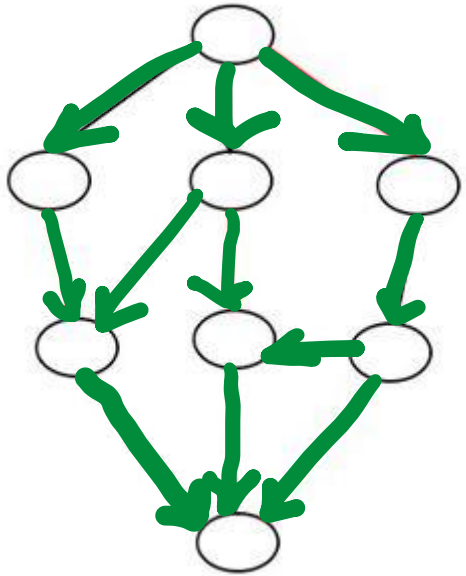
Максимальный поток, минимальный разрез, паросочетания

ИТМО ИС

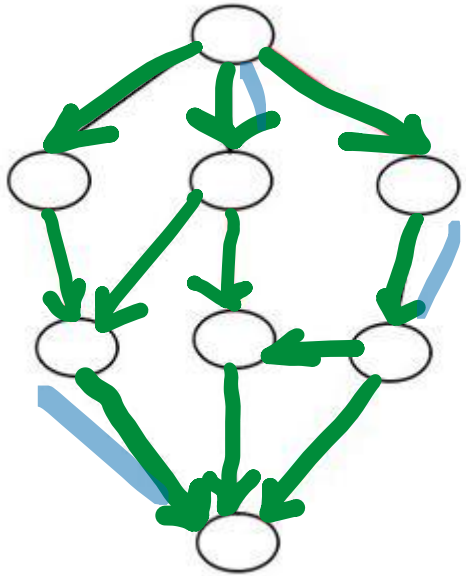
Задача о максимальном паросочетании

Паросочетание (англ. matching) M – произвольное множество рёбер графа такое, что никакие два ребра не имеют общей вершины.



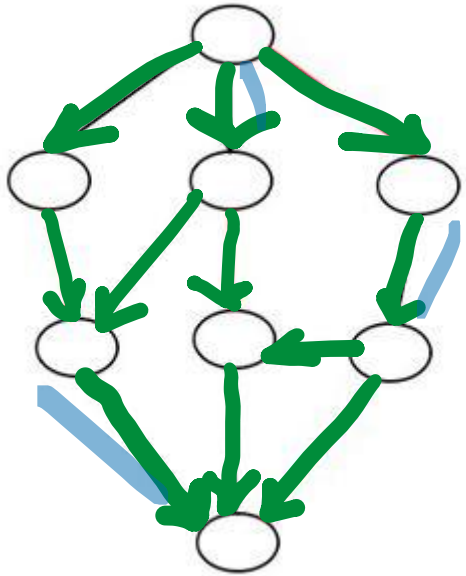
Задача о максимальном паросочетании

Паросочетание (англ. matching) M – произвольное множество рёбер графа такое, что никакие два ребра не имеют общей вершины.



Задача о максимальном паросочетании

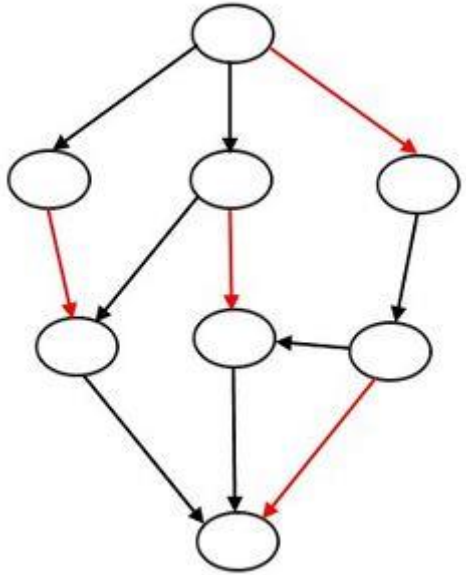
Паросочетание (англ. matching) M – произвольное множество рёбер графа такое, что никакие два ребра не имеют общей вершины.



можно ли лучше?

Задача о максимальном паросочетании

Паросочетание (англ. matching) M – произвольное множество рёбер графа такое, что никакие два ребра не имеют общей вершины.

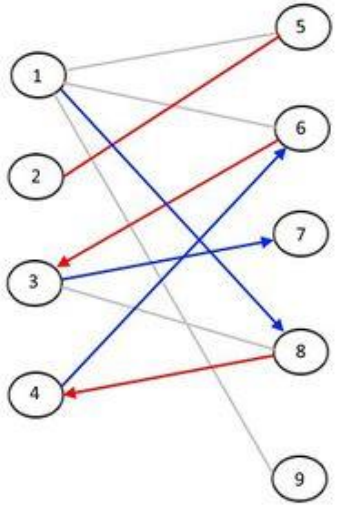


Задача о максимальном паросочетании

Определение:

Чередующаяся цепь (англ. *alternating path*) — путь в двудольном графе, для любых двух соседних рёбер которого верно, что одно из них принадлежит паросочетанию M, а другое нет.

Чередующаяся цепь:
1−8−4−6−3−7

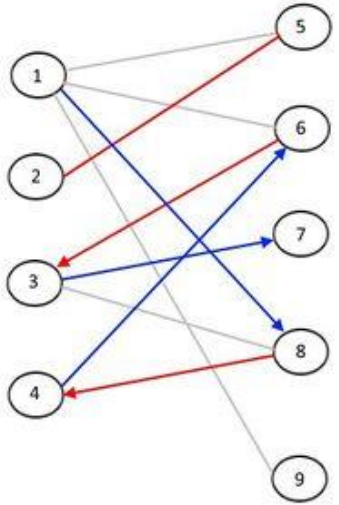


Задача о максимальном паросочетании

Определение:

Чередующаяся цепь (англ. *alternating path*) — путь в двудольном графе, для любых двух соседних рёбер которого верно, что одно из них принадлежит паросочетанию M, а другое нет.

Чередующаяся цепь:
1−8−4−6−3−7



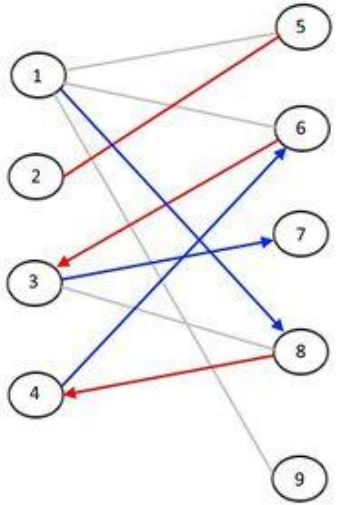
Задача о максимальном паросочетании

Определение:

Чередующаяся цепь (англ. *alternating path*) — путь в двудольном графе, для любых двух соседних рёбер которого верно, что одно из них принадлежит паросочетанию M, а другое нет.

Как еще может быть?

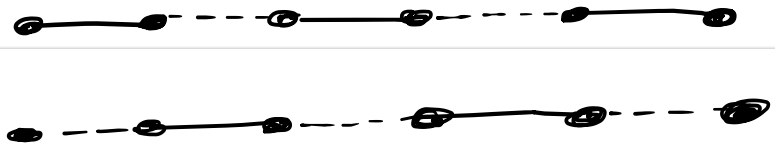
Чередующаяся цепь:
1−8−4−6−3−7



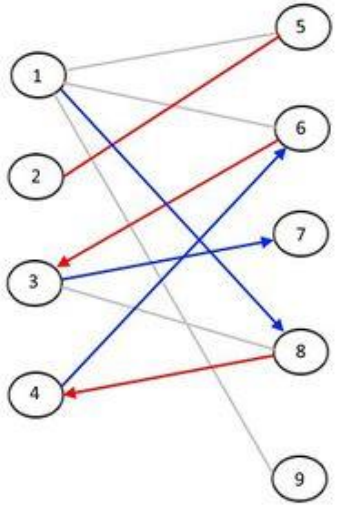
Задача о максимальном паросочетании

Определение:

Чередующаяся цепь (англ. *alternating path*) — путь в двудольном графе, для любых двух соседних рёбер которого верно, что одно из них принадлежит паросочетанию M, а другое нет.



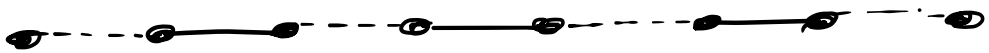
Чередующаяся цепь:
1−8−4−6−3−7



Задача о максимальном паросочетании

Определение:

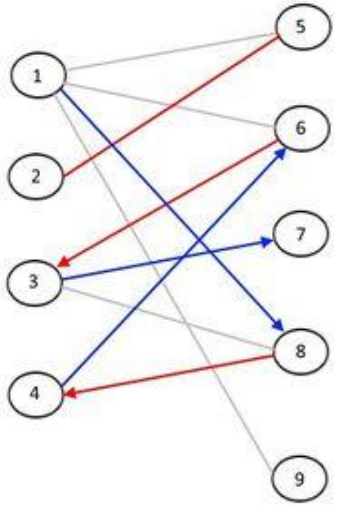
Чередующаяся цепь (англ. *alternating path*) — путь в двудольном графе, для любых двух соседних рёбер которого верно, что одно из них принадлежит паросочетанию M, а другое нет.



Определение:

Дополняющая цепь (или увеличивающая цепь) (англ. *augmenting path*) — чередующаяся цепь, у которой оба конца свободны.

Чередующаяся цепь:
1−8−4−6−3−7



Задача о максимальном паросочетании

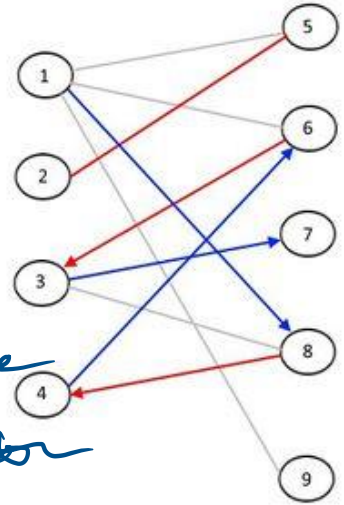
Определение:

Чередующаяся цепь (англ. *alternating path*) — путь в двудольном графе, для любых двух соседних рёбер которого верно, что одно из них принадлежит паросочетанию M, а другое нет.

Определение:

Дополняющая цепь (или увеличивающая цепь) (англ. *augmenting path*) — чередующаяся цепь, у которой оба конца свободны.

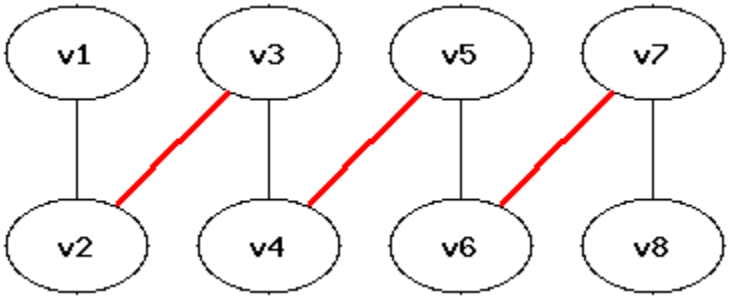
Чередующаяся цепь:
1−8−4−6−3−7



Задача о максимальном паросочетании

Теорема:

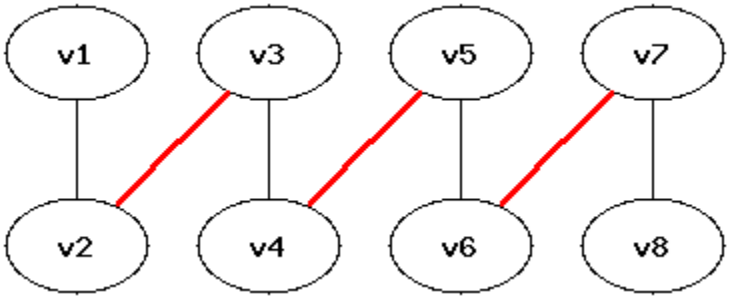
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



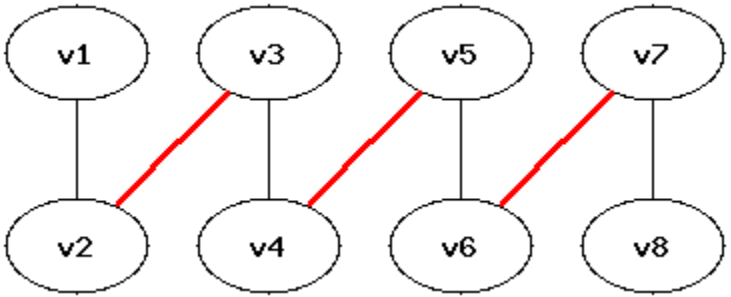
Идея доказательства?

Задача о максимальном паросочетании

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.

красные — были
черные — нет



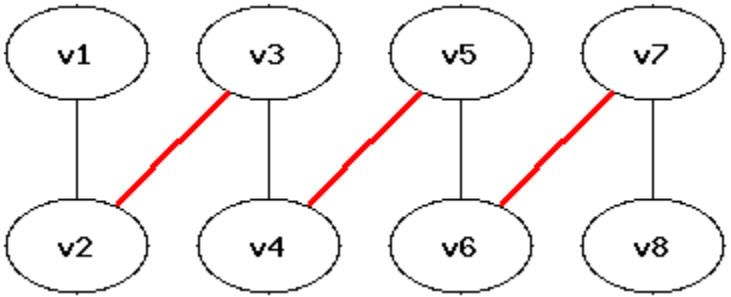
Задача о максимальном паросочетании

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.

есть M max

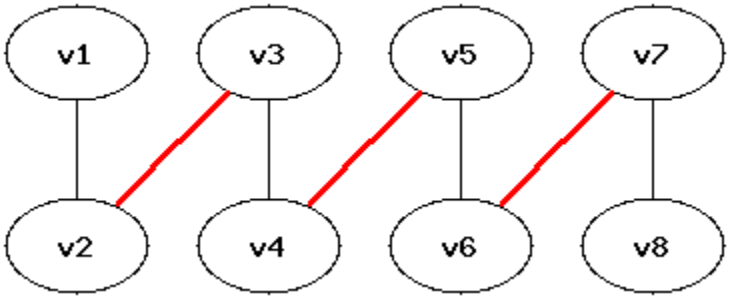
M' — мы нашли



Задача о максимальном паросочетании

Теорема:

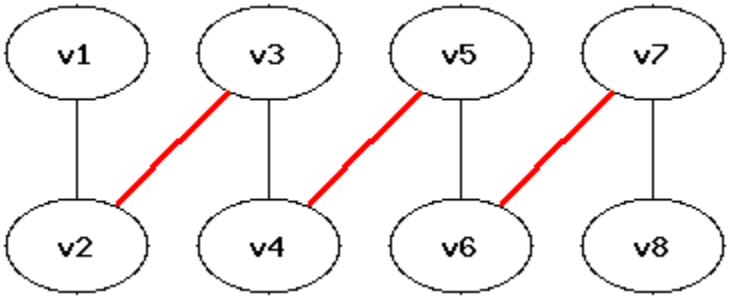
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



есть M max

M' — мы нашли

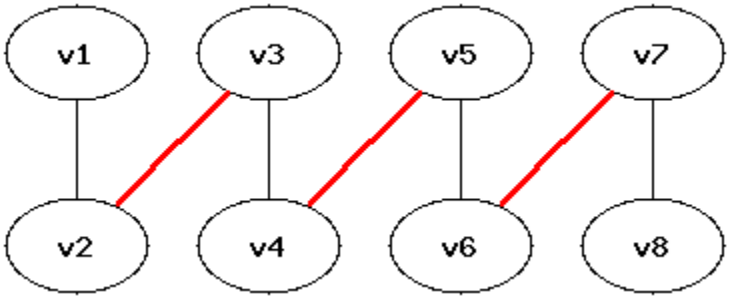
уберем остальные

сколько выходит ребер из вершины?

Задача о максимальном паросочетании

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



есть M max

M' — мы нашли

уберем остальные

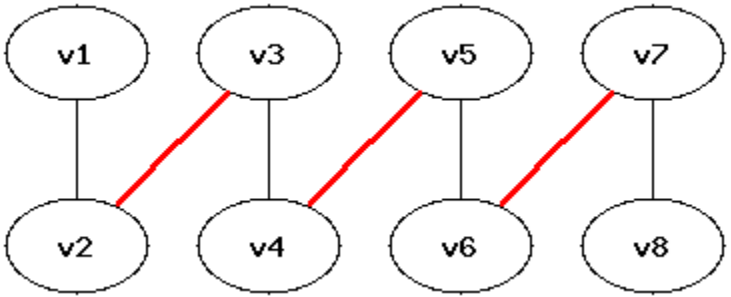
≤ 2 !

сколько выходит ребер из вершины?

Задача о максимальном паросочетании

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



что это тогда?

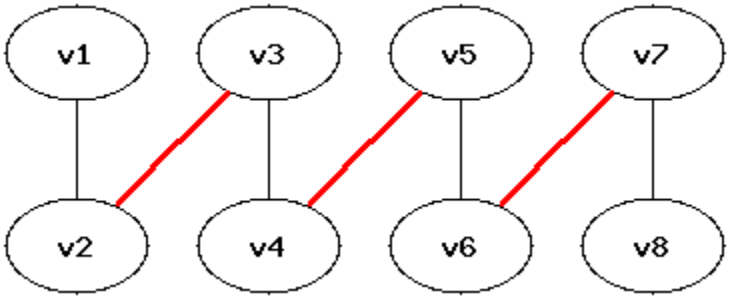
≤2!

?

Задача о максимальном паросочетании

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



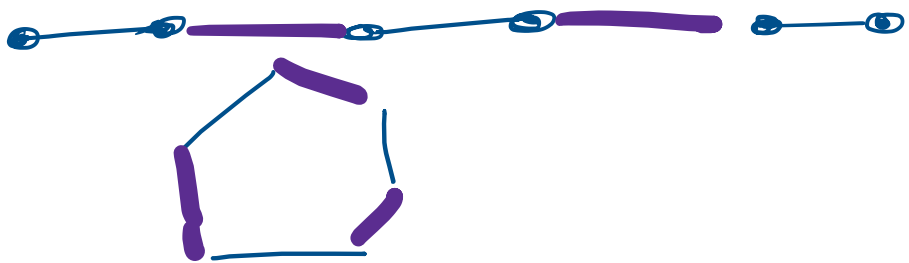
что это тогда?

≤2!

Задача о максимальном паросочетании

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Что это тогда?

≤ 2 !

