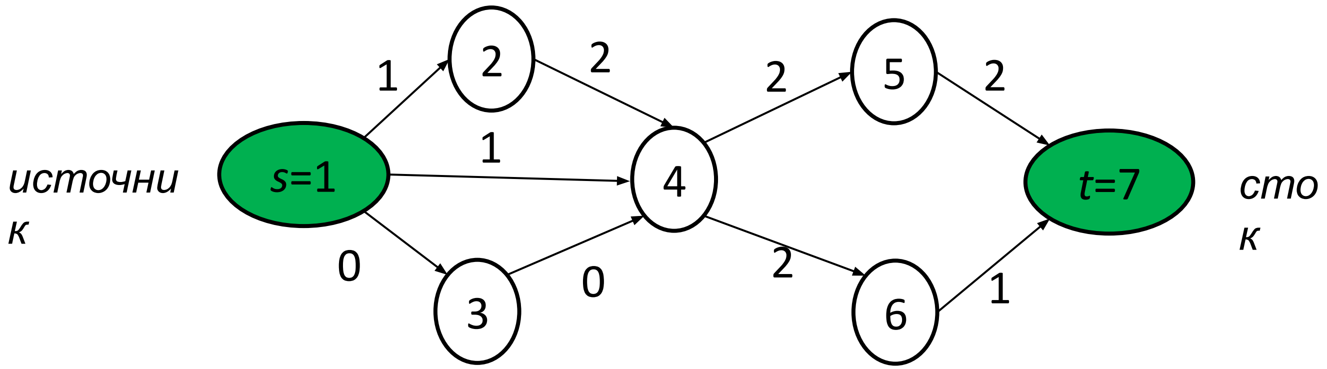
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Максимальный поток в сети и его приложения

https://github.com/larandaA/alg-ds-snippets

Поток в сети

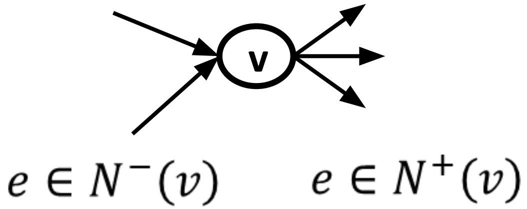
$f: \mathrm{E} \to \mathrm{R}$



<u>Определение</u>

Дивергенция функции *f* в вершине *v* (от лат. *divergere* — расхождение) определяется как разность сумм её значений на выходящих и входящих дугах:

$$\boxed{div_f(v) = \sum_{e \in N^+(v)} f(e) - \sum_{e \in N^-(v)} f(e)}$$



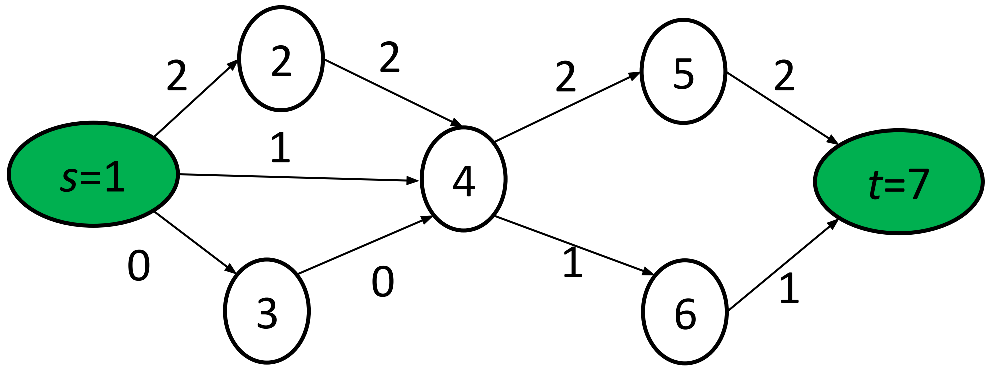
Г.М. Адельсон-Вельский, Е.А. Диниц, А.В. Карзанов. ***Потоковые алгоритмы.*** М.,Наука, 1975 г.

