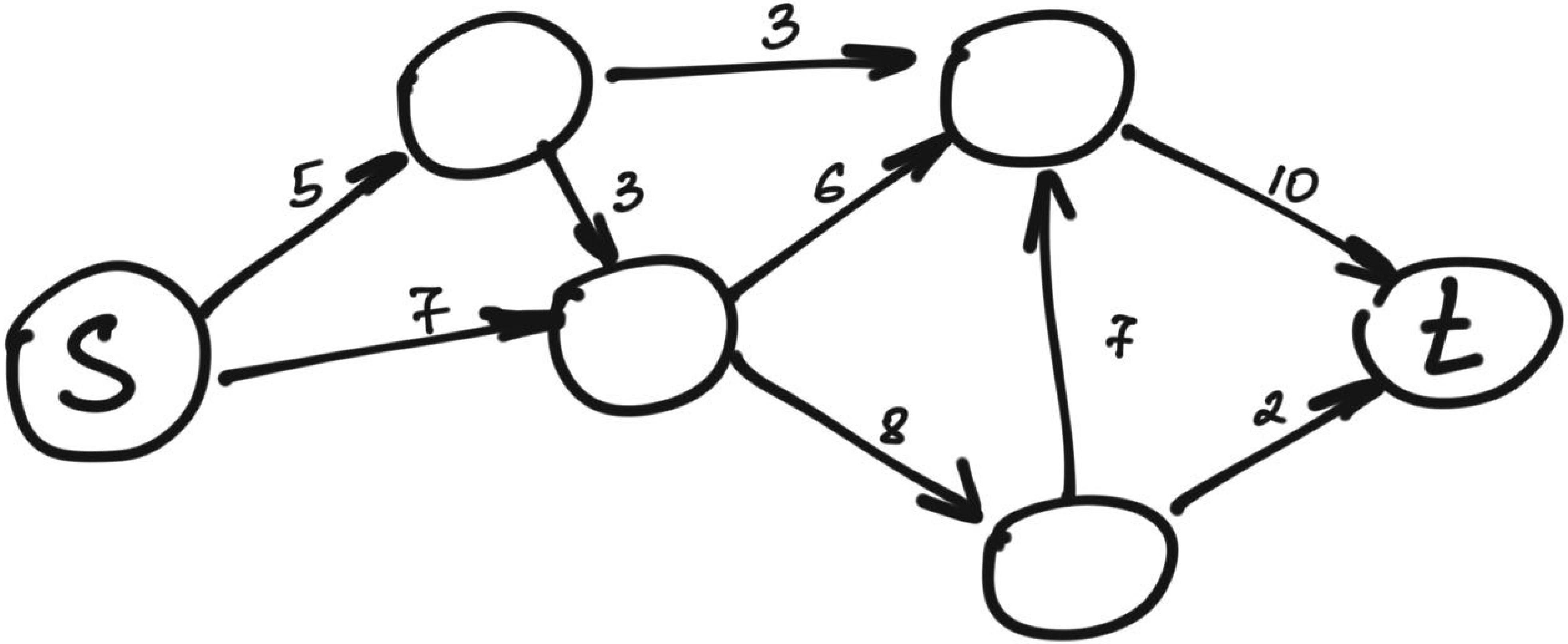
Для заданной транспортной сети найти $f^* = \arg\max_f |f|$



План

Будем действовать жадно:

1. Найдем произвольный путь из истока в сток и пустим по нему "ручеек".
2. Повторяем 1. пока не останется доступных путей.