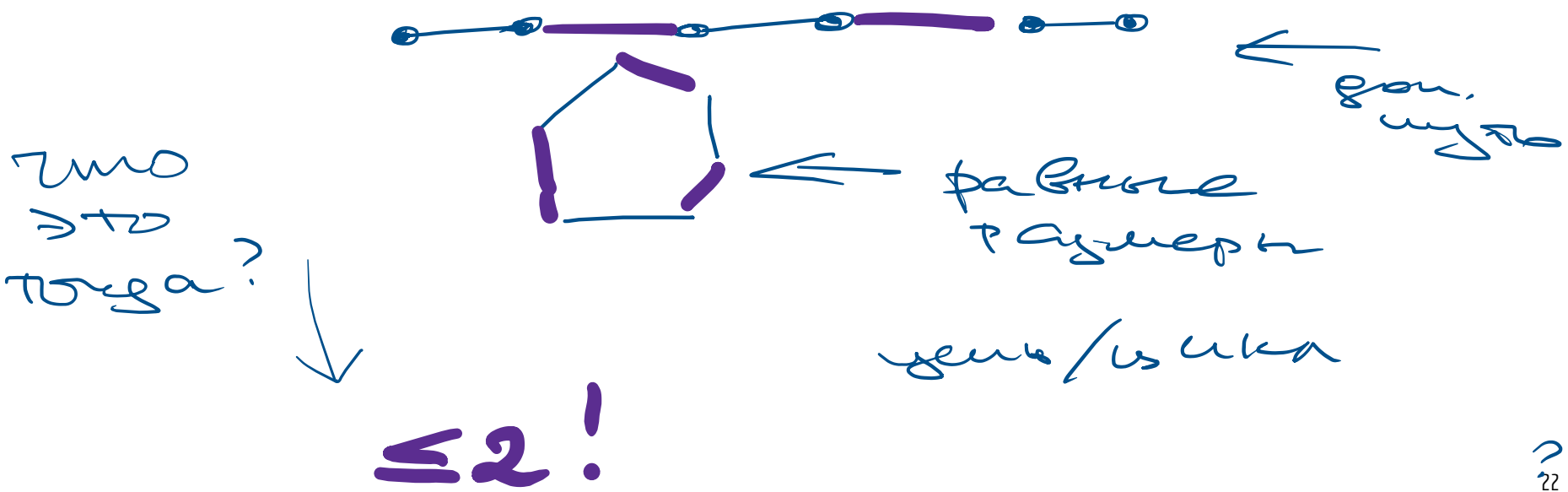
доказательство

?

Задача о максимальном паросочетании

Теорема:

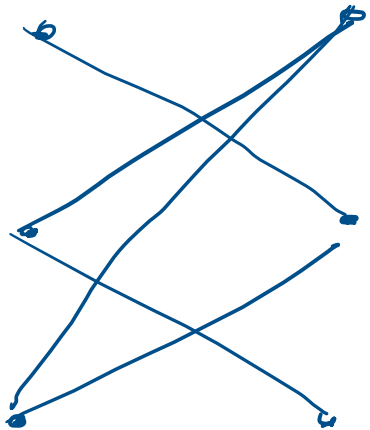
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании. Двудольный граф. Алгоритм Куна

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



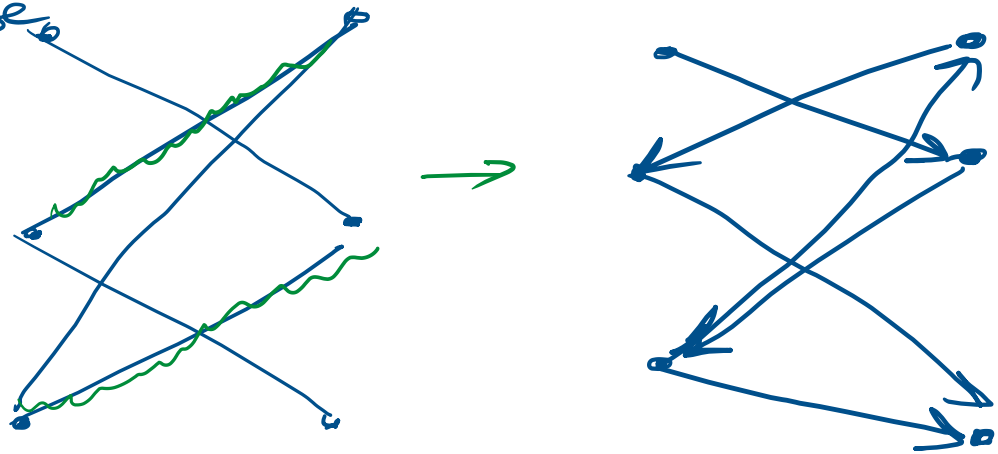
Задача о максимальном паросочетании. Двудольный граф. Алгоритм Куна

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.

ориент. вправо = свободное ребро

влево = занятое



Задача о максимальном паросочетании. Двудольный граф. Алгоритм Куна

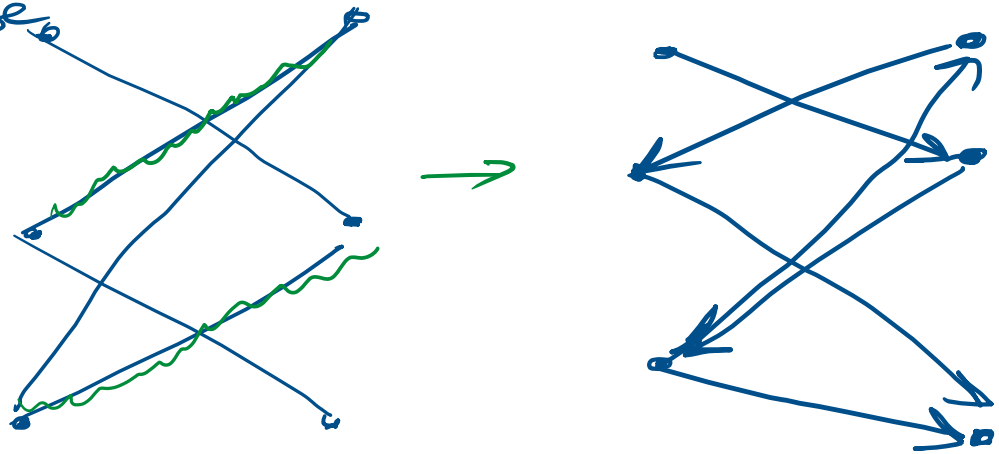
Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.

ориент. вправо = свободное ребро

влево = занятое

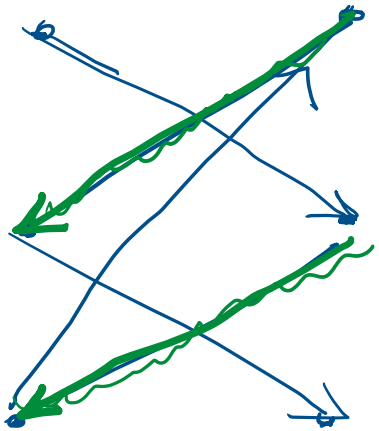
доп. путь?



Задача о максимальном паросочетании. Двудольный граф. Алгоритм Куна

Теорема:

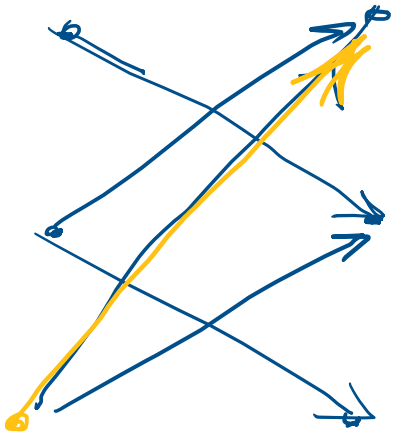
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании. Двудольный граф. Алгоритм Куна

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.

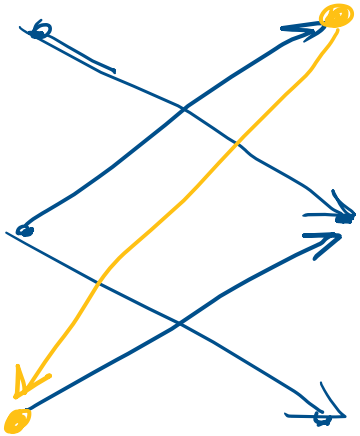


путь чередующ.
своб./зан.
вершину

Задача о максимальном паросочетании. Двудольный граф. Алгоритм Куна

Теорема:

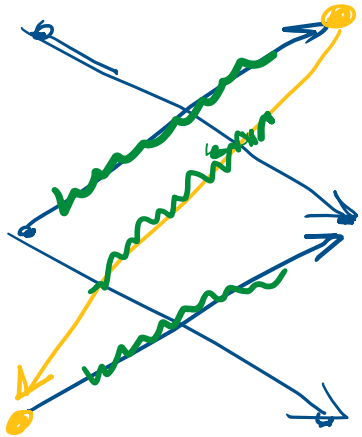
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании. Двудольный граф. Алгоритм Куна

Теорема:

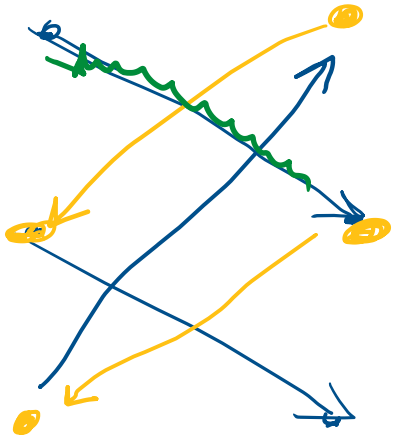
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании. Двудольный граф. Алгоритм Куна

Теорема:

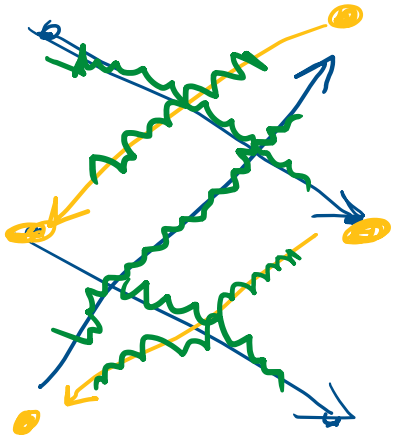
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании. Двудольный граф

Теорема:

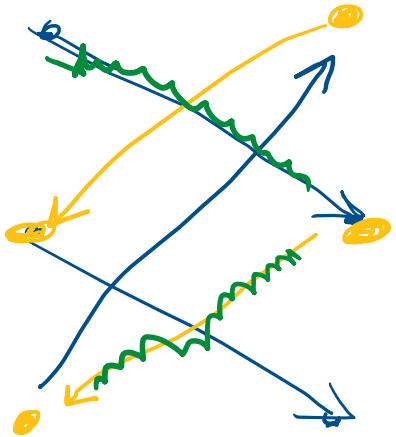
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании. Двудольный граф

Теорема:

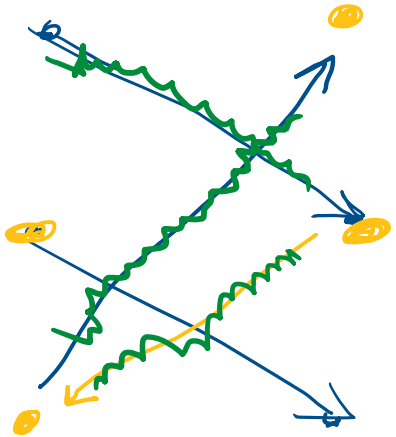
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании. Двудольный граф

Теорема:

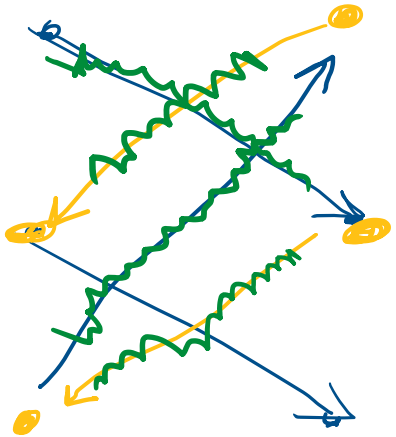
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании. Двудольный граф

Теорема:

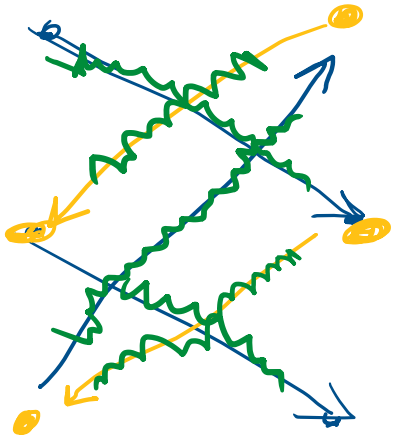
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании. Двудольный граф

Теорема:

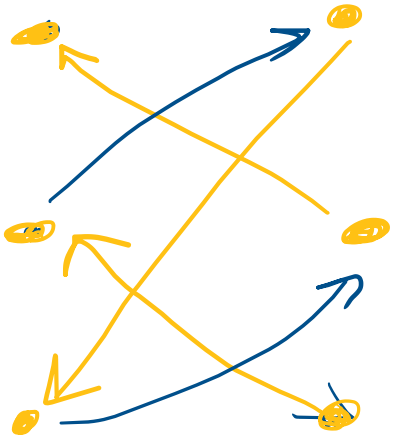
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании. Двудольный граф

Теорема:

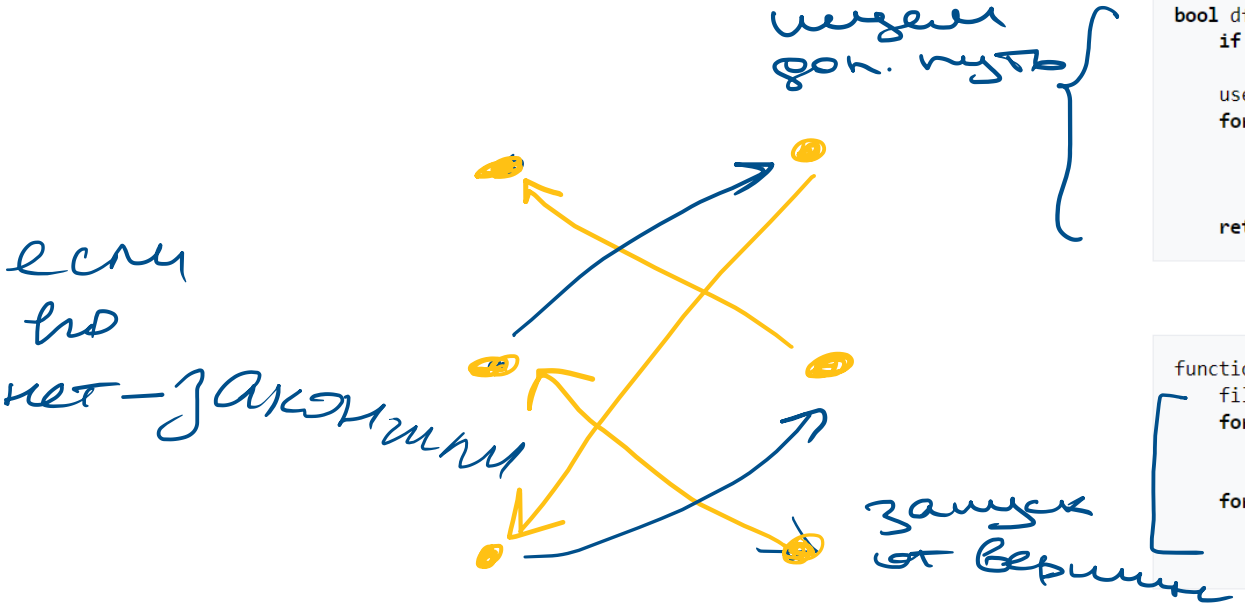
Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



Задача о максимальном паросочетании. Двудольный граф

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



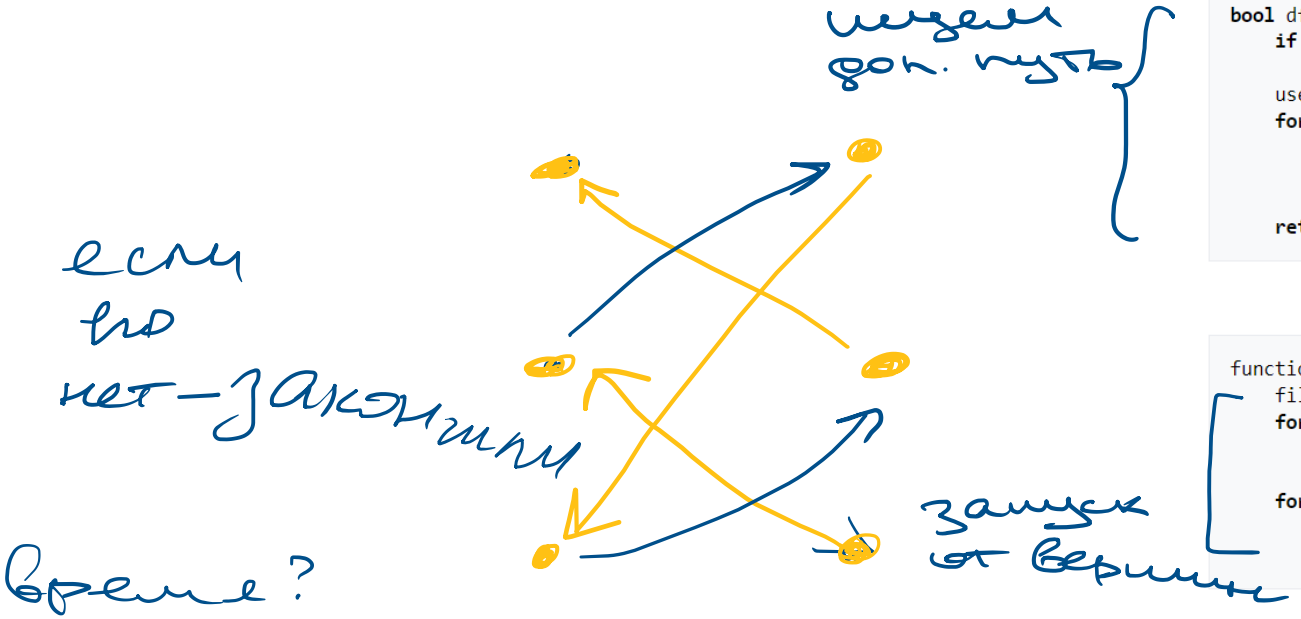

```
bool dfs(v: int):
    if (used[v])
        return false
    used[v] = true
    for to in g[v]
        if (matching[to] == -1 or dfs(matching[to])):
            matching[to] = v
            return true
    return false
```

```
function main():
    fill(matching, -1)
    for i = 1..n
        fill(used, false)
        dfs(i)
    for i = 1..n
        if (matching[i] != -1)
            print(i, " ", matching[i])
```

Задача о максимальном паросочетании. Двудольный граф

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.