Дивергенция: $div_f(v) = \sum\limits_{e \in N^+(v)} f(e) - \sum\limits_{e \in N^-(v)} f(e)$

Определение

Потоком в сети D называют функцию $f: \mathrm{E} \to \mathrm{R}$ дивергенция которой на внутренних вершинах сети равна 0 .

Внутренние вершины сети – это вершины, отличные от источника и стока.



Пусть f – поток.

$$\sum_{v \in V} div_f(v) = div_f(s) + \sum_{w \in V \setminus \{s,t\}} div_f(w) + div_f(t) = \mathbf{0}$$

$$div_f(s) + div_f(t) = 0$$
$$div_f(s) = -div_f(t)$$

$$M(f) = div_f(s) = -div_f(t)$$

величина потока f

Зададим на дугах сети *D* для потока *f* ограничения:

$$d(e) \le f(e) \le c(e).$$

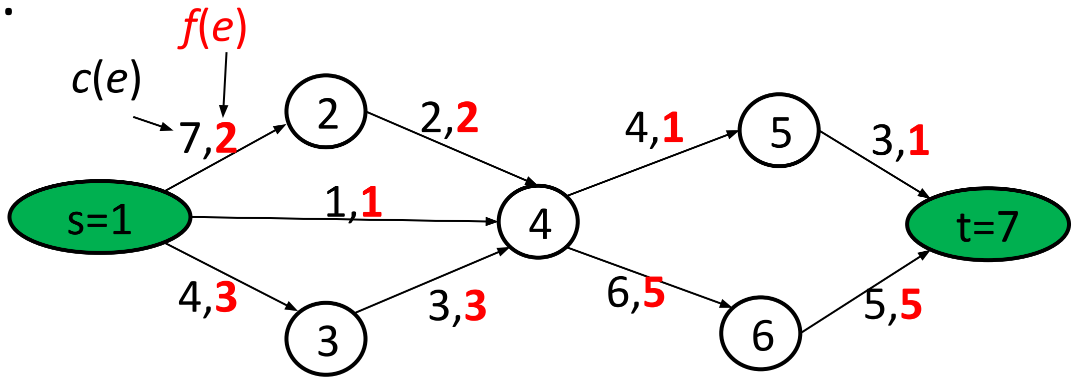
в классической задаче нижнее ограничение d(e)=0

верхнее ограничение называют пропускной способностью дуги

Классическая задача о максимальном потоке:

для двухполюсной сети *D* требуется найти поток ***f*** максимальной величины, удовлетворяющий ограничениям: $0 \le f(e) \le c(e), \forall e \in E.$

Поток, величина которого максимальна, называется **максимальным потоком**.

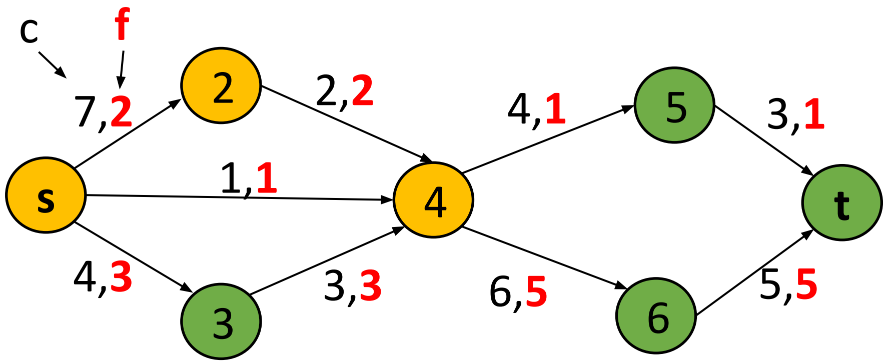


$$M(f) = 6$$

Замечания

1. В дальнейшем мы будем работать с **целочисленными потоками**, то есть все ограничения – целые числа.
2. Считаем, что любая внутренняя вершина сети лежит на некотором (s,t)-пути ($m \geq n - 1$).
3. Предполагаем, что в сети **нет кратных дуг**.