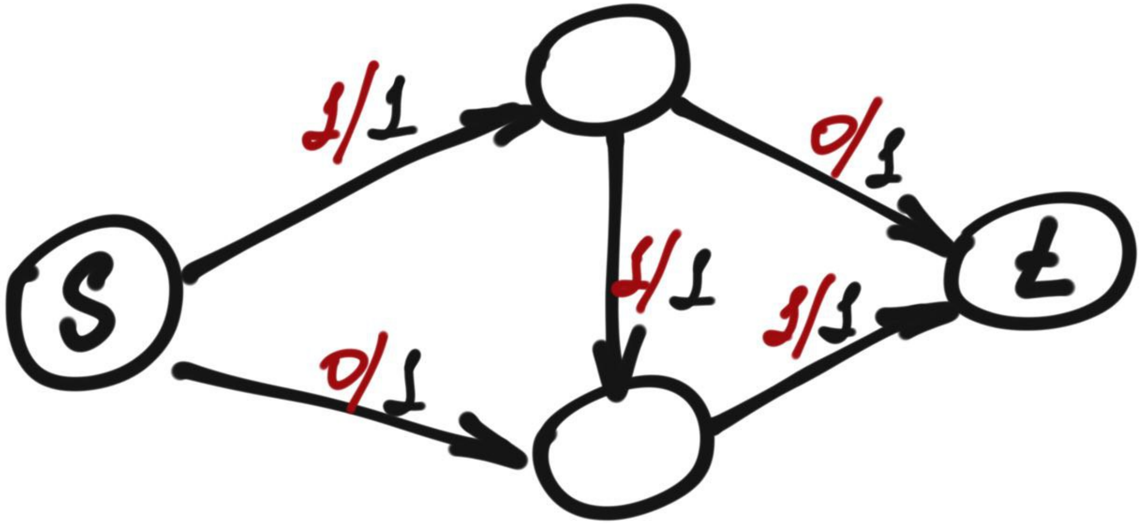


Изъян в плане

Будем действовать жадно:

1. Найдем произвольный путь из истока в сток и пустим по нему "ручеек".
2. Повторяем 1. пока не останется доступных путей.

К сожалению, возможна ситуация, когда пути исчерпаются раньше времени.



Проблема в том, что мы не умеем исправлять "плохие ручейки".

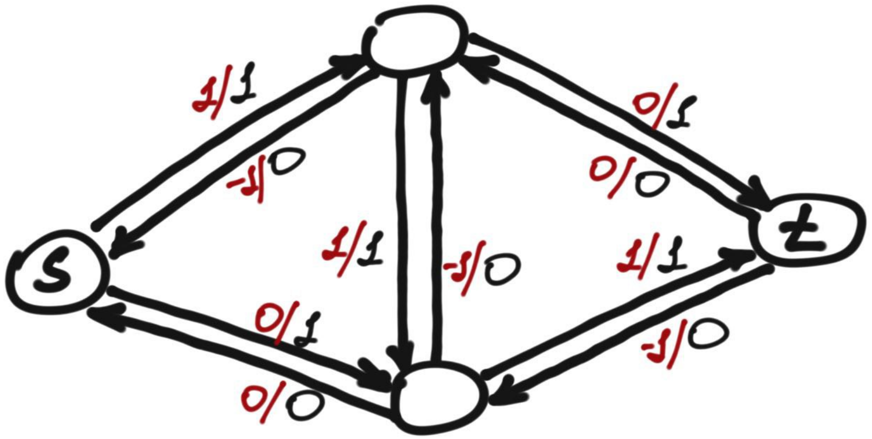
Симметризованный поток

Введем более удобное на практике определение потока:

Опр. *Потоком* в транспортной сети называется функция $f : V \times V \to \mathbb{R}_+$:

1. $\forall v, u \in V : f(v, u) = -f(u, v)$
2. $\forall v, u \in V : f(v, u) \leq c(v, u)$
3. $\forall v \in V \setminus \{s, t\} : \sum\limits_{u \in V} f(v, u) = 0$

Найдите 3 отличия от предыдущего определения.



Величина потока

Опр. *Величиной исходящего потока* называется число $|f|_+ = \sum\limits_{u \in V} f(s,u)$

Опр. *Величиной входящего потока* называется число $|f|_- = \sum\limits_{u \in V} f(u,t)$

Утверждение. $|f|_+ = |f|_-$

Доказательство.