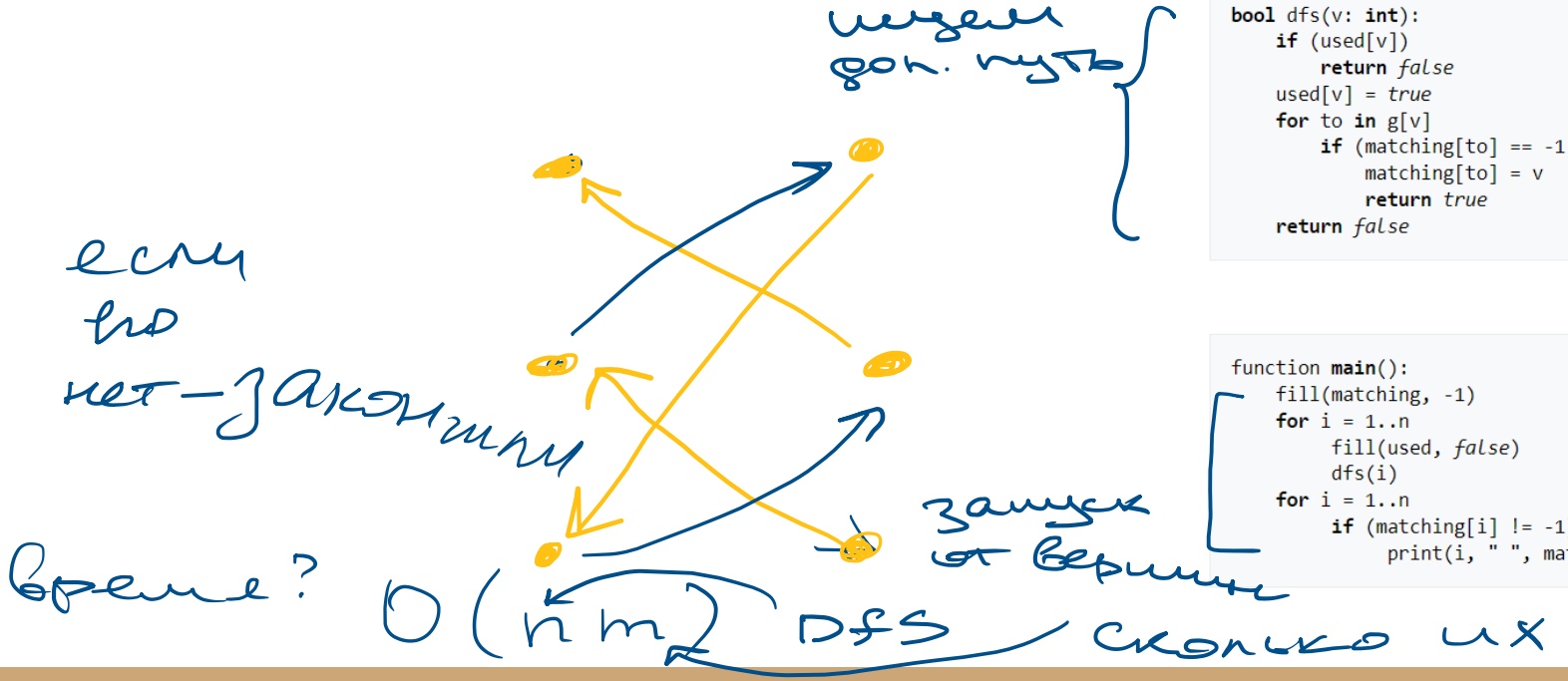

```
bool dfs(v: int):
    if (used[v])
        return false
    used[v] = true
    for to in g[v]
        if (matching[to] == -1 or dfs(matching[to])):
            matching[to] = v
            return true
    return false
```

```
function main():
    fill(matching, -1)
    for i = 1..n
        fill(used, false)
        dfs(i)
    for i = 1..n
        if (matching[i] != -1)
            print(i, " ", matching[i])
```

Задача о максимальном паросочетании. Двудольный граф

Теорема:

Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.




```
bool dfs(v: int):
    if (used[v])
        return false
    used[v] = true
    for to in g[v]
        if (matching[to] == -1 or dfs(matching[to])):
            matching[to] = v
            return true
    return false
```

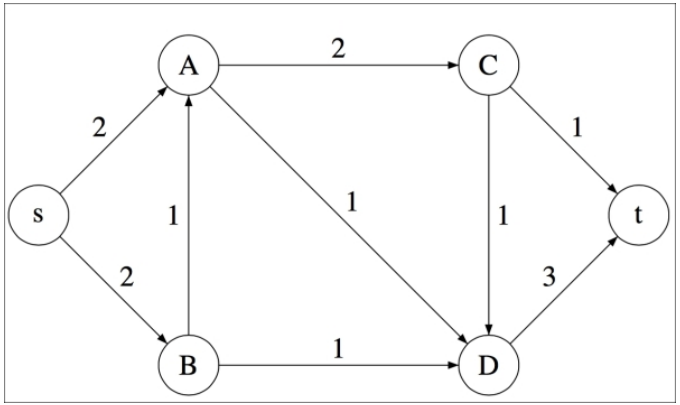
```
function main():
    fill(matching, -1)
    for i = 1..n
        fill(used, false)
        dfs(i)
    for i = 1..n
        if (matching[i] != -1)
            print(i, " ", matching[i])
```

Задача о максимальном потоке. Сеть

Определение:

Сеть (англ. *flow network*) $G = (V, E)$ представляет собой ориентированный граф, в котором каждое ребро $(u, v) \in E$ имеет положительную **пропускную способность** (англ. *capacity*) $c(u, v) > 0$. Если $(u, v) \notin E$, предполагается что $c(u, v) = 0$.

В транспортной сети выделяются две вершины: **исток** s и **сток** t.

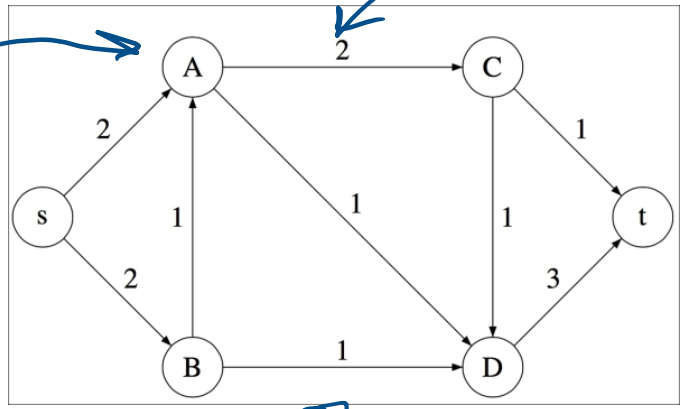


Задача о максимальном потоке. Сеть

Определение:

Сеть (англ. *flow network*) $G = (V, E)$ представляет собой ориентированный граф, в котором каждое ребро $(u, v) \in E$ имеет положительную **пропускную способность** (англ. *capacity*) $c(u, v) > 0$. Если $(u, v) \notin E$, предполагается что $c(u, v) = 0$.

В транспортной сети выделяются две вершины: **исток** s и **сток** t.



Задача о максимальном потоке. Поток

Определение:

Потоком (англ. *flow*) f в G является действительная функция $f\colon V \times V \to R$, удовлетворяющая условиям:

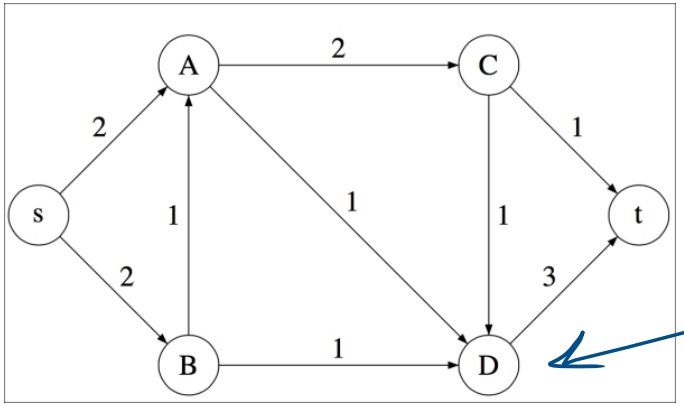
1) $f(u, v) = -f(v, u)$ (антисимметричность);

2) $f(u, v) \leqslant c(u, v)$ (ограничение пропускной способности), если ребра нет, то $f(u, v) = 0$;

3) $\sum\limits_{v} f(u, v) = 0$ для всех вершин u, кроме s и t (закон сохранения потока).

Величина потока f определяется как $|f| = \sum\limits_{v \in V} f(s, v)$.

сколько жижи течет через трубу



согласованно в узлах

Задача о максимальном потоке. Пример потока

Определение:

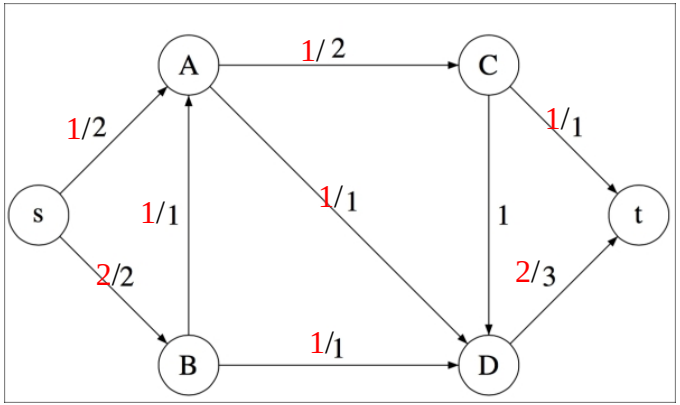
Потоком (англ. *flow*) f в G является действительная функция $f: V \times V \to R$, удоволетворяющая условиям:

1) $f(u, v) = -f(v, u)$ (антисимметричность);

2) $f(u, v) \leqslant c(u, v)$ (ограничение пропускной способности), если ребра нет, то $f(u, v) = 0$;

3) $\sum\limits_{v} f(u, v) = 0$ для всех вершин u, кроме s и t (закон сохранения потока).

Величина потока f определяется как $|f| = \sum\limits_{v \in V} f(s, v)$.

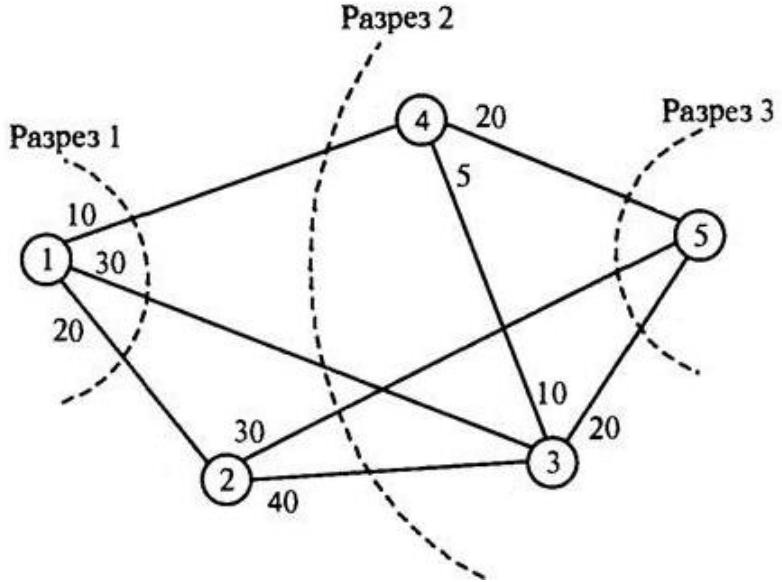


Задача о максимальном потоке. Разрез

Определение:

(s,t)-разрезом (англ. *s-t cut*) $\langle S,T\rangle$ в сети G называется пара множеств S,T, удоволетворяющих условиям:

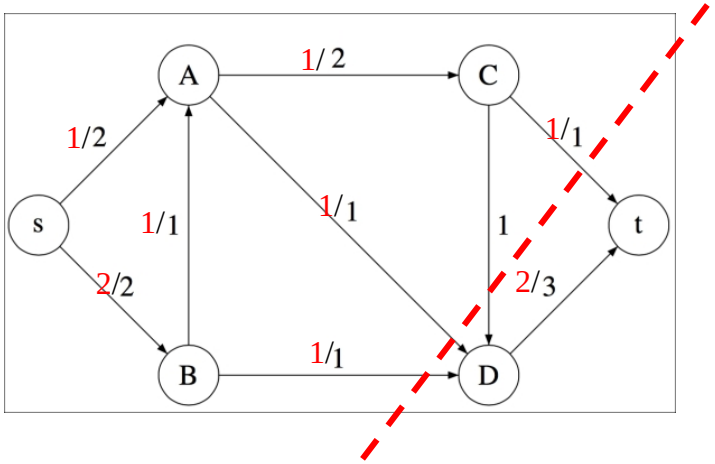
1. $s \in S, t \in T$
2. $S = V \setminus T$



Задача о максимальном потоке. Разрез

Поток через разрез
$f(S, T) = 3$

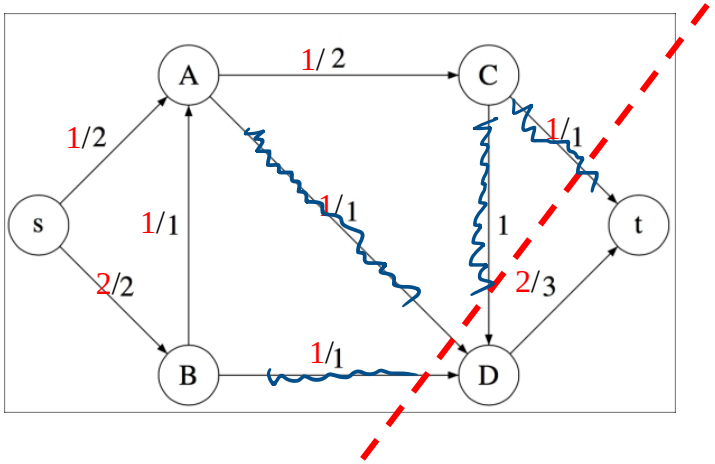
Пропускная способность разреза
$c(S, T) = 4$



Задача о максимальном потоке. Разрез

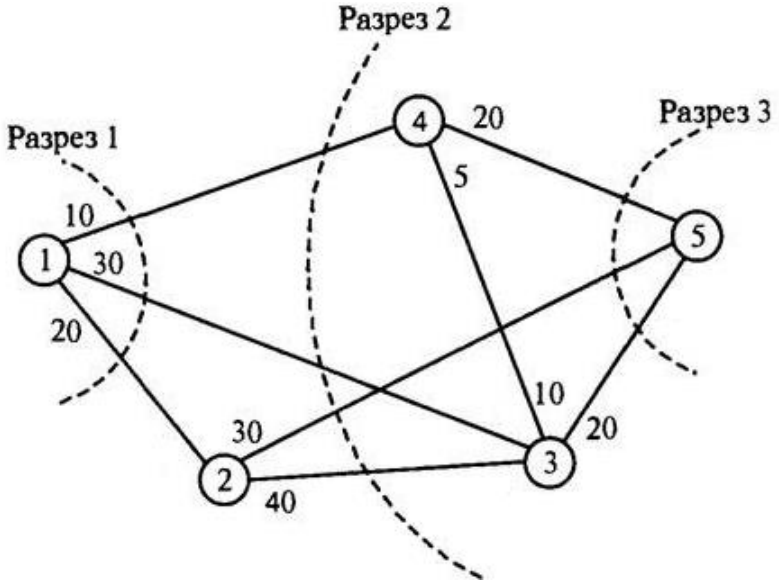
Поток через разрез
$f(S, T) = 3$

Пропускная способность разреза
$c(S, T) = 4$



Задача о максимальном потоке. Леммы о разрезе

- Минимальный разрез - разрез с минимальной пропускной способностью
- Поток в сети $|f|$ равен потоку разреза $f(S, T)$
- $f(S, T) \leq c(S, T)$ - и для двух **разных** разрезов тоже
- Если $f(S, T) = c(S, T)$, то поток - максимален, а разрез - минимален



Задача о максимальном потоке. Остаточная сеть

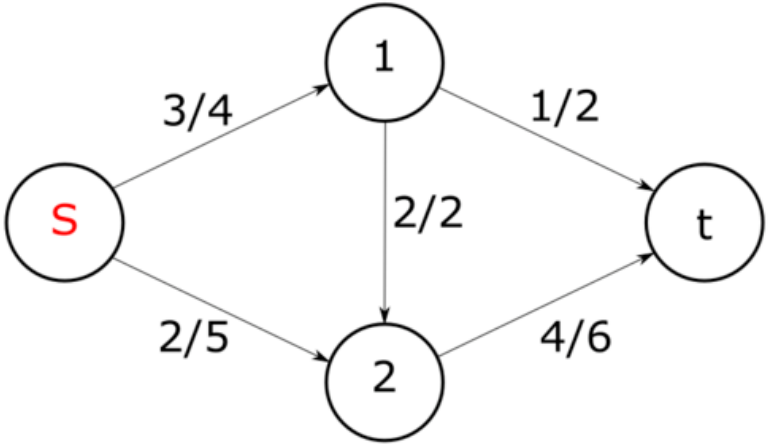
Определение:

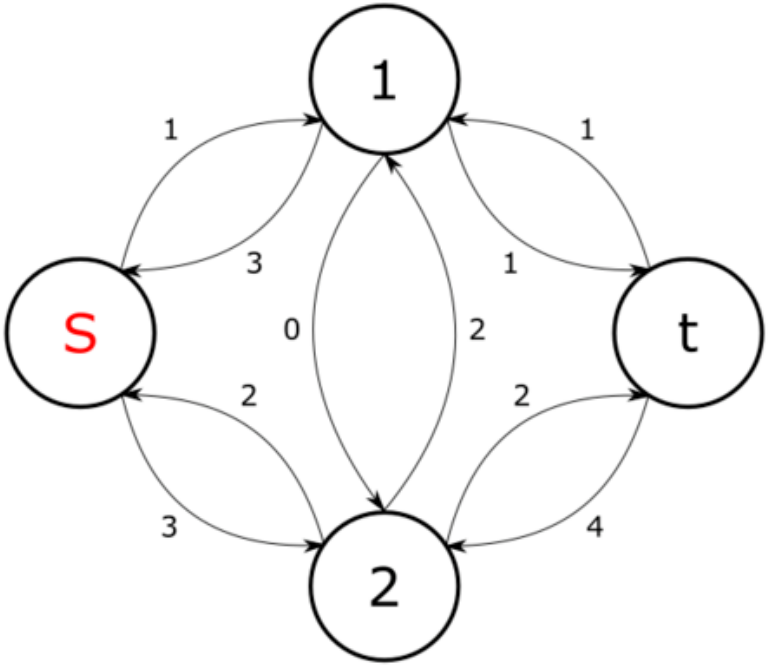
Остаточной пропускной способностью (англ. *residual capacity*) ребра (u, v) называется величина дополнительного потока, который мы можем направить из u в v, не превысив пропускную способность $c(u, v)$. Иными словами $c_f(u, v) = c(u, v) - f(u, v)$.