Максимальный поток и минимальный разрез



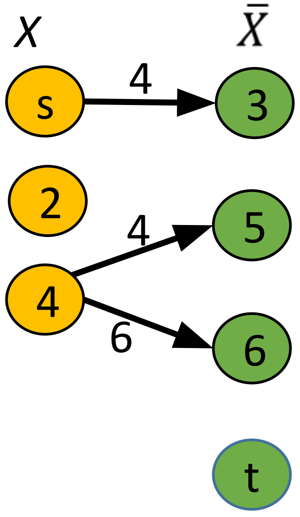
$R = (X, \overline{X}),\ X \subseteq V, \overline{X} = V \setminus X,\ s \in X,\ t \in \overline{X}$

$E^+(R)$ – дуги, которые идут из X в $\overline{X}$

$E^-(R)$ – дуги, которые идут из $\overline{X}$ в X

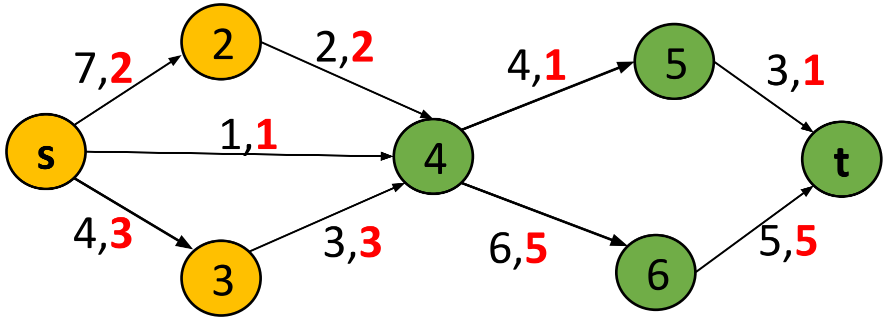
остальные дуги сети – внутренние дуги разреза

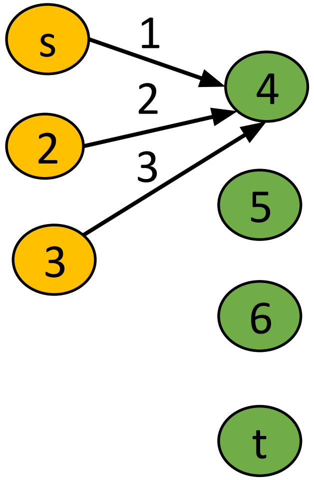
$$c(R) = \sum_{e \in E^+(R)} c(e) \text{ – пропускная способность разреза}$$



$c(R) = 14$

Максимальный поток и минимальный разрез





$c(R) = 6$

$M(f) = 6$

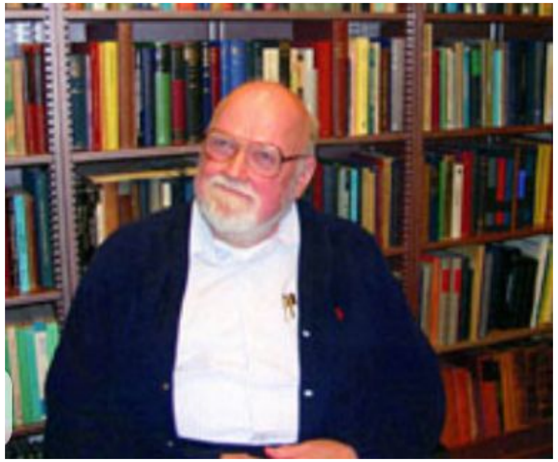
Разрез, пропускная способность которого минимальна, называется **минимальным разрезом.**

Утверждение

Для сети G **величина любого потока** f **не превосходит пропускной способности любого разреза** R**:** $M(f) \leq c(R)$.

Значит, величина максимального потока не превосходит пропускной способности минимального разреза $M(f^{max}) \leq c(R^{min})$. Поэтому, если для некоторого потока f справедливо, что $c(R) = M(f)$, то это будет означать, что f – максимальный поток.