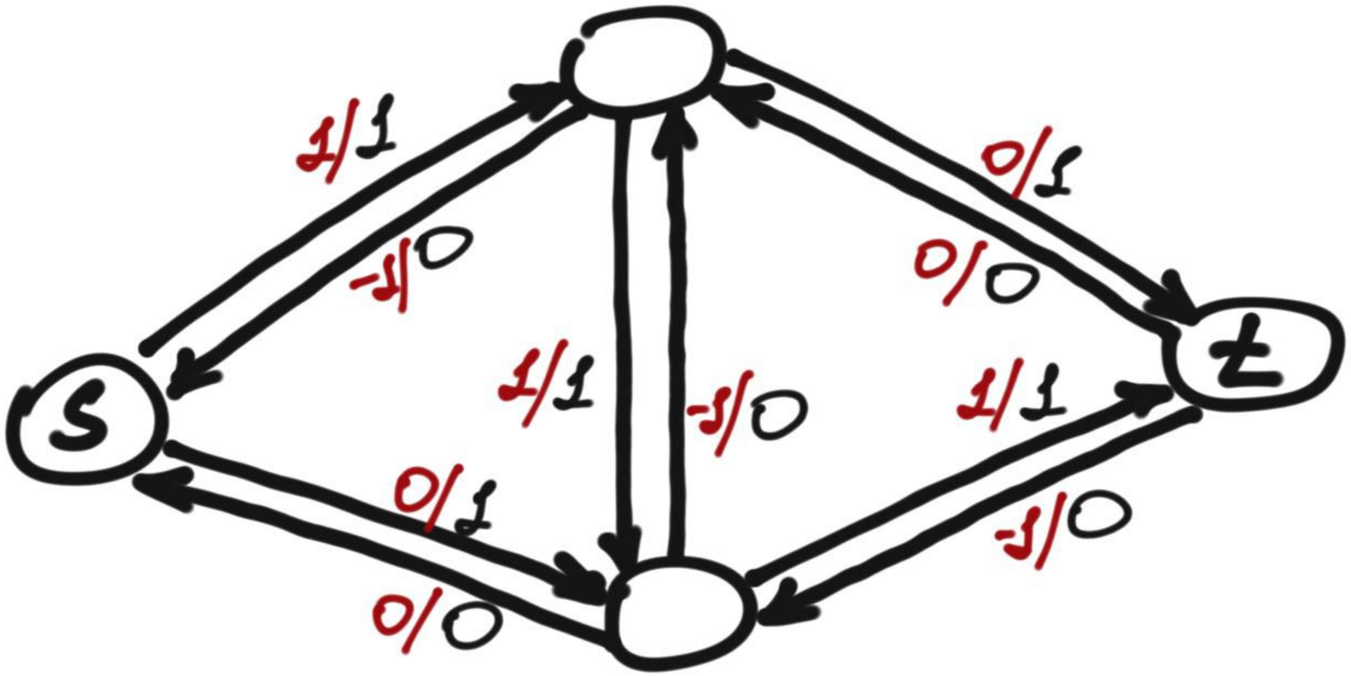
$$0 = \sum_{v \in V} \sum_{u \in V} f(v,u) = \sum_{u \in V} f(s,u) + \sum_{u \in V} f(t,u) + \sum_{\substack{v \neq s \\ v \neq t}} \sum_{u \in V} f(v,u) =$$

$$\sum_{u \in V} f(s,u) + \sum_{u \in V} f(t,u) = \sum_{u \in V} f(s,u) - \sum_{u \in V} f(u,t) \blacksquare$$

Почему симметризация решает проблему

Вернемся к плану:

1. Найдем произвольный путь из истока в сток и пустим по нему "ручеек".
2. Повторяем 1. пока не останется доступных путей.