Определение:

Для заданной транспортной сети $G = (V, E)$ и потока f, **остаточной сетью**, (**дополняющая сеть**, англ. *residual network*) в G, порожденной потоком f, является сеть $G_f = (V, E_f)$, где $E_f = \{(u, v) \in V \times V \mid c_f(u, v) > 0\}$

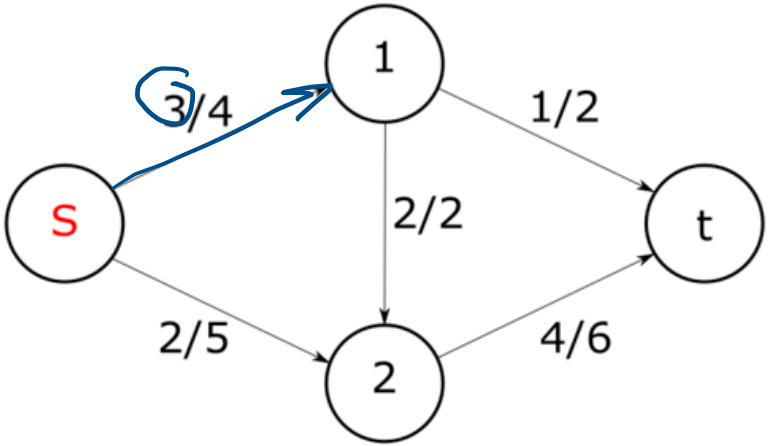
Задача о максимальном потоке. Остаточная сеть

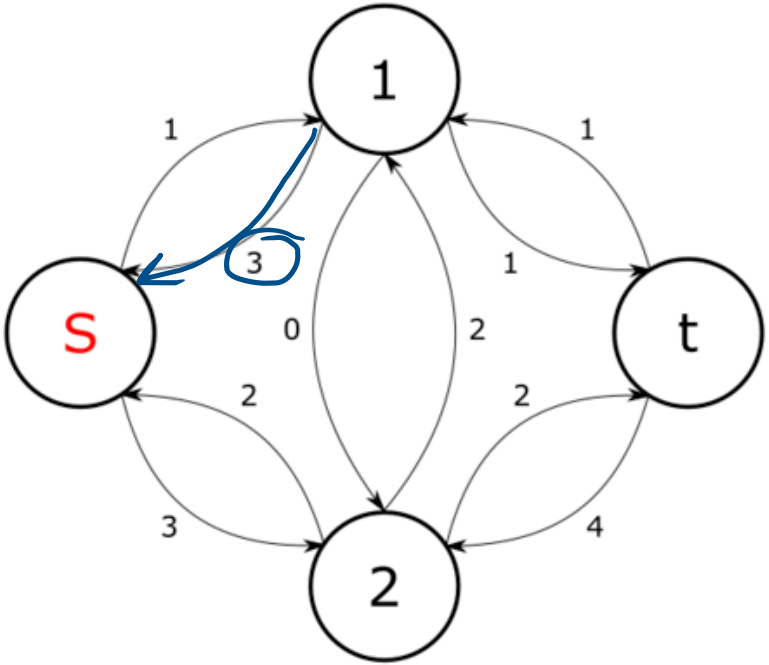




обратные ребра появляются из-за свойства антисимметричности (см. 47 сл.)

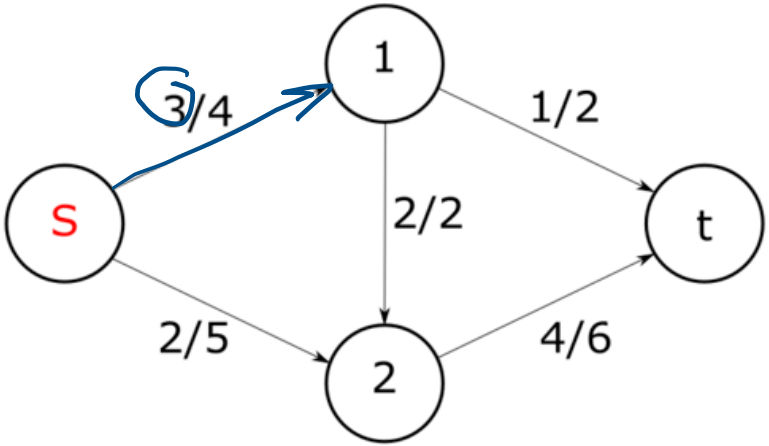
Задача о максимальном потоке. Остаточная сеть

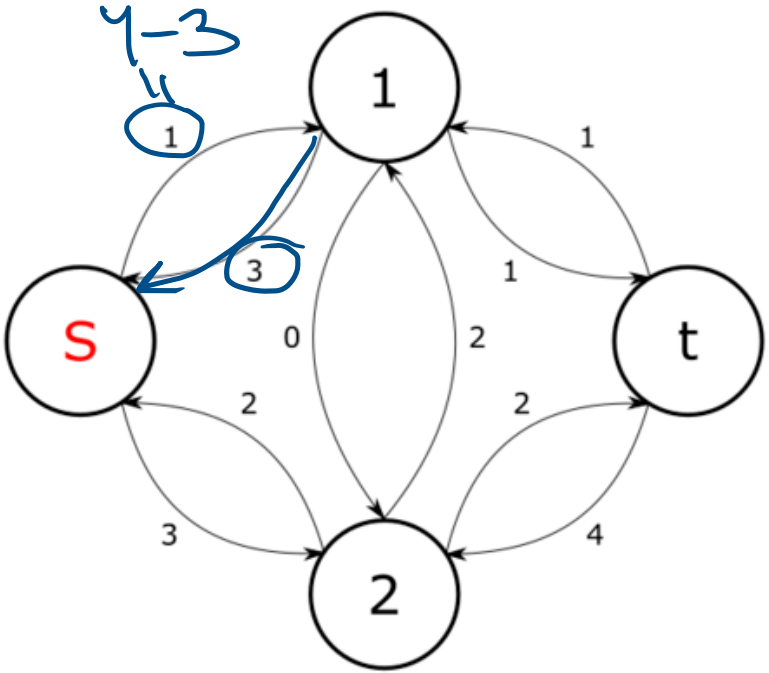




обратные ребра появляются из-за свойства антисимметричности (см. 47 сл.)

Задача о максимальном потоке. Остаточная сеть



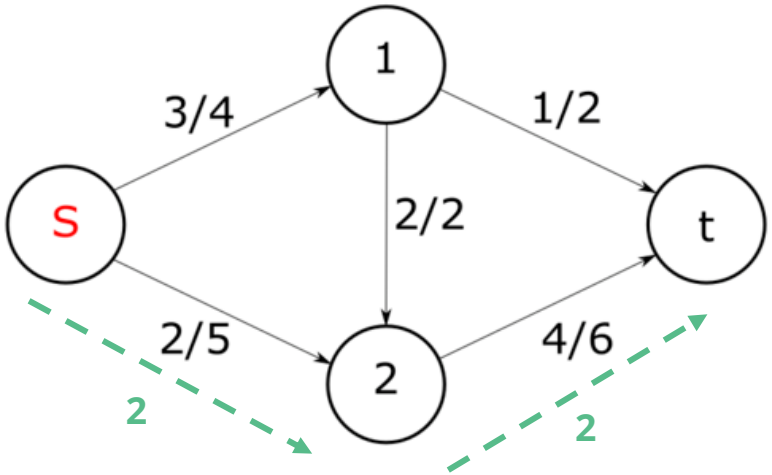


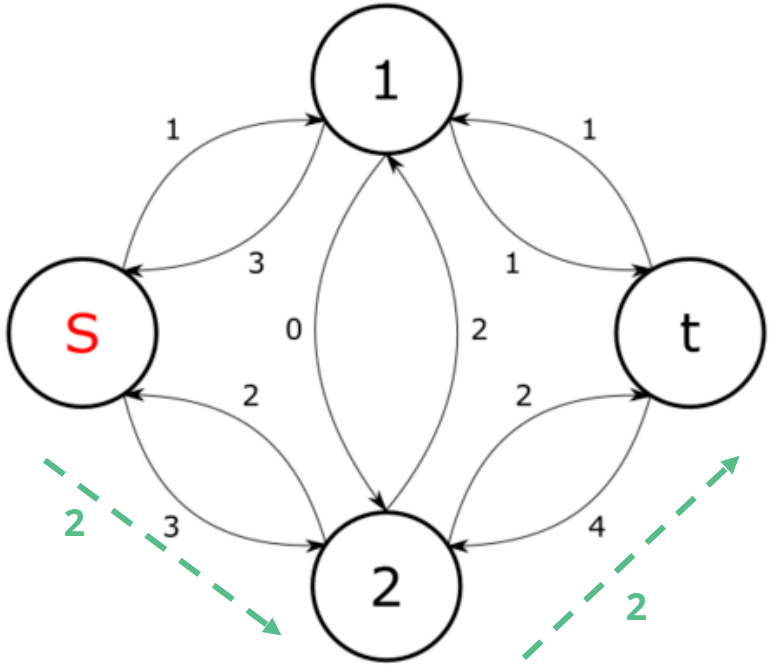
обратные ребра появляются из-за свойства антисимметричности (см. 47 сл.)

Задача о максимальном потоке. Дополняющий путь

Определение:

Для заданной транспортной сети $G = (V, E)$ и потока f **дополняющим путем** (англ. *augmenting path*) p является простой путь из истока в сток в остаточной сети $G_f = (V, E_f)$.

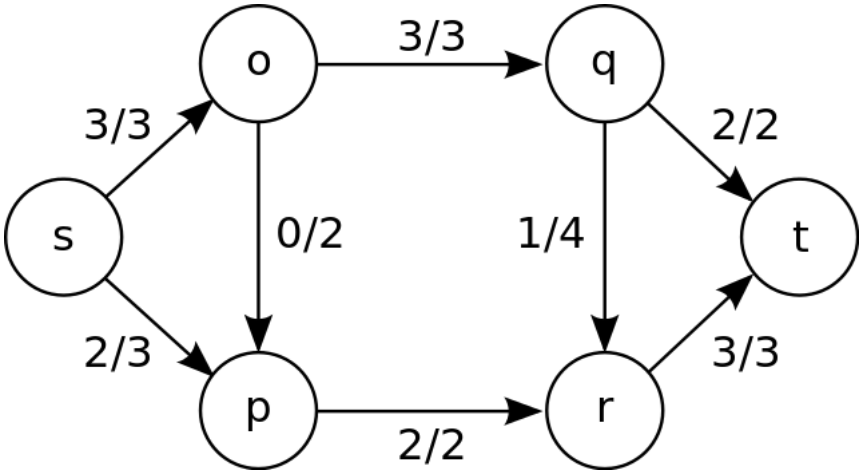


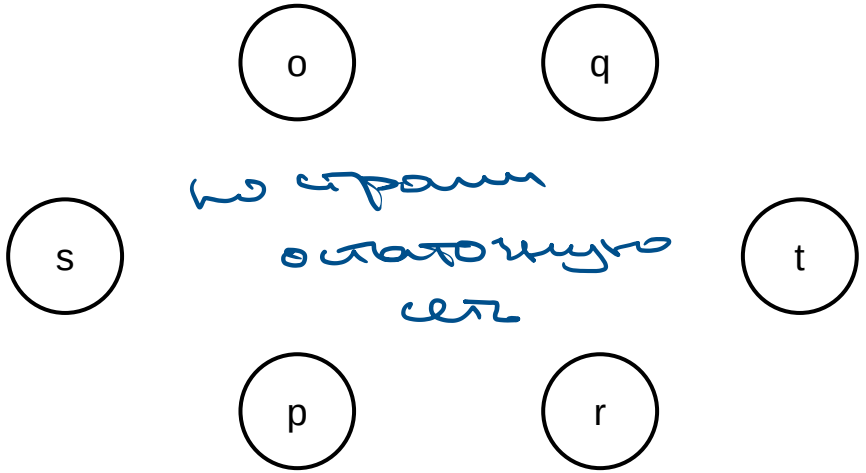


Теорема Форда-Фалкерсона

Если f — некоторый поток в сети $G = (V, E)$ с источником s и стоком t, то следующие утверждения эквивалентны:

1. Поток f максимален.
2. В G_f не существует пути $s \rightsquigarrow t$.
3. $|f| = c(S, T)$ для некоторого разреза (S, T) сети G.

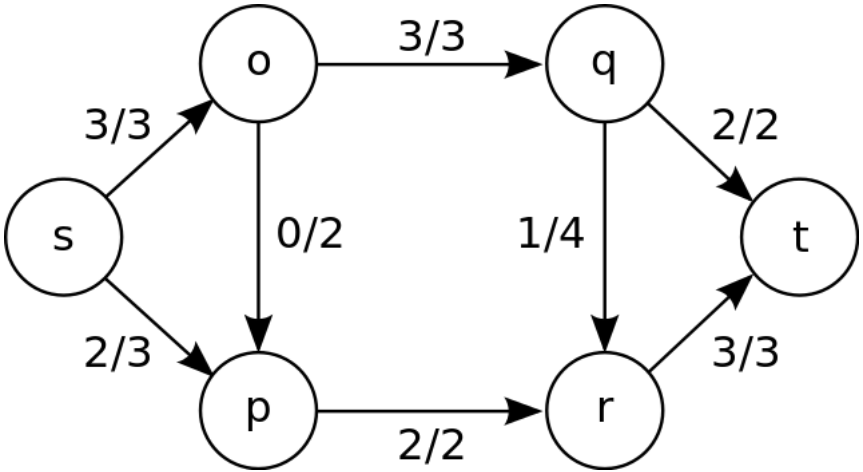


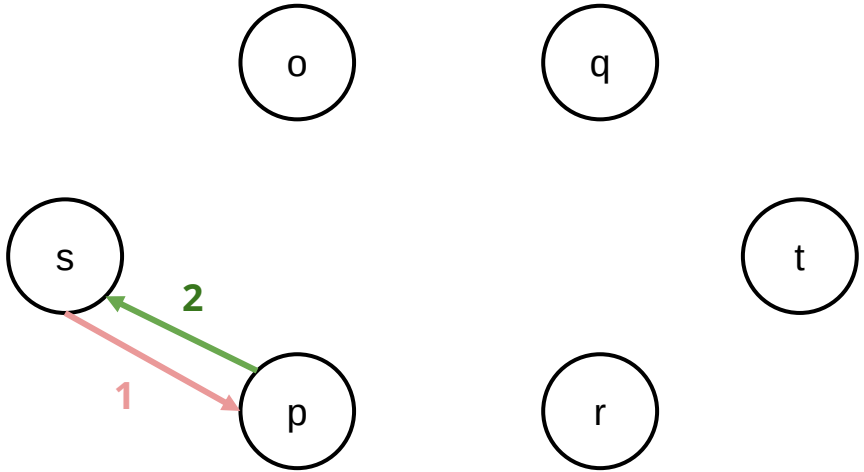


Теорема Форда-Фалкерсона

Если f — некоторый поток в сети $G = (V, E)$ с источником s и стоком t, то следующие утверждения эквивалентны:

1. Поток f максимален.
2. В G_f не существует пути $s \rightsquigarrow t$.
3. $|f| = c(S, T)$ для некоторого разреза (S, T) сети G.

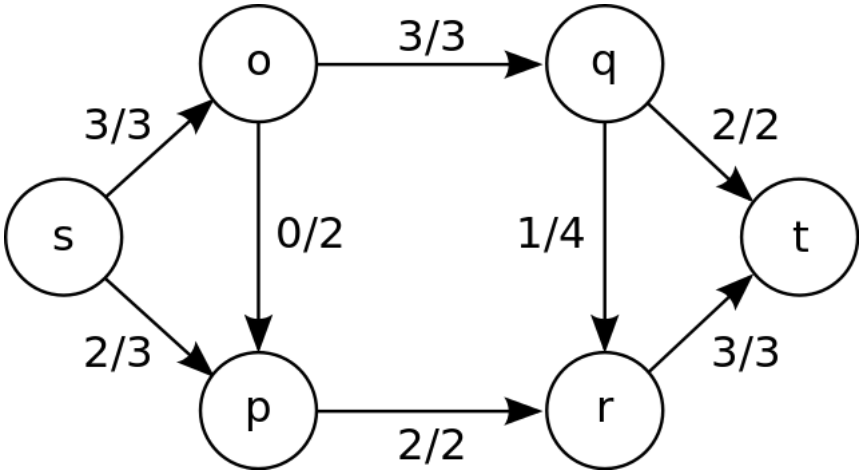


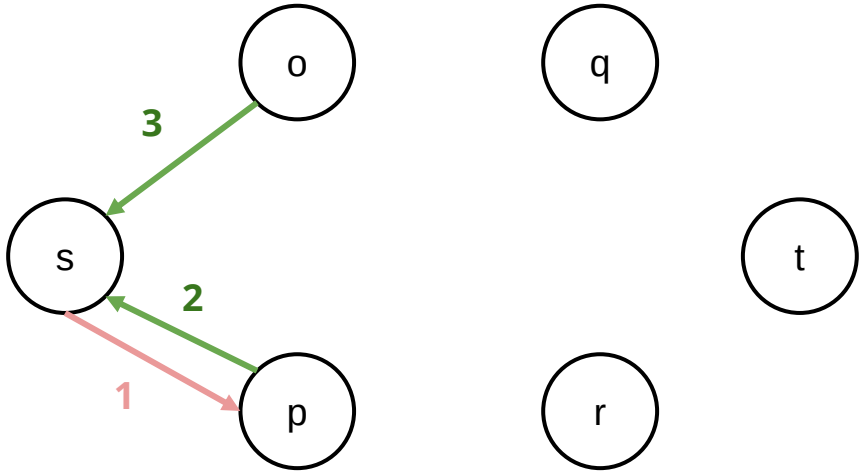


Теорема Форда-Фалкерсона

Если f — некоторый поток в сети $G = (V, E)$ с источником s и стоком t, то следующие утверждения эквивалентны:

1. Поток f максимален.
2. В G_f не существует пути $s \rightsquigarrow t$.
3. $|f| = c(S, T)$ для некоторого разреза (S, T) сети G.

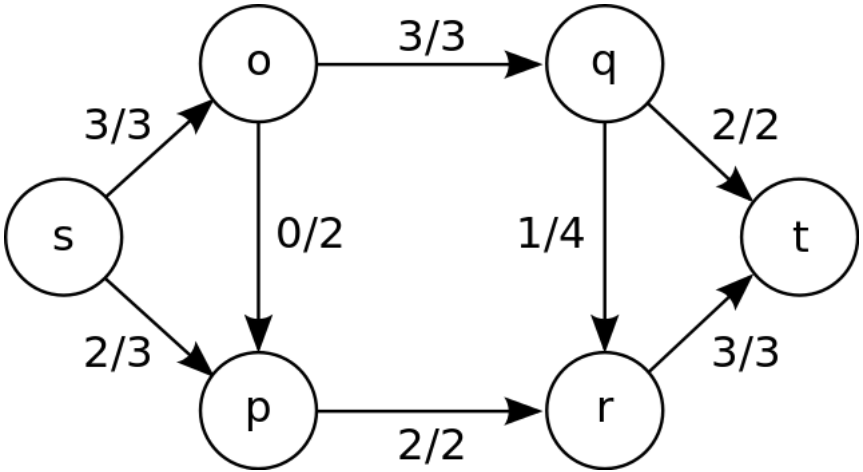


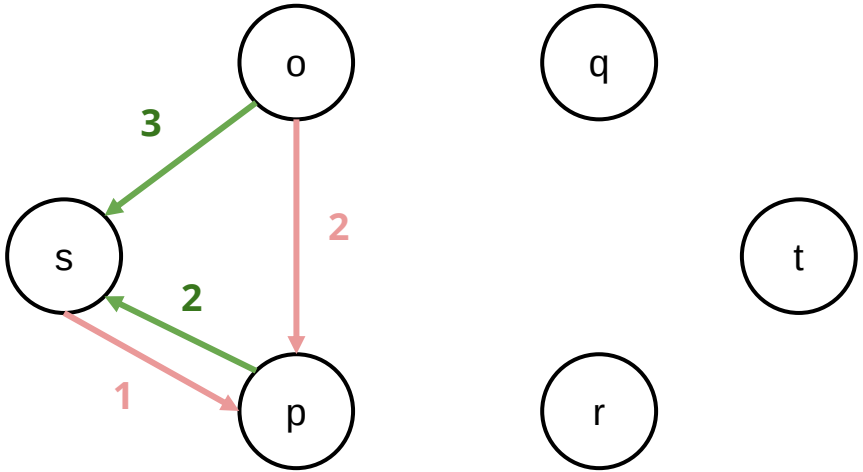


Теорема Форда-Фалкерсона

Если f — некоторый поток в сети $G = (V, E)$ с источником s и стоком t, то следующие утверждения эквивалентны:

1. Поток f максимален.
2. В G_f не существует пути $s \rightsquigarrow t$.
3. $|f| = c(S, T)$ для некоторого разреза (S, T) сети G.