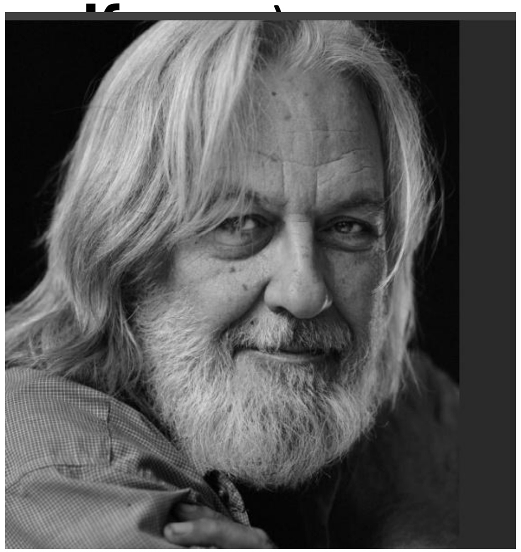
1955 год (**метод Форда-Фалкерсона**)



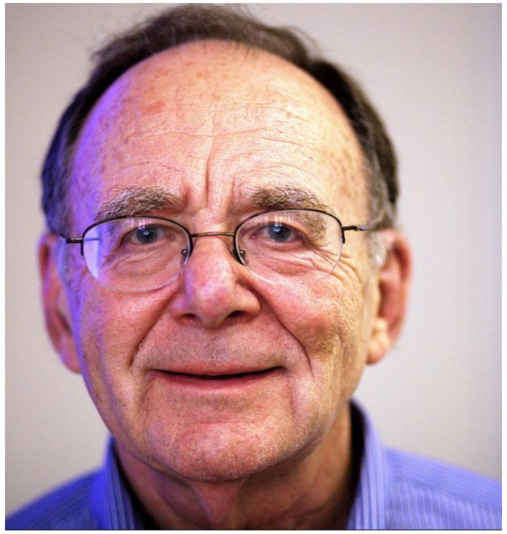
Лестер Рэндольф Форд младший
англ. *Lester Randolph Ford, Jr.*
1927 – 2017
США
Научная сфера - математик

Делберт Рей Фалкерсон
англ. *Delbert Ray Fulkerson*
1924 – 1976
США
Научная сфера - комбинаторика

1972 год (**алгоритм Эдмондса-**

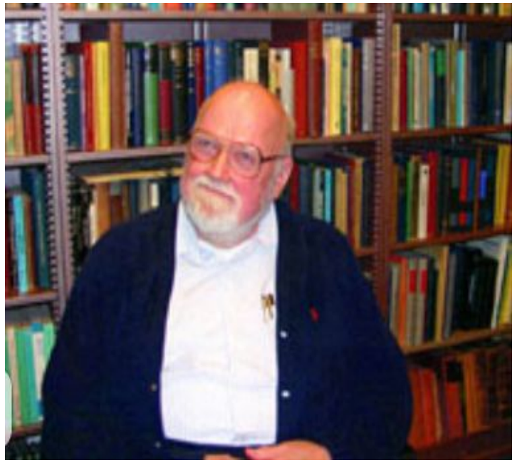


Джек Р. Эдмондс
англ. *Jack Edmonds*
1934
США
Научная сфера - комбинаторная оптимизация



Ричард Мэннинг Карп
англ. *Richard Manning Karp*
1935
США
Научная сфера – теория алгоритмов и биоинформатика

Метод **Форда-Фалкерсона**



Лестер Рэндольф Форд младший

Делберт Рей Фалкерсон

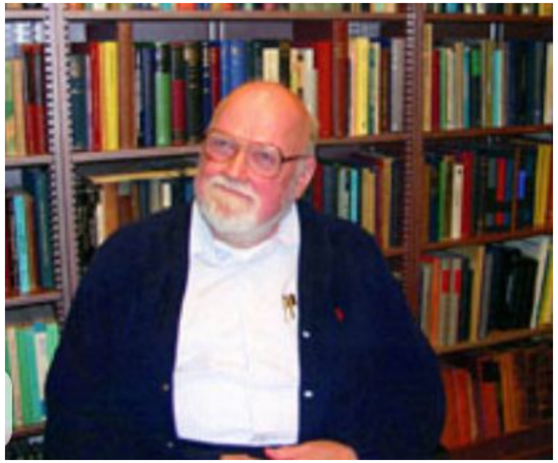
Теорема Форда – Фалкерсона

Пусть f – некоторый поток в сети D, тогда следующие утверждения эквивалентны:

(1) f – максимальный поток;

(2) для потока f в сети остаточных пропускных способностей D_f нет увеличивающего (s, t)-пути

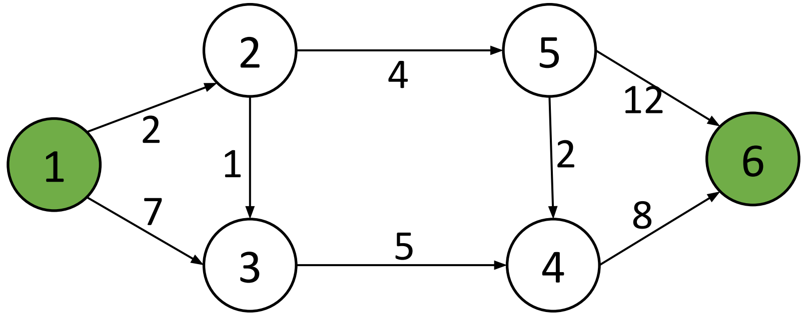
(3) $M(f) = c(R)$ для некоторого разреза R сети.



Лестер Рэндольф Форд младший
англ. *Lester Randolph Ford, Jr.*

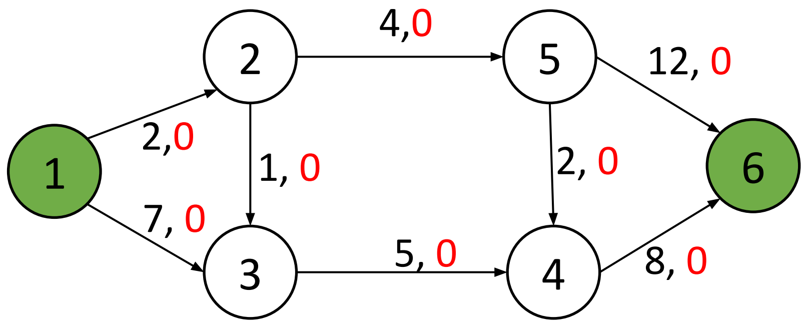
Делберт Рей Фалкерсон
англ. *Delbert Ray Fulkerson*