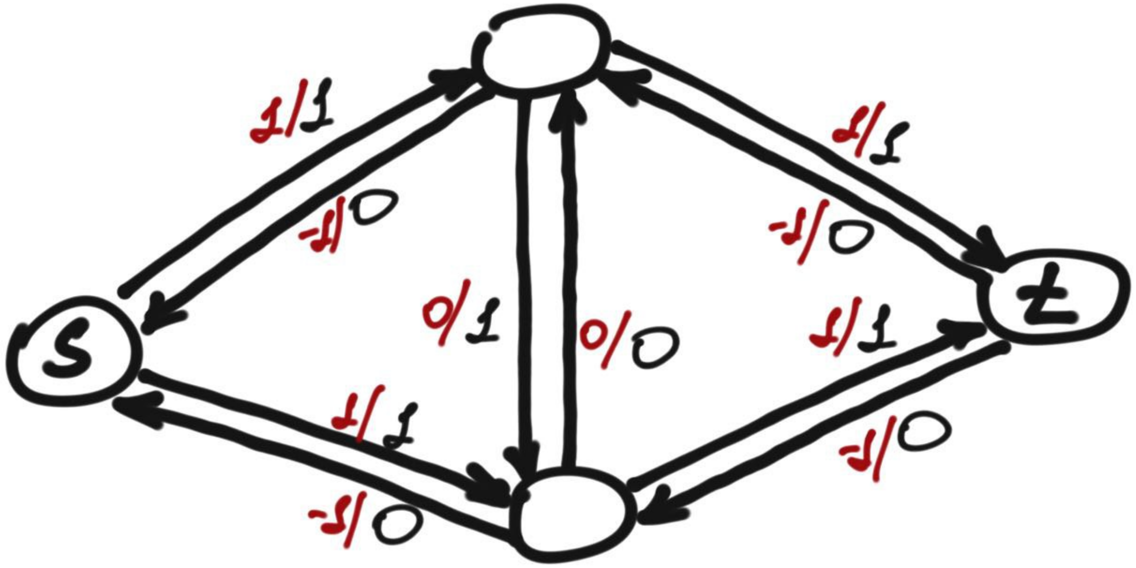


Почему симметризация решает проблему

Вернемся к плану:

1. Найдем произвольный путь из истока в сток и пустим по нему "ручеек".
2. Повторяем 1. пока не останется доступных путей.

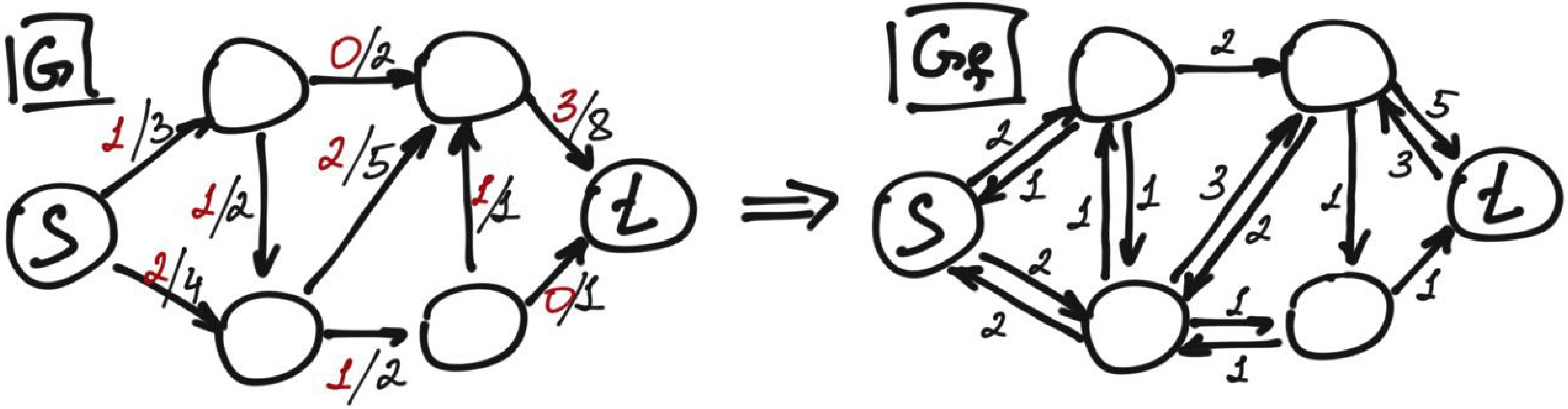
Теперь можем пускать потоки и в обратном направлении, то есть исправлять ошибки прошлого.



Почему это работает

Арифметика потоков

Опр. *Остаточной сетью* для сети $G = (V, E, c)$ и потока f называется сеть $G_f = (V, E_f, c_f)$, где $c_f = c - f$.



Арифметика потоков

Лемма 1 (сложение потоков)

Пусть f - поток в G, а h - поток в G_f. Тогда $f + h$ - поток в G, причем $|f + h| = |f| + |h|$.