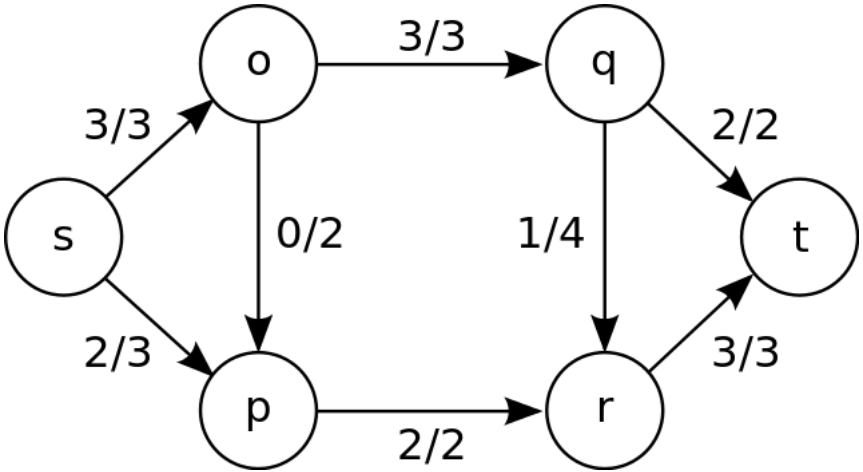


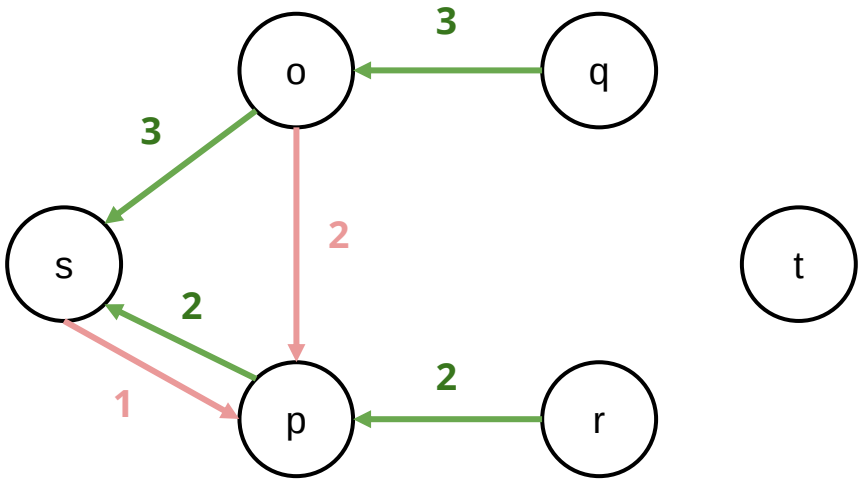


Теорема Форда-Фалкерсона

Если f — некоторый поток в сети $G = (V, E)$ с источником s и стоком t, то следующие утверждения эквивалентны:

1. Поток f максимален.
2. В G_f не существует пути $s \rightsquigarrow t$.
3. $|f| = c(S, T)$ для некоторого разреза (S, T) сети G.

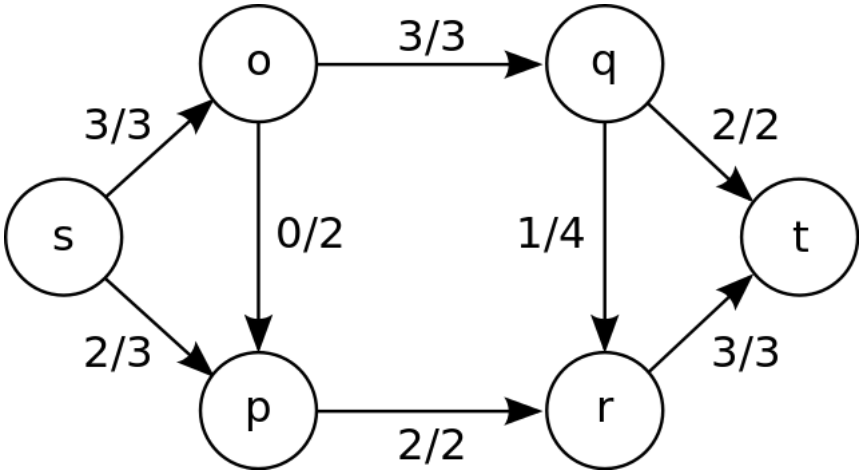


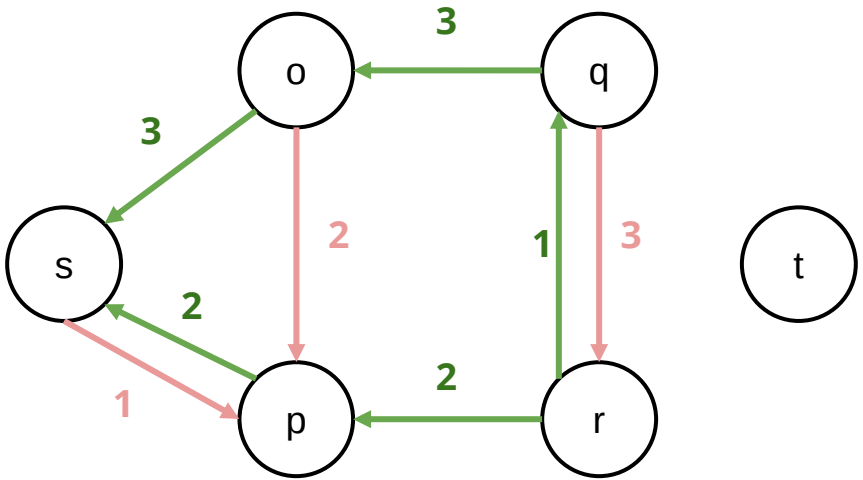


Теорема Форда-Фалкерсона

Если f — некоторый поток в сети $G = (V, E)$ с источником s и стоком t, то следующие утверждения эквивалентны:

1. Поток f максимален.
2. В G_f не существует пути $s \rightsquigarrow t$.
3. $|f| = c(S, T)$ для некоторого разреза (S, T) сети G.

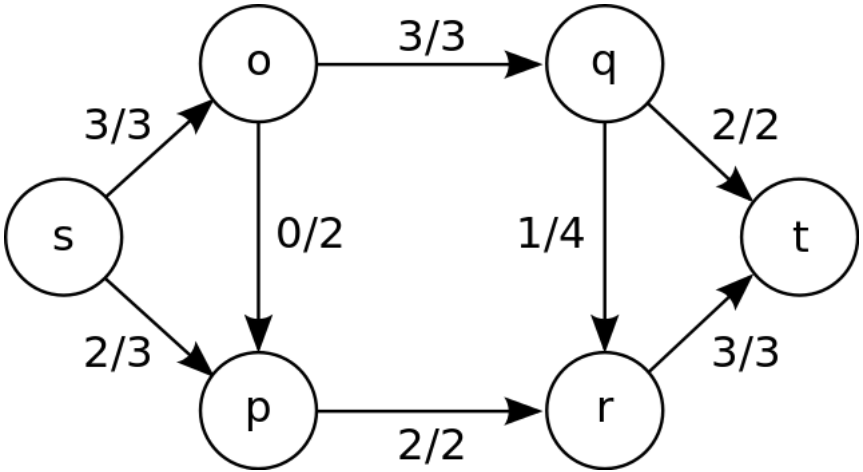


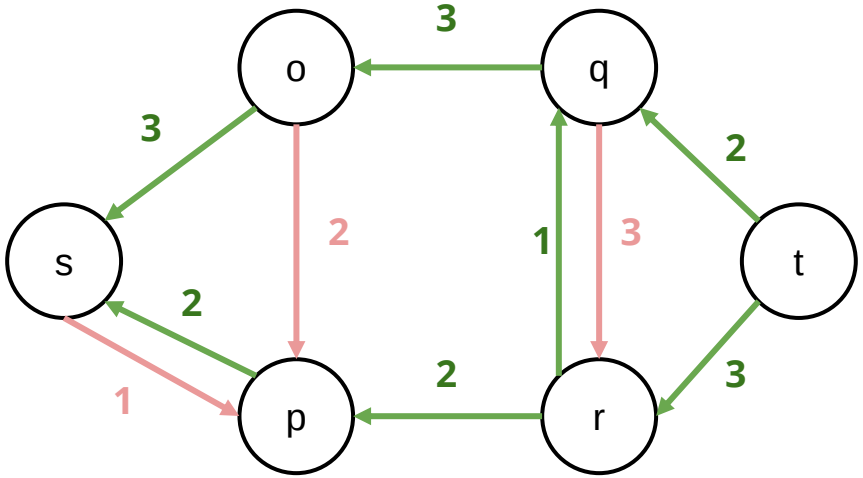


Теорема Форда-Фалкерсона

Если f — некоторый поток в сети $G = (V, E)$ с источником s и стоком t, то следующие утверждения эквивалентны:

1. Поток f максимален.
2. В G_f не существует пути $s \rightsquigarrow t$.
3. $|f| = c(S, T)$ для некоторого разреза (S, T) сети G.

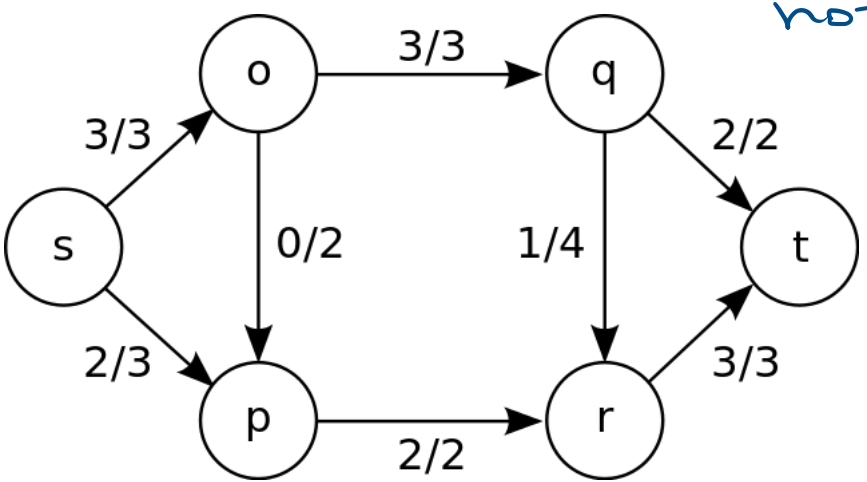




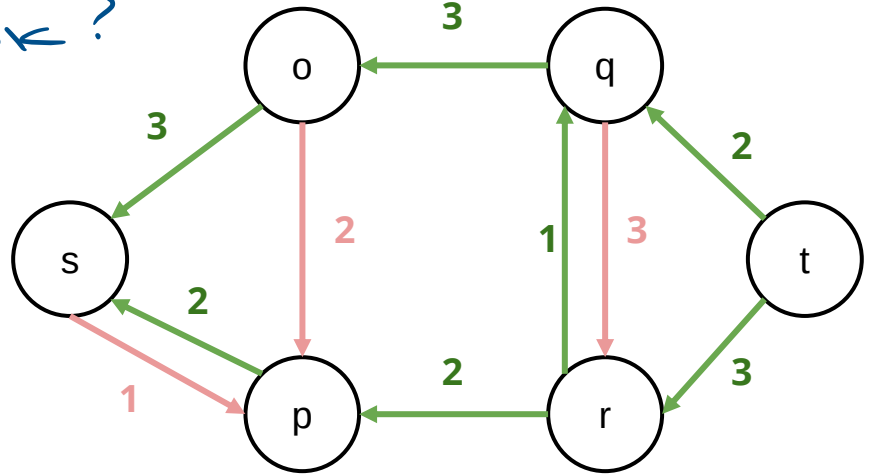
Теорема Форда-Фалкерсона

Если f — некоторый поток в сети $G = (V, E)$ с источником s и стоком t, то следующие утверждения эквивалентны:

1. Поток f максимален.
2. В G_f не существует пути $s \rightsquigarrow t$.
3. $|f| = c(S, T)$ для некоторого разреза (S, T) сети G.



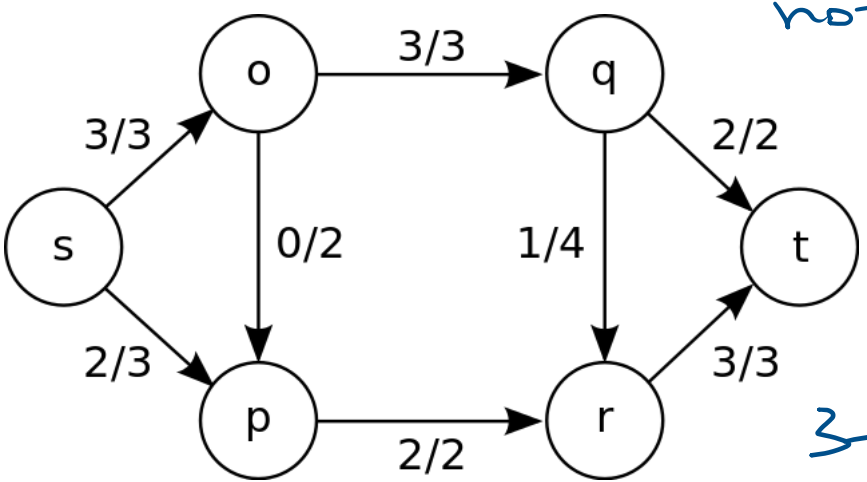
max поток ?



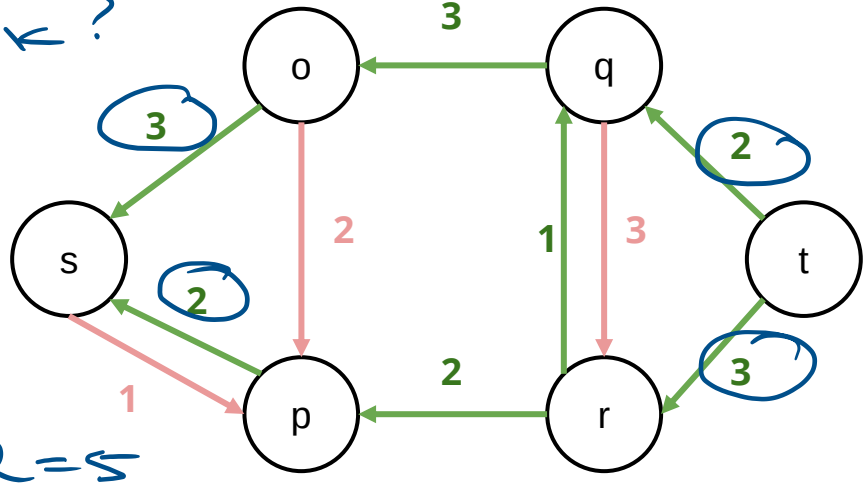
Теорема Форда-Фалкерсона

Если f — некоторый поток в сети $G = (V, E)$ с источником s и стоком t, то следующие утверждения эквивалентны:

1. Поток f максимален.
2. В G_f не существует пути $s \rightsquigarrow t$.
3. $|f| = c(S, T)$ для некоторого разреза (S, T) сети G.

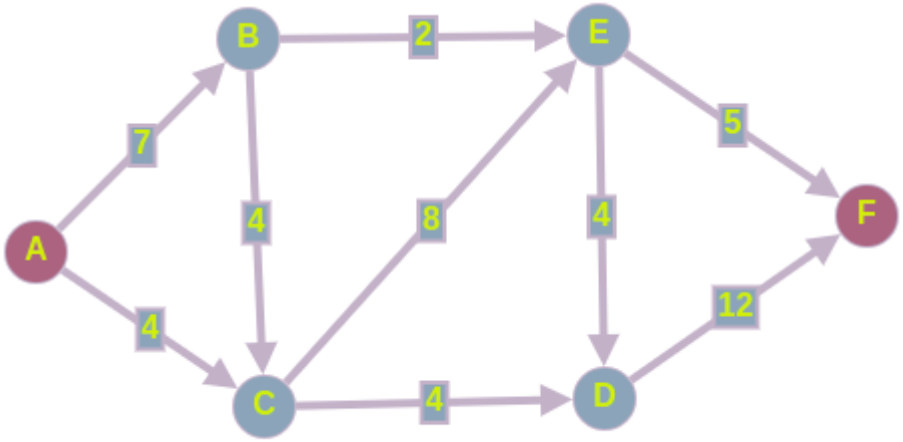


max поток?