Найти максимальный поток в сети методом Форда-Фалкерсона

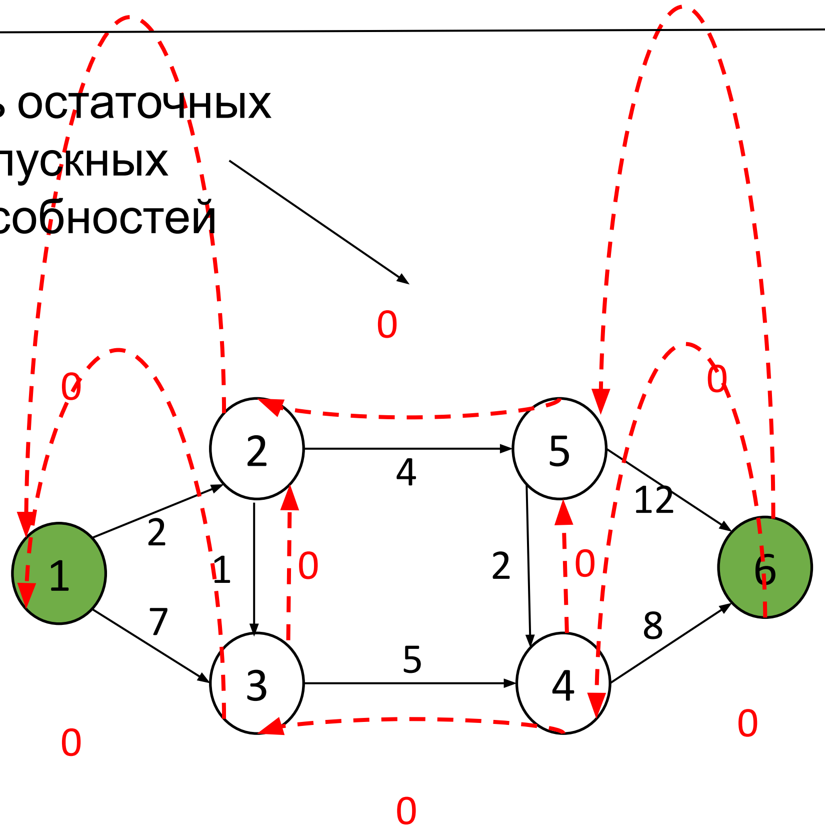


1-я итерация

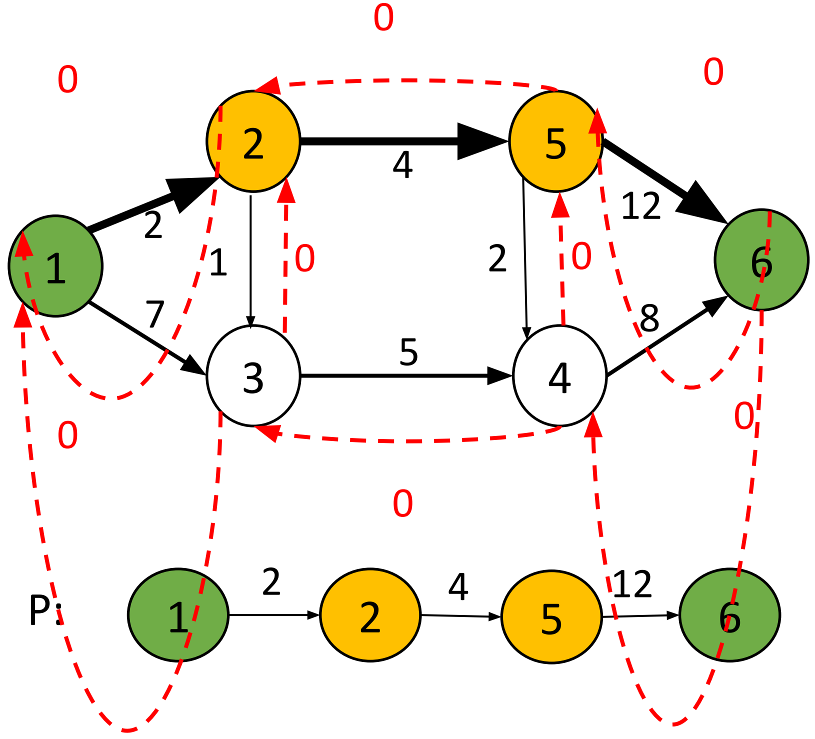
$M(f) = 0$

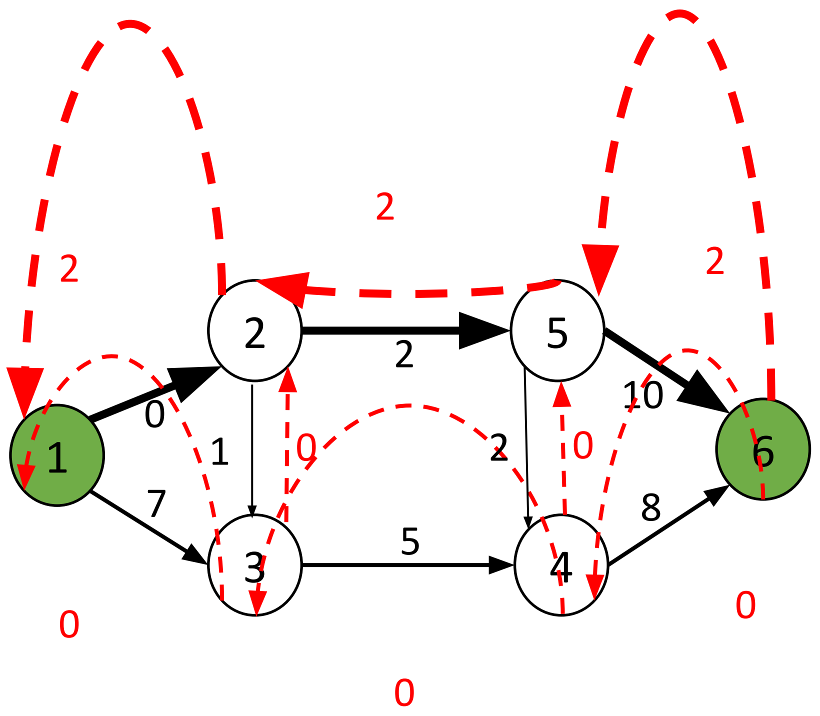