Доказательство.

Проверим, что $f + h$ является потоком.

1. $\forall v, u : f(v, u) + h(v, u) = -f(u, v) - h(u, v) = -(f(u, v) + h(u, v))$
2. Так как h - поток в G_f: $\forall v, u : h(v, u) \leq c(v, u) - f(v, u) \Rightarrow f + h \leq c$
3. $\forall v \neq s, t : \sum\limits_{u \in V} (f + h)(v, u) = \sum\limits_{u \in V} f(v, u) + \sum\limits_{u \in V} h(v, u) = 0 + 0 = 0$ ■

Лемма 2 (разность потоков)

Пусть f_1 и f_2 - потоки в G. Тогда $f_2 - f_1$ - поток в G_{f_1}, причем $|f_2 - f_1| = |f_2| - |f_1|$.

Алгоритм Форда-Фалкерсона

Так как потоки суммируются (лемма 1), можно построить следующий алгоритм:

0. Инициализируем $f \equiv 0$
1. Ищем в G_f путь из s в t (по ребрам с положительной остаточной пропускной способностью) с помощью DFS, пускаем по нему максимально возможный поток Δf.
2. Обновляем $f = f + \Delta f$
3. Повторяем 1-3 пока в G_f есть путь из s в t.

Кстати, путь найденный на шаге 2 называется *дополняющим путем*.

Время работы для целочисленных c: $O(E \cdot |f_{max}|)$. Неполиномиальненько

Теорема Форда-Фалкерсона

Теорема (L.Ford, D.Fulkerson, 1962).
Поток f в сети G максимален $\iff$ В сети G_f нет дополняющего пути.

Доказательство.
$\Rightarrow$ Допустим нашли дополняющий путь $h : |h| > 0$ в сети G_f. По лемме о сумме потоков $(f + h)$ - поток в G и $|f + h| = |f| + |h| > |f|$!!! (f максимален).
$\Leftarrow$ Пусть f_{max} некоторый максимальный поток для сети G. По лемме о разности потоков $(f_{max} - f)$ - поток в G_f и $|f_{max} - f| = |f_{max}| - |f| \geq 0$.
Но в G_f нет дополняющего пути, поэтому $|f_{max} - f| \leq 0$.
Таким образом $|f_{max}| = |f|$. ■

Алгоритм Эдмондса-Карпа

Алгоритм Эдмондса-Карпа

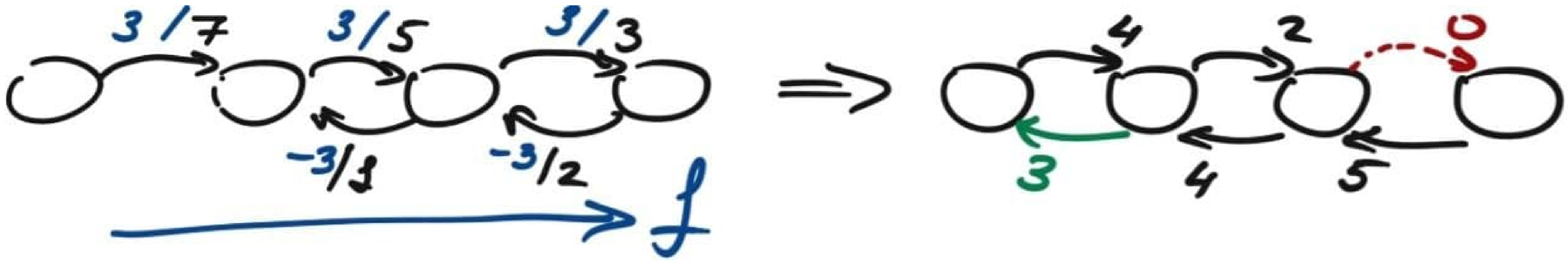
0. Инициализируем $f \equiv 0$
1. Ищем в G_f путь из s в t (по ребрам с положительной остаточной пропускной способностью) с помощью BFS, пускаем по нему максимально возможный поток Δf.
2. Обновляем $f = f + \Delta f$
3. Повторяем 1-3 пока в G_f есть путь из s в t.