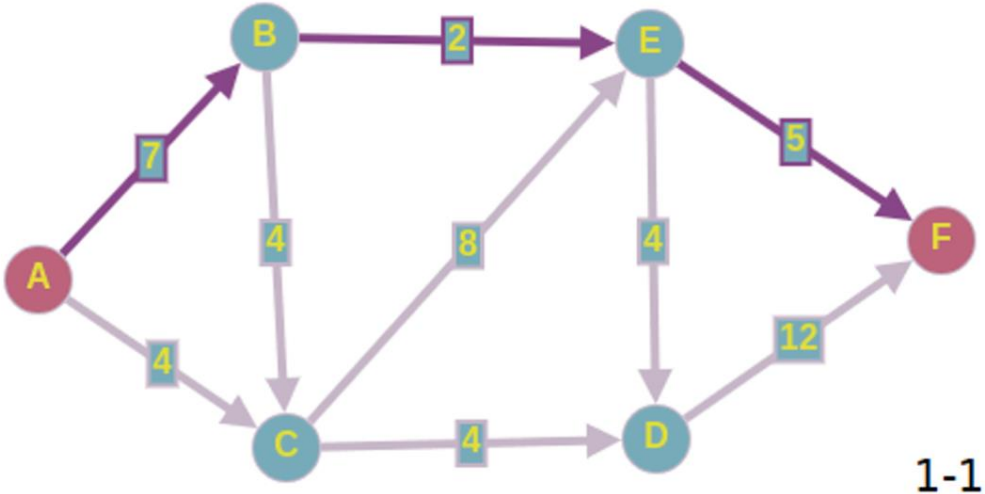
3+2=5

Алгоритм Форда-Фалкерсона

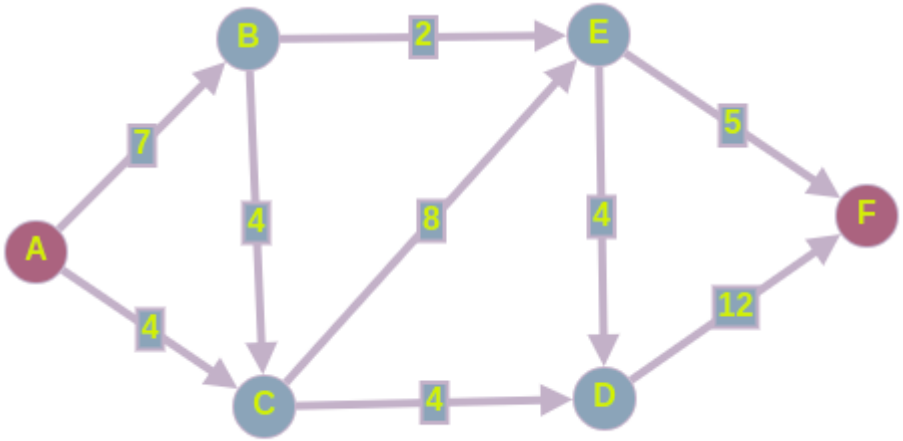


Исходный граф

Остаточная сеть



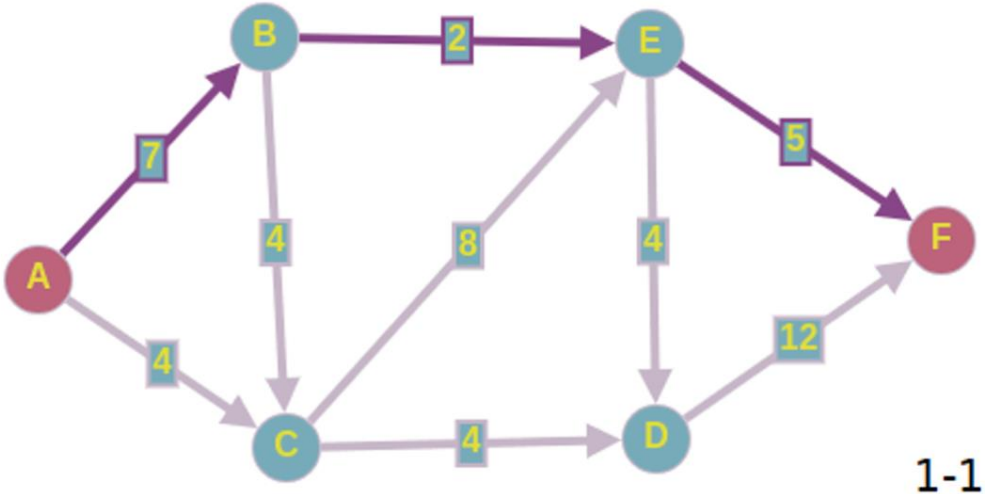
Алгоритм Форда-Фалкерсона



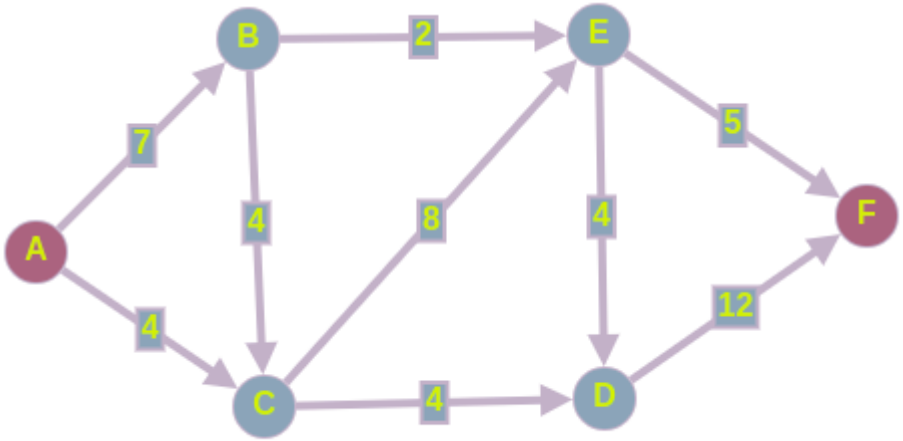
Исходный граф

поток = 2

Остаточная сеть

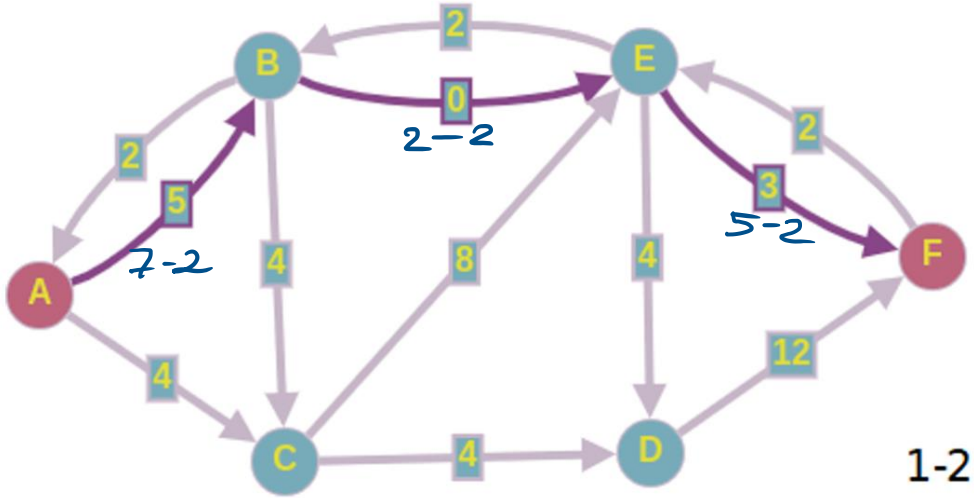


Алгоритм Форда-Фалкерсона

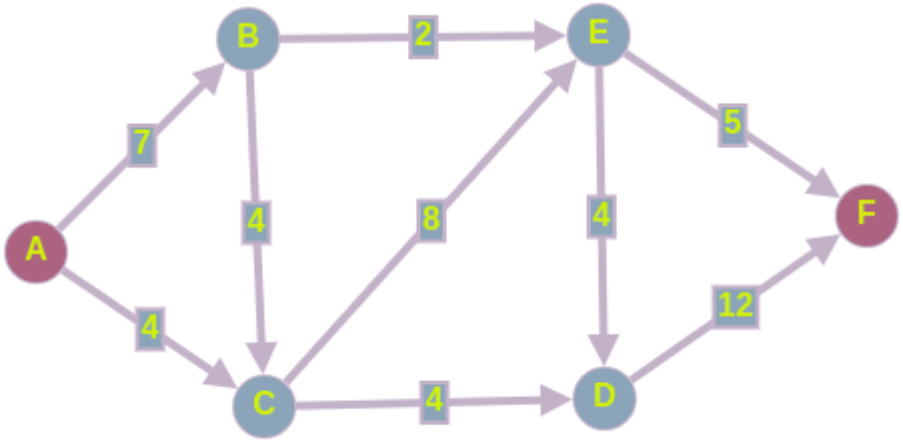


Исходный граф

Остаточная сеть



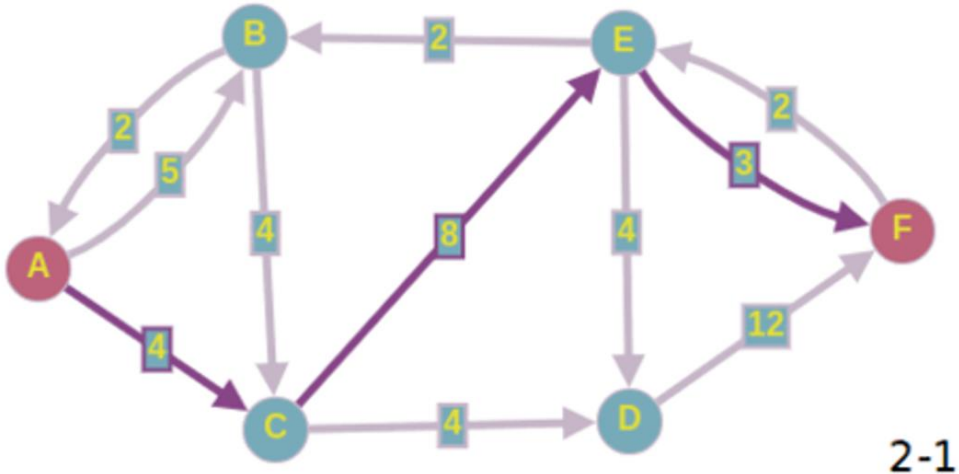
Алгоритм Форда-Фалкерсона



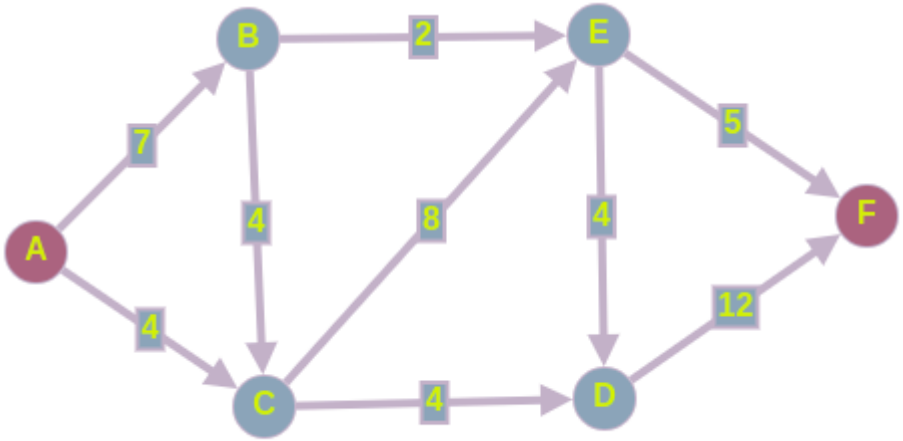
Исходный граф

Cmin - ?

Остаточная сеть



Алгоритм Форда-Фалкерсона

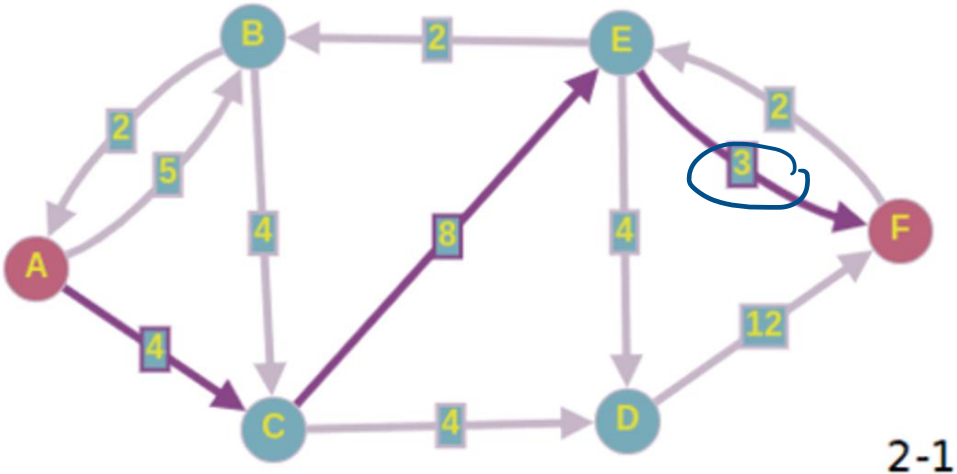


Исходный граф

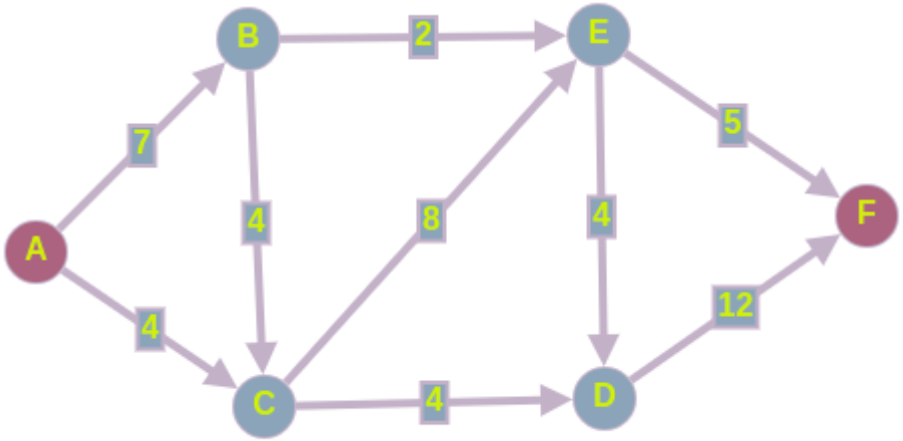
Cmin - ?