сеть остаточных пропускных способностей



1-я итерация
(продолжение)



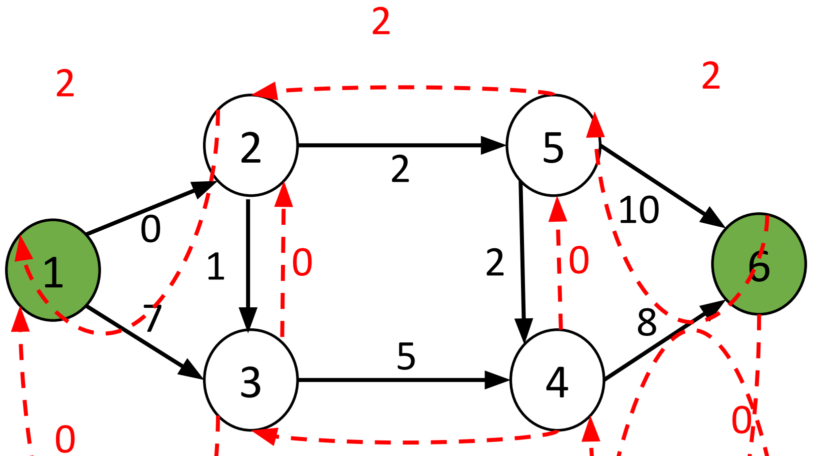


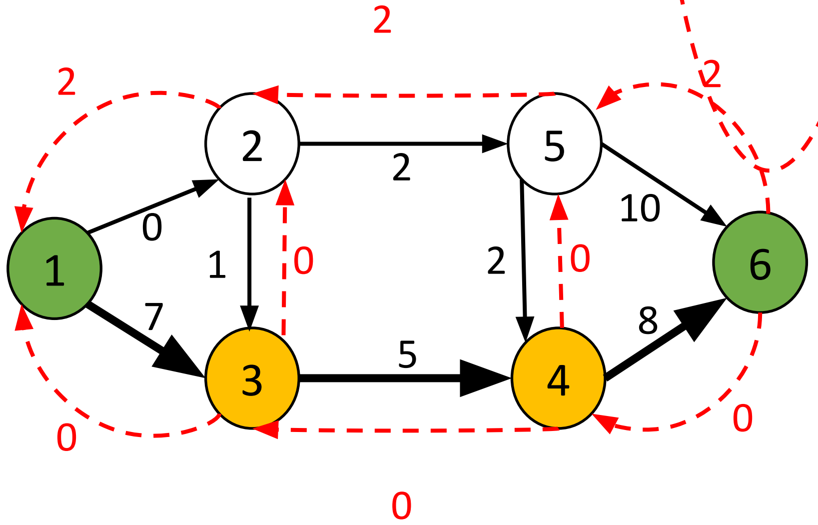
$c_f^{\min}$ =min {c′(1,2), c′(2,5), c′(5,6)}= min { 2, 4, 12}=2