Работает за $O(VE^2)$. Придется доказывать.

Поиск потока максимальной величины: Алгоритм Эдмондса-Карпа

Лемма.
После каждой итерации алгоритма (шагов 1-2), длины кратчайших путей (в смысле **количества** ребер) из s не убывают.

Доказательство.
Что происходит с остаточной сетью после пропускания по некоторому пути дополняющего потока? 1) Некоторые ребра исчезают; 2) Могут появиться ребра в обратном направлении.



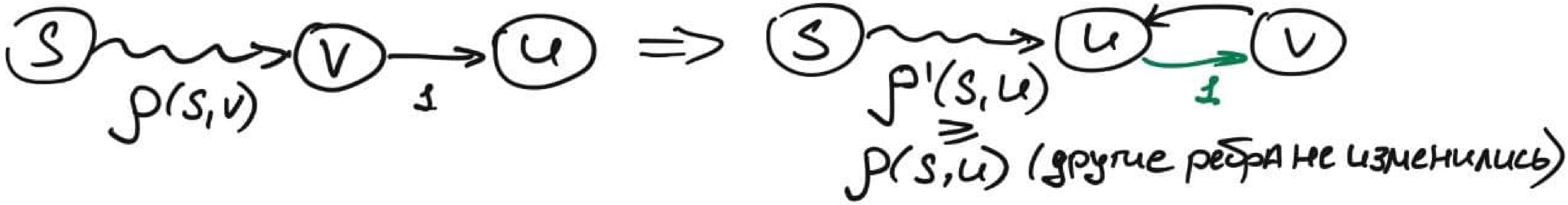
Лемма.

После каждой итерации алгоритма (шагов 1-2), длины кратчайших путей из s не убывают.

Доказательство (продолжение).

Покажем, что ни удаление ребер, ни добавление обратных не может привести к уменьшению кратчайших путей.

1. С удалением очевидно.
2. Пусть кратчайший путь из s в u проходит по ребру vu. Добавим ему обратное. Покажем, что путь из s в v не стал короче. $\rho'(s,v) \stackrel{?}{=} \rho'(s,u) + 1 \geq \rho(s,u) + 1 = \rho(s,v) + 1 + 1$!!!. ■



Поиск потока максимальной величины: Алгоритм Эдмондса-Карпа

Назовем ребро *критическим*, если поток его насытил (исчезло из сети).

Теорема (J.Edmonds, R.Karp, 1972).

В процессе работы алгоритма каждое ребро может стать критическим не более $|V|/2$ раз.

Доказательство.

Пусть ребро vu стало критическим в момент, когда расстояние от s до u было $\rho(s,u) \overset{*}{=} \rho(s,v)+1$. Оно вернется, когда в обратном направлении uv пустим поток, то есть uv будет лежать на кратчайшем пути из s в v: $\rho'(s,v) = \rho'(s,u)+1 \overset{\text{лемма}}{\geq} \rho(s,u)+1$.

Теорема (J.Edmonds, R.Karp, 1972).

В процессе работы алгоритма каждое ребро может стать критическим не более $|V|/2$ раза.

Доказательство (продолжение).

Имеем:

1. $\rho(s,u) = \rho(s,v) + 1$
2. $\rho'(s,v) \geq \rho(s,u) + 1$

Значит, когда ребро vu снова станет критическим расстояние до u будет $\rho''(s,u) = \rho''(s,v) + 1 \overset{\text{лемма}}{\geq} \rho'(s,v) + 1 \overset{2.}{\geq} \rho(s,u) + 2$, то есть между двумя последовательными "критическими" моментами для ребра vu расстояния до u увеличилось не менее чем на 2. Так как расстояние не может превышать $|V| - 1$, ребро vu может стать критическим не более $|V|/2$ раз. ■

Алгоритм Эдмондса-Карпа: анализ

0. Инициализируем $f \equiv 0$
1. Ищем в G_f путь из s в t (по ребрам с положительной остаточной пропускной способностью) с помощью BFS, пускаем по нему максимально возможный поток Δf.
2. Обновляем $f = f + \Delta f$
3. Повторяем 1-3 пока в G_f есть путь из s в t.