3

Остаточная сеть



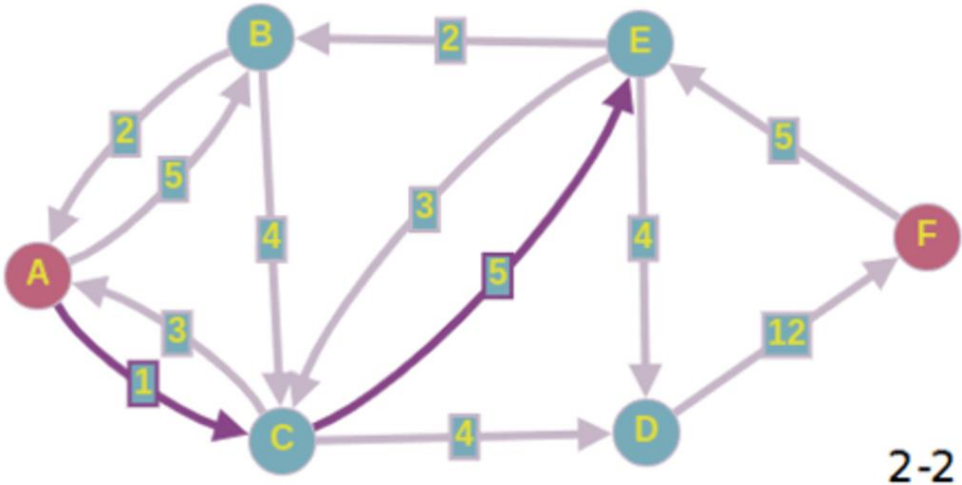
Алгоритм Форда-Фалкерсона



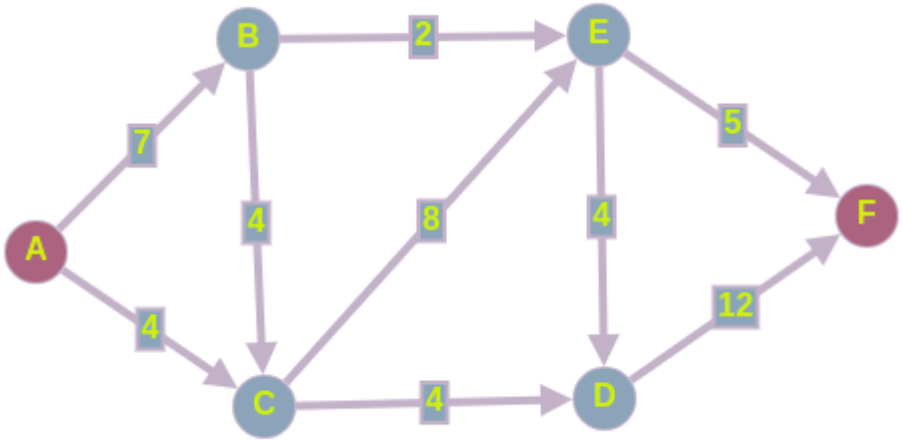
Исходный граф

Вычитаем на пути

Остаточная сеть

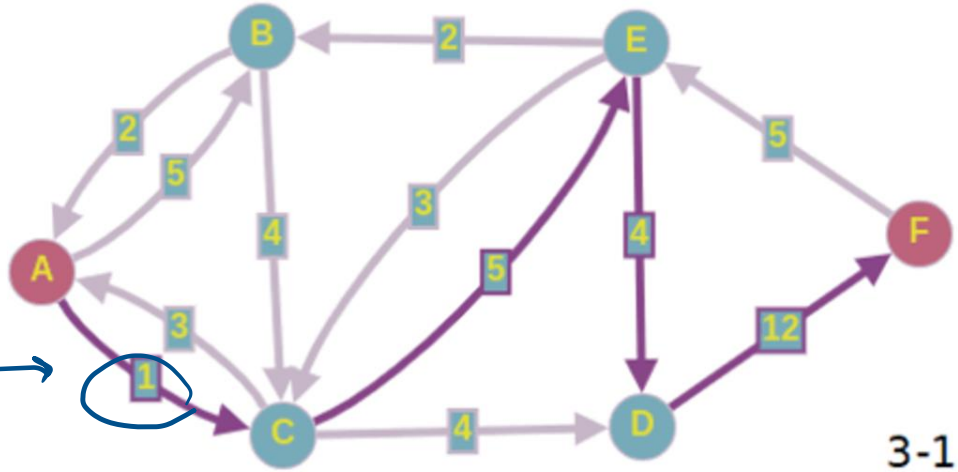


Алгоритм Форда-Фалкерсона

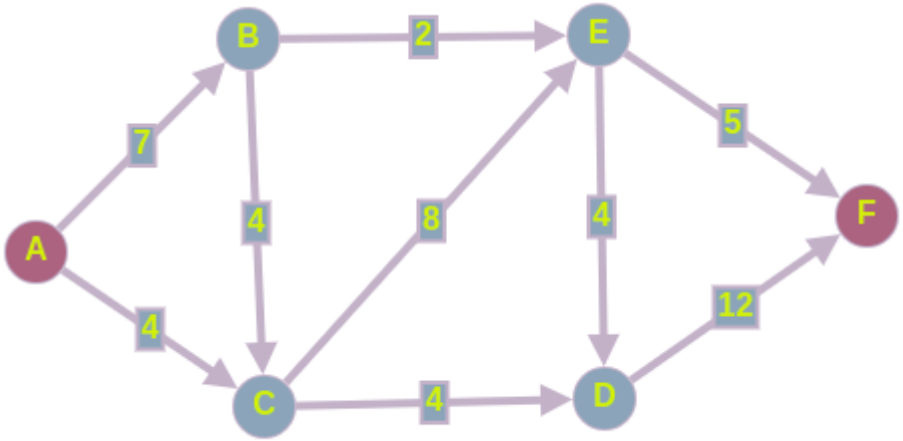


Исходный граф

Остаточная сеть

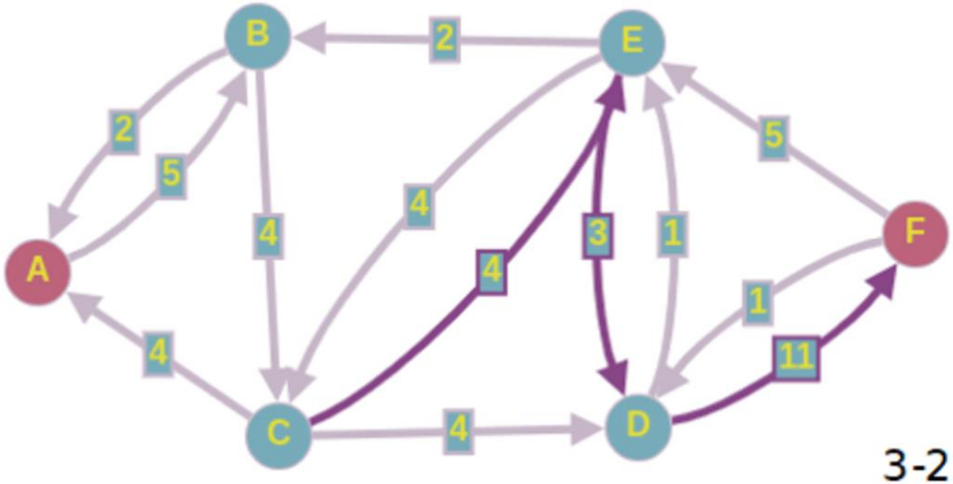


Алгоритм Форда-Фалкерсона

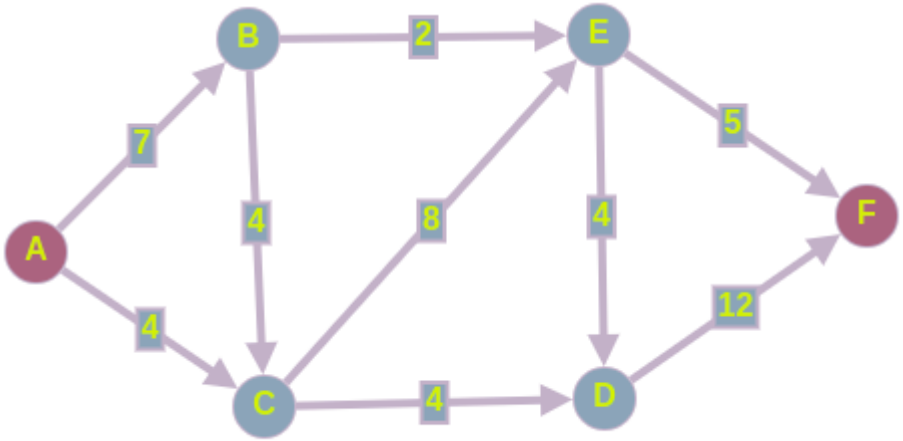


Исходный граф

Остаточная сеть

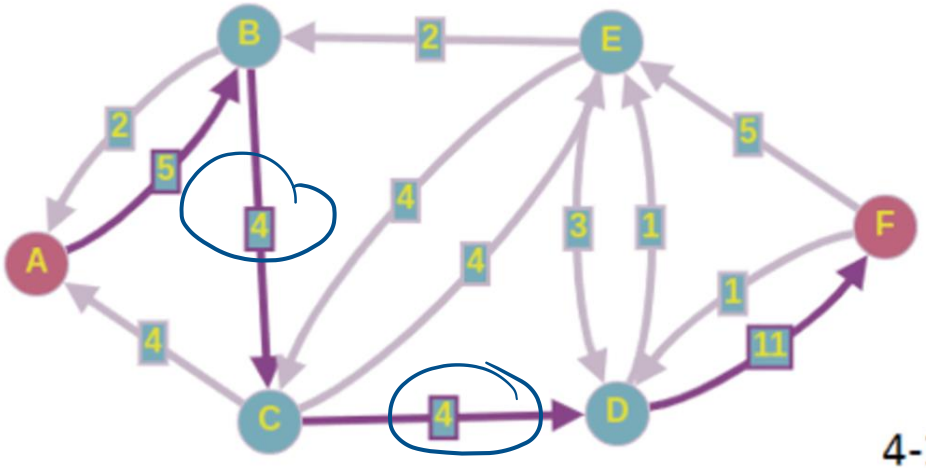


Алгоритм Форда-Фалкерсона

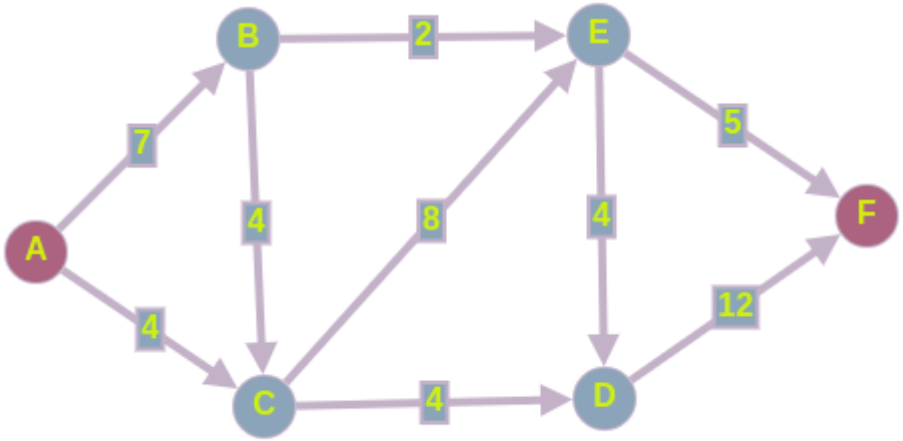


Исходный граф

Остаточная сеть

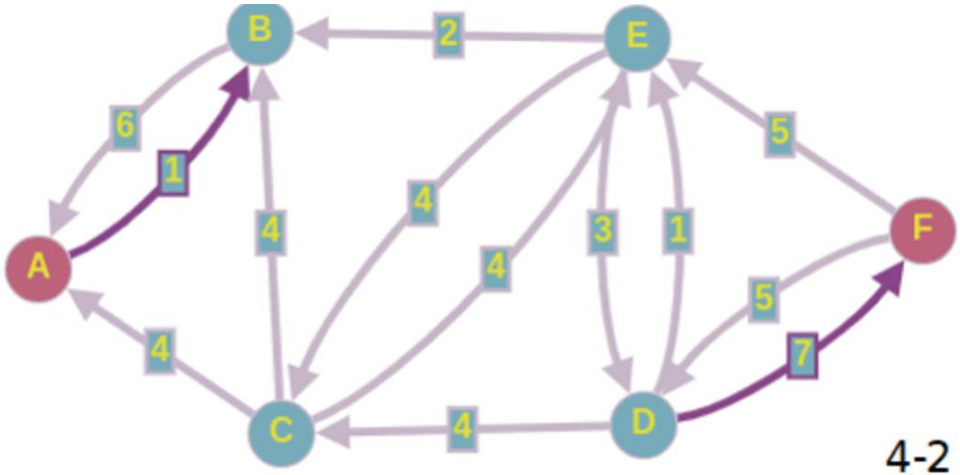


Алгоритм Форда-Фалкерсона



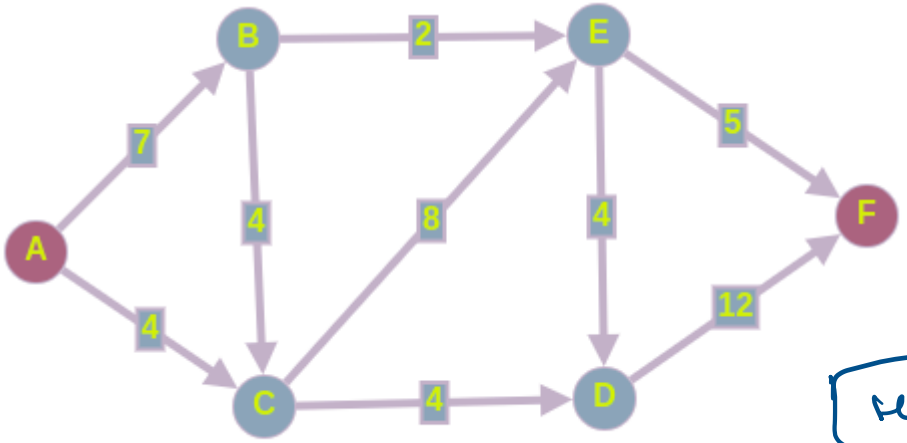
Исходный граф

Остаточная сеть



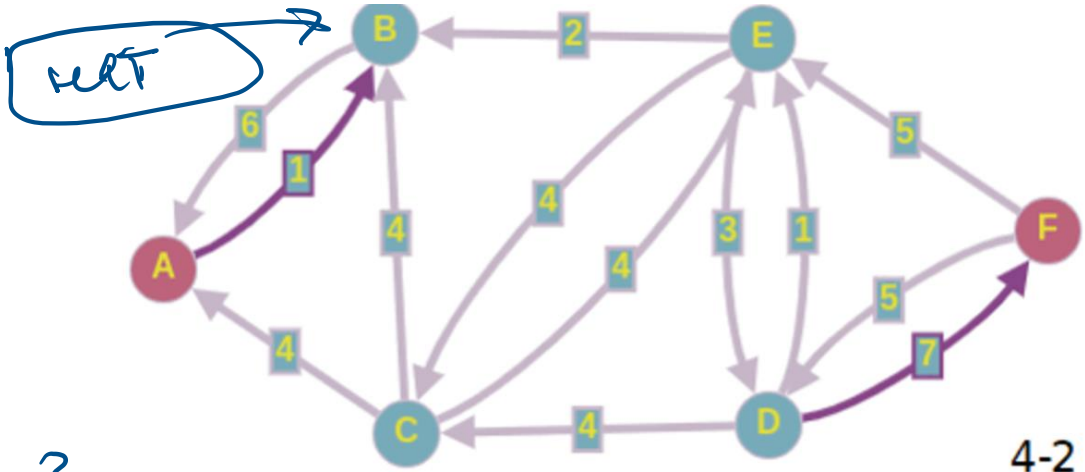
есть ли еще путь?

Алгоритм Форда-Фалкерсона



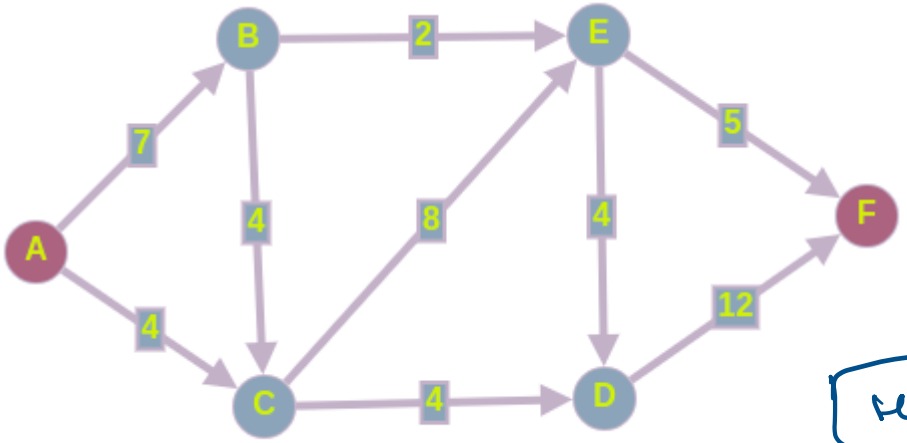
Исходный граф

Остаточная сеть



есть ли еще путь?

Алгоритм Форда-Фалкерсона

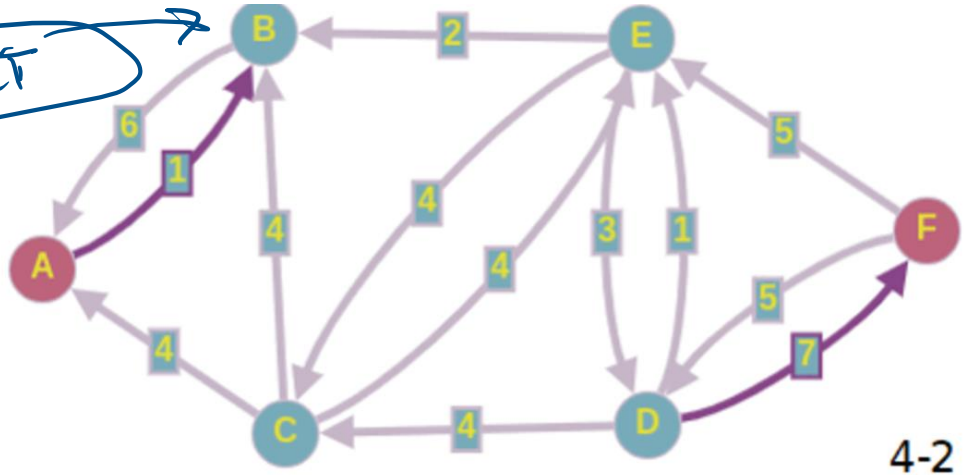


Исходный граф

поток через сеть?

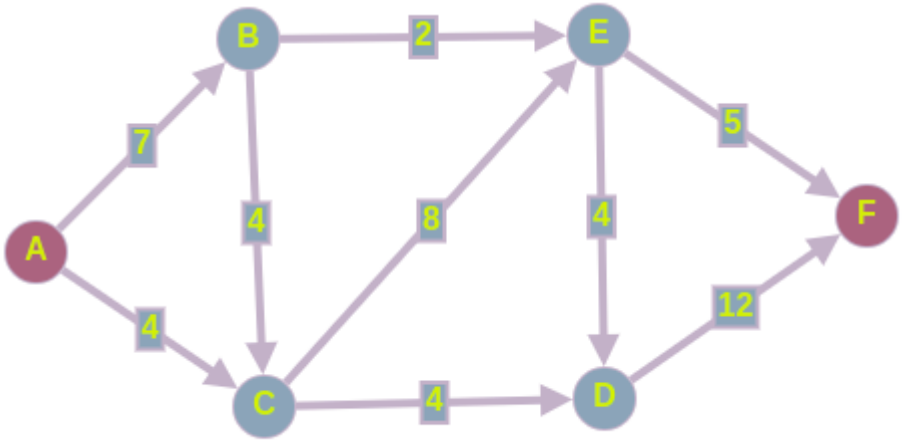
Остаточная сеть

нет



есть ли еще путь?

Алгоритм Форда-Фалкерсона

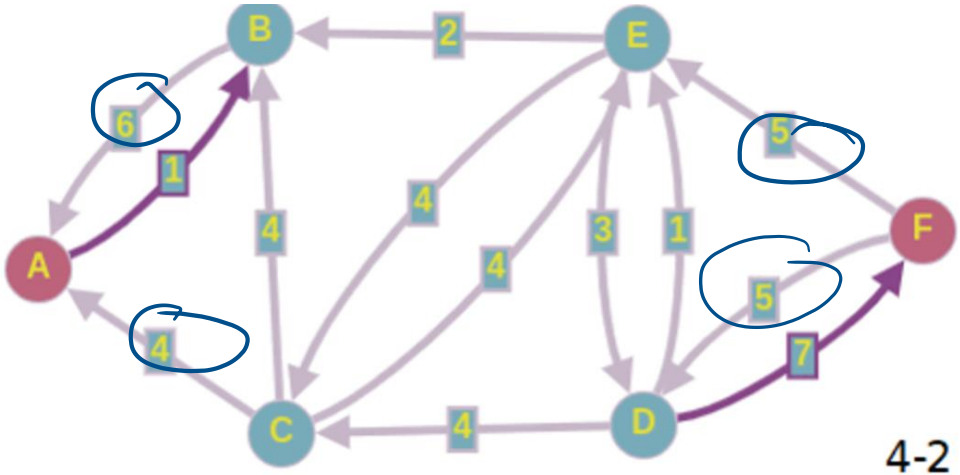


Исходный граф

поток через сеть?

10

Остаточная сеть



6+4=10

5+5=10

Алгоритм Форда-Фалкерсона

1. Обнуляем все потоки. Остаточная сеть изначально совпадает с исходной сетью.
2. В остаточной сети находим любой путь из источника в сток. Если такого пути нет, останавливаемся.
3. Пускаем через найденный путь (***дополняющий путь***) максимально возможный поток:
    1. На найденном пути в остаточной сети ищем ребро с минимальной пропускной способностью ***Cmin***.
    2. Для каждого ребра на найденном пути увеличиваем поток на ***Cmin***, а в противоположном ему — уменьшаем на ***Cmin***.
    3. Модифицируем остаточную сеть. Для всех рёбер на найденном пути, а также для противоположных им рёбер, вычисляем новую пропускную способность. Если она стала ненулевой, добавляем ребро к остаточной сети, а если обнулилась, стираем его.
4. Возвращаемся на шаг 2.

Алгоритм Форда-Фалкерсона

1. Обнуляем все потоки. Остаточная сеть изначально совпадает с исходной сетью.
2. В остаточной сети находим любой путь из источника в сток. Если такого пути нет, останавливаемся.
3. Пускаем через найденный путь (***дополняющий путь***) максимально возможный поток:
    1. На найденном пути в остаточной сети ищем ребро с минимальной пропускной способностью ***Cmin***.
    2. Для каждого ребра на найденном пути увеличиваем поток на ***Cmin***, а в противоположном ему — уменьшаем на ***Cmin***.
    3. Модифицируем остаточную сеть. Для всех рёбер на найденном пути, а также для противоположных им рёбер, вычисляем новую пропускную способность. Если она стала ненулевой, добавляем ребро к остаточной сети, а если обнулилась, стираем его.
4. Возвращаемся на шаг 2.

Асимптотика с целыми пропускными способностями?

Алгоритм Форда-Фалкерсона

1. Обнуляем все потоки. Остаточная сеть изначально совпадает с исходной сетью.
2. В остаточной сети находим любой путь из источника в сток. Если такого пути нет, останавливаемся.
3. Пускаем через найденный путь (***дополняющий путь***) максимально возможный поток:
    1. На найденном пути в остаточной сети ищем ребро с минимальной пропускной способностью ***Cmin***.
    2. Для каждого ребра на найденном пути увеличиваем поток на ***Cmin***, а в противоположном ему — уменьшаем на ***Cmin***.
    3. Модифицируем остаточную сеть. Для всех рёбер на найденном пути, а также для противоположных им рёбер, вычисляем новую пропускную способность. Если она стала ненулевой, добавляем ребро к остаточной сети, а если обнулилась, стираем его.
4. Возвращаемся на шаг 2.

Асимптотика с целыми пропускными способностями?