Анализ времени работы. На каждой итерации (пункт 3) какое-то ребро становится критическим. Всего ребер E, каждое будет критическим не более $V/2$ раз. То есть $E\frac{V}{2}$ итераций. На каждой итерации запускается BFS ($O(E)$). Итого: $O(VE^2)$.

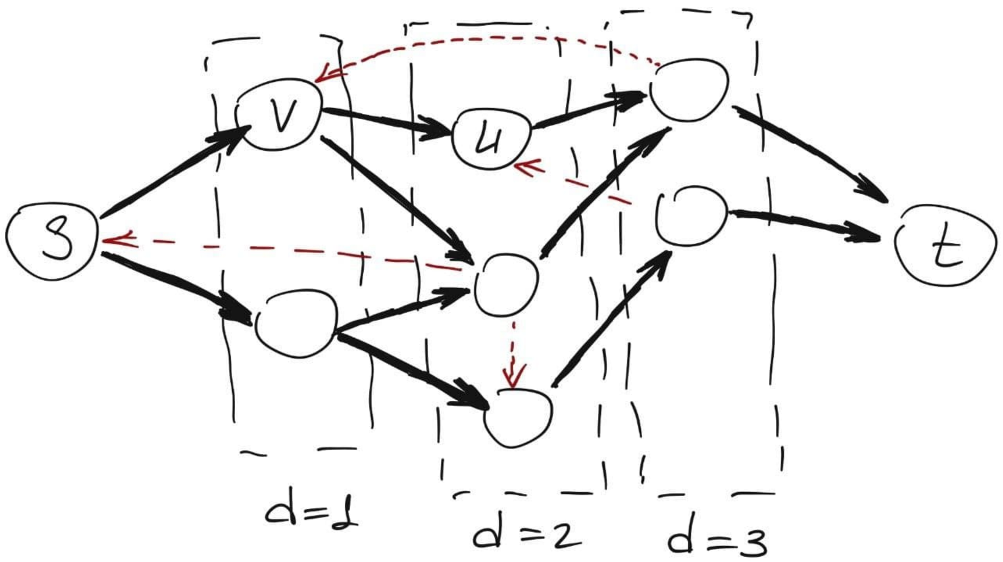
Алгоритм Диница

Алгоритм Диница

В чем проблема алгоритма Эдмондса-Карпа?

Слишком много раз вызываем BFS. Но ведь результат одного вызова BFS можно переиспользовать несколько раз!

Построим *слоистую сеть* (граф, где ребра vu удовлетворяют $d[u] = d[v] + 1$)



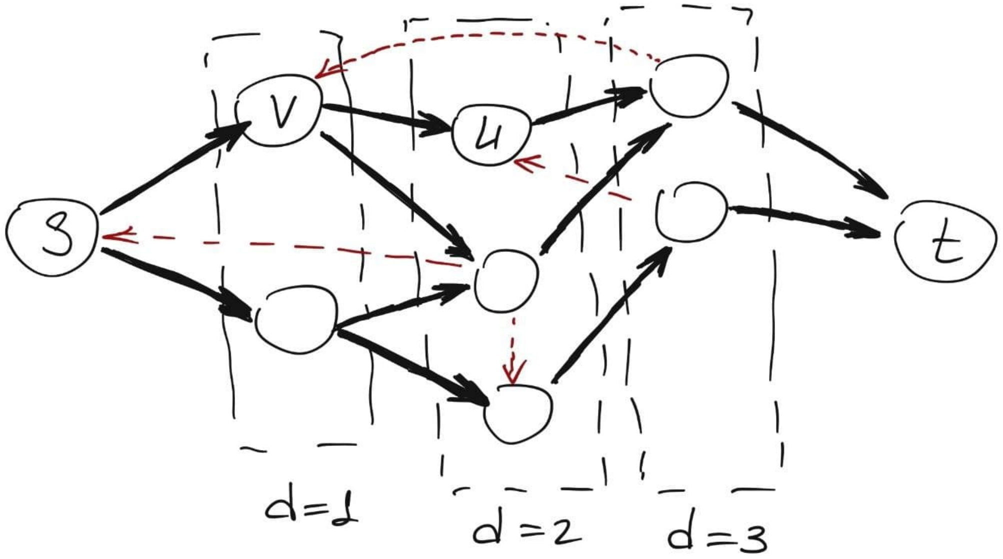
Алгоритм Диница

0. Инициализируем $f \equiv 0$

1. Строим слоистую сеть по G_f

2. Пока возможно, ищем дополняющие пути из s в t в сл-й сети, обновляем f.

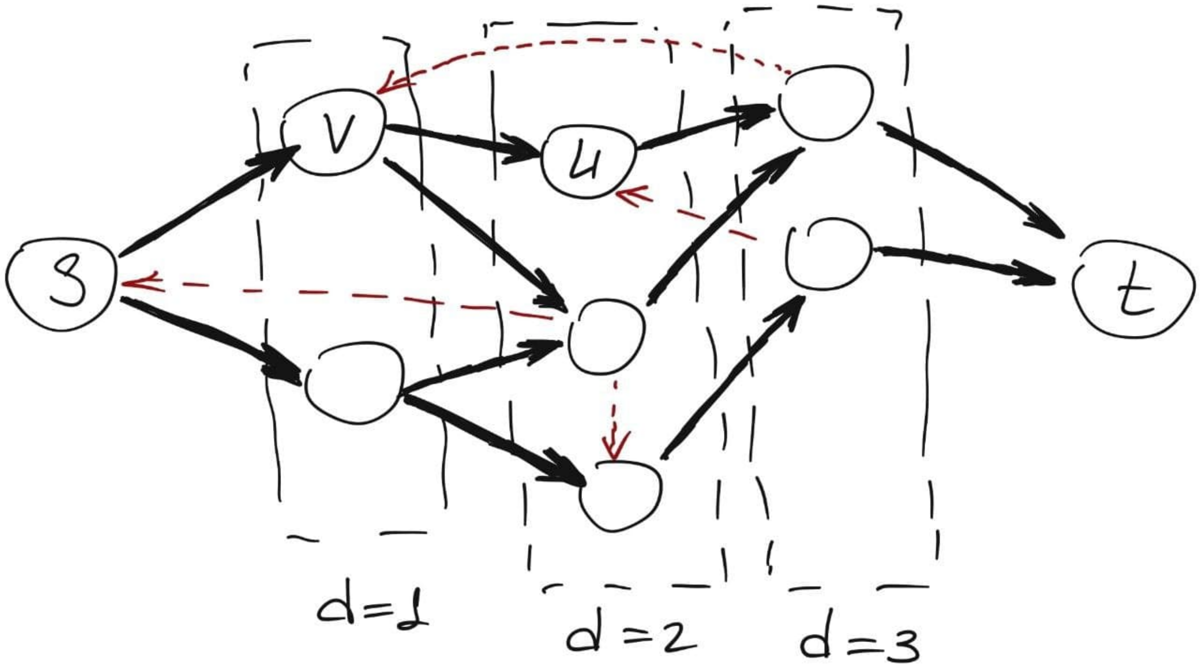
3. Повторяем 1-2 пока в G_f есть путь из s в t.



Алгоритм Диница

Для оценки времени работы, ответьте на следующие вопросы.

- Сколько раз нужно перестраивать слоистую сеть?
- Сколько времени занимает поиск пути в слоистой сети?
- Сколько всего путей можно найти в слоистой сети?



Алгоритм Диница

- Сколько раз нужно перестраивать слоистую сеть?
  $\leq V$ раз. Так как после каждой итерации расстояние от s до t увеличивается.
- Сколько времени занимает поиск пути в слоистой сети?
  $\leq V$. Просто переходим от слоя к слою.
- Сколько всего путей можно найти в слоистой сети?
  $\leq E$. Каждый путь находит хотя бы одно критическое ребро.

Итого: $O(V^2E)$ (+ $O(E)$ на каждой итерации на удаление критических ребер, но это мелочь).

Поиск потока максимальной величины: что дальше?

- Алгоритм Малхотра-Кумар-Махешвари, 1978. $O(V^3)$
- Алгоритм Орлин, 2013 (основан не на теореме Форда-Фалкерсона). $O(VE)$