$O(E \cdot |\max f|)$

Сколько раз запустится

Выполнение одного шага

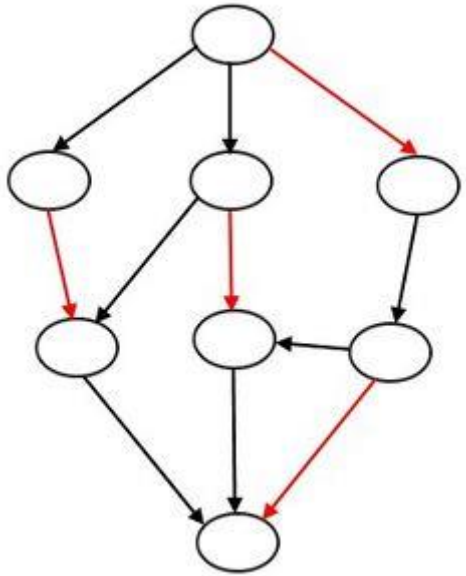
Алгоритм Форда-Фалкерсона

1. Обнуляем все потоки. Остаточная сеть изначально совпадает с исходной сетью.
2. В остаточной сети находим любой путь из источника в сток. Если такого пути нет, останавливаемся.
3. Пускаем через найденный путь (***дополняющий путь***) максимально возможный поток:
    1. На найденном пути в остаточной сети ищем ребро с минимальной пропускной способностью ***Cmin***.
    2. Для каждого ребра на найденном пути увеличиваем поток на ***Cmin***, а в противоположном ему — уменьшаем на ***Cmin***.
    3. Модифицируем остаточную сеть. Для всех рёбер на найденном пути, а также для противоположных им рёбер, вычисляем новую пропускную способность. Если она стала ненулевой, добавляем ребро к остаточной сети, а если обнулилась, стираем его.
4. Возвращаемся на шаг 2.

Асимптотика с целыми пропускными способностями?

Задача о максимальном паросочетании

Паросочетание (англ. matching) M – произвольное множество рёбер графа такое, что никакие два ребра не имеют общей вершины.

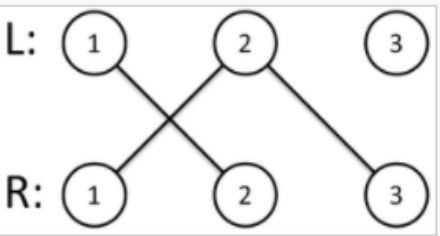


Алгоритм Форда-Факлерсона

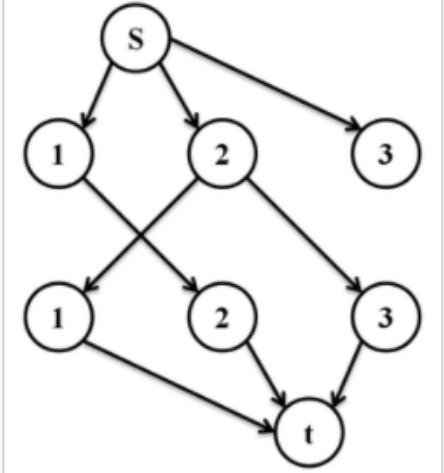
Пусть дан неориентированный двудольный граф $G(V, E)$ и требуется найти максимальное паросочетание в нём. Обозначим доли исходного графа как L и R. Построим граф $G'(V', E')$ следующим образом:

$V' = V \cup \{s, t\}$ (т.е. добавим новый исток s и сток t);

$E' = \{(s, u) : u \in L\} \cup \{(u, v) : u \in L, v \in R\,, (u, v) \in E\} \cup \{(v, t) : v \in R\}.$



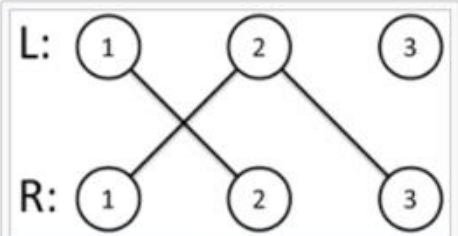
Пример графа G.



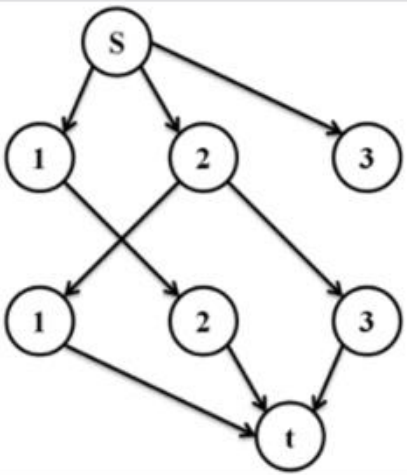
Соответствующий граф G'.

Алгоритм Форда-Факлерсона

1. Ищем в графе G' путь из s в t поиском в глубину.
2. Если путь найден, перезаписываем текущее паросочетание. Далее инвертируем все рёбра на пути (ребро (u, v) становится ребром (v, u)) и удаляем (s, L) и (R, t) ребра, покрывающие вершины, принадлежащие текущему паросочетанию.
3. Если путь не был найден, значит текущее паросочетание является максимальным, и алгоритм завершает работу. Иначе переходим к пункту 1.



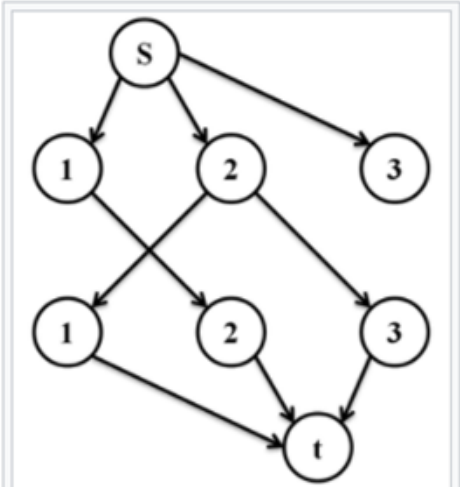
Пример графа G.



Соответствующий граф G'.

Алгоритм Форда-Факлерсона. Разбор примера

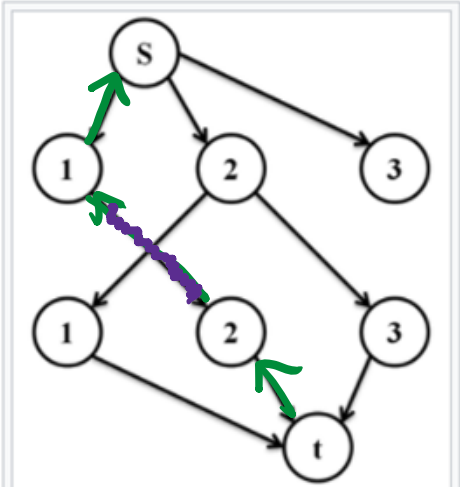
1. Ищем в графе G' путь из s в t поиском в глубину.
2. Если путь найден, перезаписываем текущее паросочетание. Далее инвертируем все рёбра на пути (ребро (u, v) становится ребром (v, u)) и удаляем (s, L) и (R, t) ребра, покрывающие вершины, принадлежащие текущему паросочетанию.
3. Если путь не был найден, значит текущее паросочетание является максимальным, и алгоритм завершает работу. Иначе переходим к пункту 1.



Соответствующий граф G'.

Алгоритм Форда-Факлерсона. Разбор примера

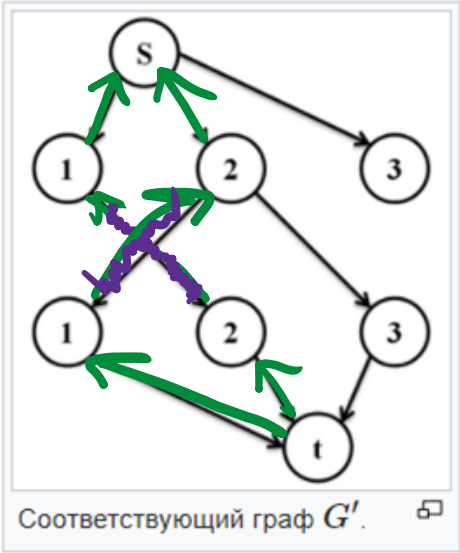
1. Ищем в графе G' путь из s в t поиском в глубину.
2. Если путь найден, перезаписываем текущее паросочетание. Далее инвертируем все рёбра на пути (ребро (u, v) становится ребром (v, u)) и удаляем (s, L) и (R, t) ребра, покрывающие вершины, принадлежащие текущему паросочетанию.
3. Если путь не был найден, значит текущее паросочетание является максимальным, и алгоритм завершает работу. Иначе переходим к пункту 1.



Соответствующий граф G'.

Алгоритм Форда-Факлерсона. Разбор примера

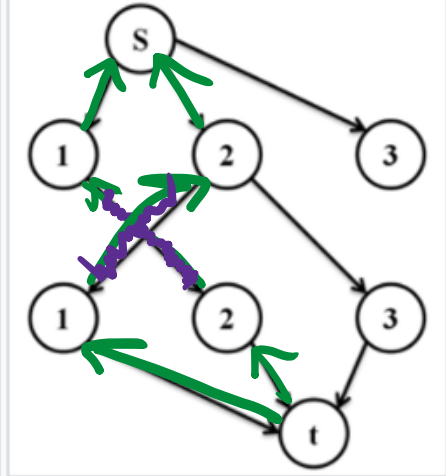
1. Ищем в графе G' путь из s в t поиском в глубину.
2. Если путь найден, перезаписываем текущее паросочетание. Далее инвертируем все рёбра на пути (ребро (u, v) становится ребром (v, u)) и удаляем (s, L) и (R, t) ребра, покрывающие вершины, принадлежащие текущему паросочетанию.
3. Если путь не был найден, значит текущее паросочетание является максимальным, и алгоритм завершает работу. Иначе переходим к пункту 1.



Соответствующий граф G'.

Алгоритм Форда-Факлерсона. Разбор примера

1. Ищем в графе G' путь из s в t поиском в глубину.
2. Если путь найден, перезаписываем текущее паросочетание. Далее инвертируем все рёбра на пути (ребро (u, v) становится ребром (v, u)) и удаляем (s, L) и (R, t) ребра, покрывающие вершины, принадлежащие текущему паросочетанию.
3. Если путь не был найден, значит текущее паросочетание является максимальным, и алгоритм завершает работу. Иначе переходим к пункту 1.

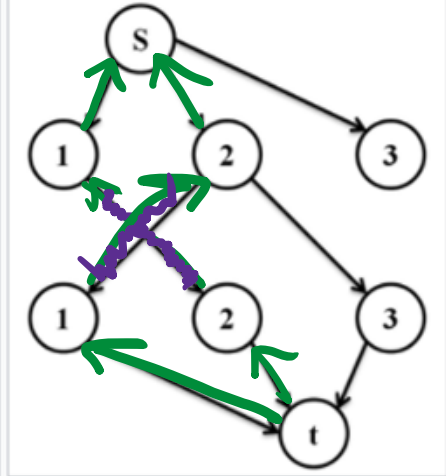


Соответствующий граф G'.

есть ли еще путь?

Алгоритм Форда-Факлерсона. Разбор примера

1. Ищем в графе G' путь из s в t поиском в глубину.
2. Если путь найден, перезаписываем текущее паросочетание. Далее инвертируем все рёбра на пути (ребро (u, v) становится ребром (v, u)) и удаляем (s, L) и (R, t) ребра, покрывающие вершины, принадлежащие текущему паросочетанию.
3. Если путь не был найден, значит текущее паросочетание является максимальным, и алгоритм завершает работу. Иначе переходим к пункту 1.



Соответствующий граф G'.

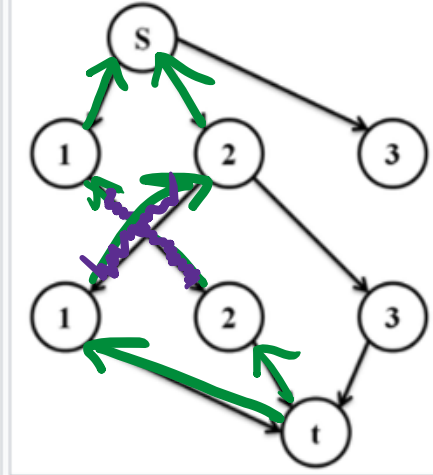
есть ли еще путь?

нет :с

Алгоритм Форда-Факлерсона. Разбор примера

1. Ищем в графе G' путь из s в t поиском в глубину.
2. Если путь найден, перезаписываем текущее паросочетание. Далее инвертируем все рёбра на пути (ребро (u, v) становится ребром (v, u)) и удаляем (s, L) и (R, t) ребра, покрывающие вершины, принадлежащие текущему паросочетанию.
3. Если путь не был найден, значит текущее паросочетание является максимальным, и алгоритм завершает работу. Иначе переходим к пункту 1.

Тогда
max
паросоч
2



Соответствующий граф G'.

Есть
ли
еще
путь?

Нет :С

Алгоритм Форда-Факлерсона. Псевдокод

- $px[]$ — массив вершин $y \in R$, инцидентные $x_i \in L$ в текущем паросочетании,
- $py[]$ — массив вершин $x \in L$, инцидентные $y_i \in R$ в текущем паросочетании,
- $vis[]$ — массив, где помечаются посещенные вершины.

Максимальное паросочетание — такие ребра (x, y), что $x \in L, y \in R, px[x] = y$.

Инициализация и внешний цикл:

```
func fordFulkerson():
    fill(px, -1)
    fill(py, -1)
    isPath = true
    while isPath
        isPath = false
        fill(vis, false)
        for x ∈ L
            if px[x] == -1
                if dfs(x)
                    isPath = true
```

проверка наличия пути

Поиск в глубину, одновременно инвертирующий ребра:

```
bool dfs(x):
    if vis[x]
        return false
    vis[x] = true
    for (x, y) ∈ E
        if py[y] == -1
            py[y] = x
            px[x] = y
            return true
        else
            if dfs(py[y])
                py[y] = x
                px[x] = y
                return true
    return false
```

разворот

Алгоритм Форда-Факлерсона. Псевдокод

- $px[]$ — массив вершин $y \in R$, инцидентные $x_i \in L$ в текущем паросочетании,
- $py[]$ — массив вершин $x \in L$, инцидентные $y_i \in R$ в текущем паросочетании,
- $vis[]$ — массив, где помечаются посещенные вершины.

Максимальное паросочетание — такие ребра (x, y), что $x \in L, y \in R, px[x] = y$.

Инициализация и внешний цикл:

```
func fordFulkerson():
    fill(px, -1)
    fill(py, -1)
    isPath = true
    while isPath
        isPath = false
        fill(vis, false)
        for x ∈ L
            if px[x] == -1
                if dfs(x)
                    isPath = true
```

Поиск в глубину, одновременно инвертирующий ребра:

```
bool dfs(x):
    if vis[x]
        return false
    vis[x] = true
    for (x, y) ∈ E
        if py[y] == -1
            py[y] = x
            px[x] = y
            return true
        else
            if dfs(py[y])
                py[y] = x
                px[x] = y
                return true
    return false
```

Асимптотика?

Алгоритм Форда-Факлерсона. Псевдокод

- $px[]$ — массив вершин $y \in R$, инцидентные $x_i \in L$ в текущем паросочетании,
- $py[]$ — массив вершин $x \in L$, инцидентные $y_i \in R$ в текущем паросочетании,
- $vis[]$ — массив, где помечаются посещенные вершины.

Максимальное паросочетание — такие ребра (x, y), что $x \in L, y \in R, px[x] = y$.

Инициализация и внешний цикл:

```
func fordFulkerson():
    fill(px, -1)
    fill(py, -1)
    isPath = true
    while isPath
        isPath = false
        fill(vis, false)
        for x ∈ L
            if px[x] == -1
                if dfs(x)
                    isPath = true
```

Поиск в глубину, одновременно инвертирующий ребра:

```
bool dfs(x):
    if vis[x]
        return false
    vis[x] = true
    for (x, y) ∈ E
        if py[y] == -1
            py[y] = x
            px[x] = y
            return true
        else
            if dfs(py[y])
                py[y] = x
                px[x] = y
                return true
    return false
```

Асимптотика?

Алгоритм Форда-Факлерсона. Псевдокод

- $px[]$ — массив вершин $y \in R$, инцидентные $x_i \in L$ в текущем паросочетании,
- $py[]$ — массив вершин $x \in L$, инцидентные $y_i \in R$ в текущем паросочетании,
- $vis[]$ — массив, где помечаются посещенные вершины.

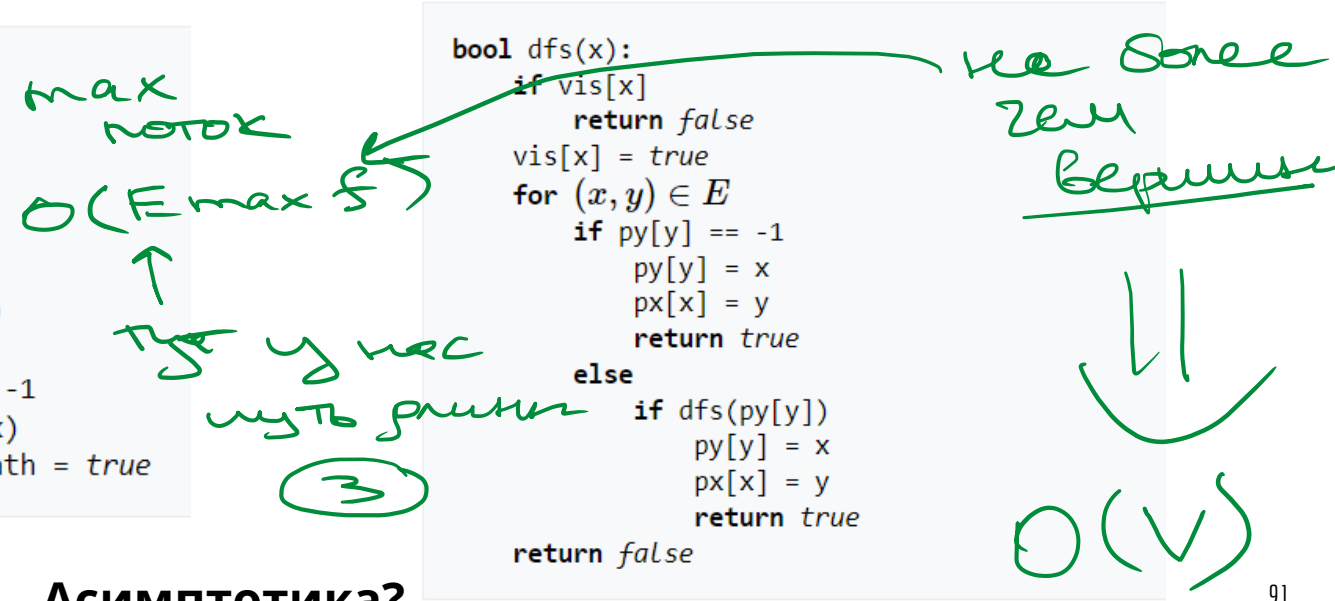
Максимальное паросочетание — такие ребра (x, y), что $x \in L, y \in R, px[x] = y$.

Инициализация и внешний цикл:

```
func fordFulkerson():
    fill(px, -1)
    fill(py, -1)
    isPath = true
    while isPath
        isPath = false
        fill(vis, false)
        for x ∈ L
            if px[x] == -1
                if dfs(x)
                    isPath = true
```

Поиск в глубину, одновременно инвертирующий ребра:

```
bool dfs(x):
    if vis[x]
        return false
    vis[x] = true
    for (x, y) ∈ E
        if py[y] == -1
            py[y] = x
            px[x] = y
            return true
        else
            if dfs(py[y])
                py[y] = x
                px[x] = y
                return true
    return false
```



Асимптотика?