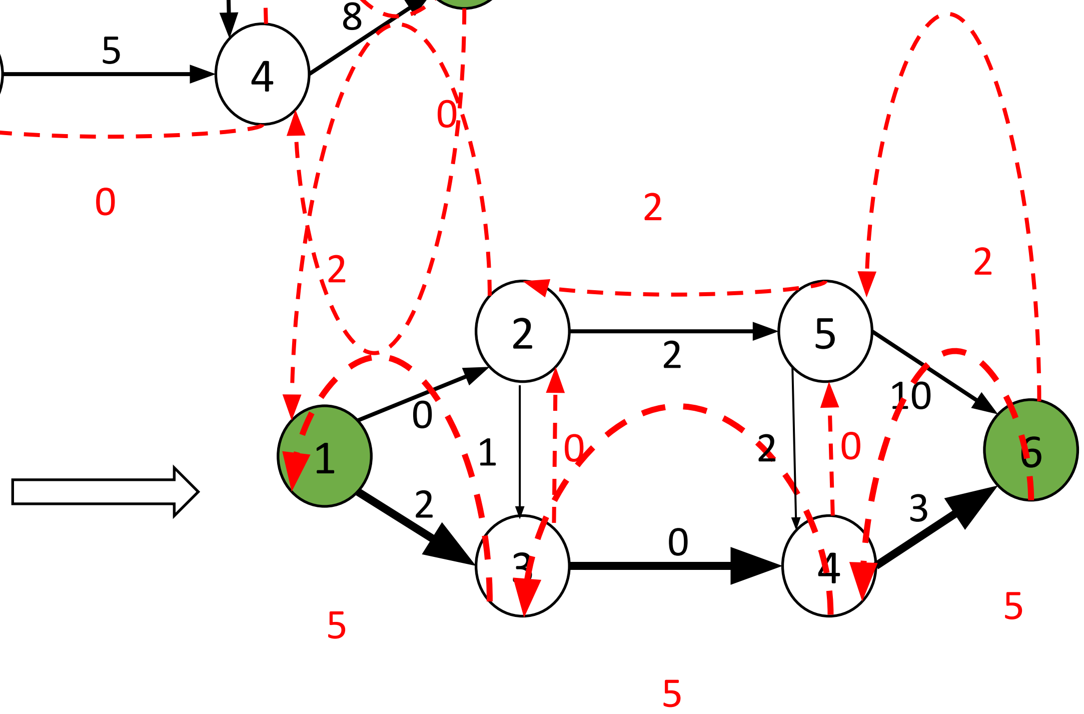
$M(f) = M(f) + c_f^{\min}$=0+2=2

2-я итерация

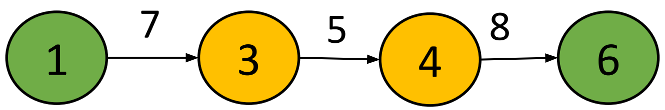
$M(f) = 2$







P:

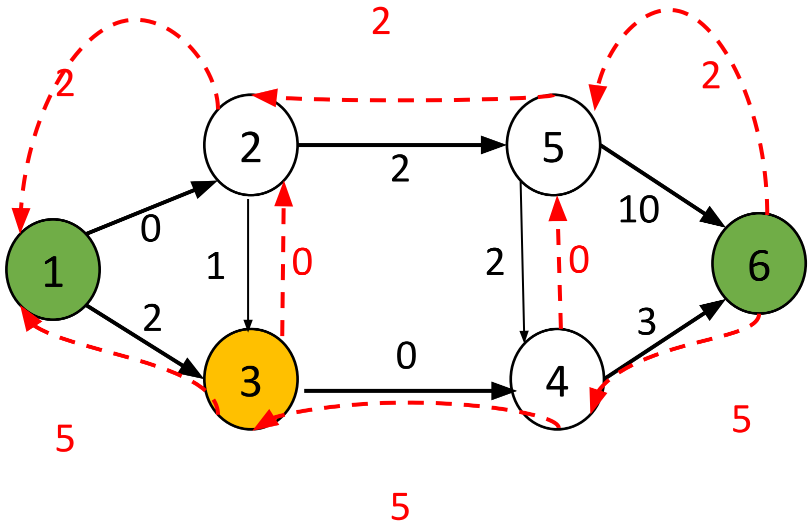


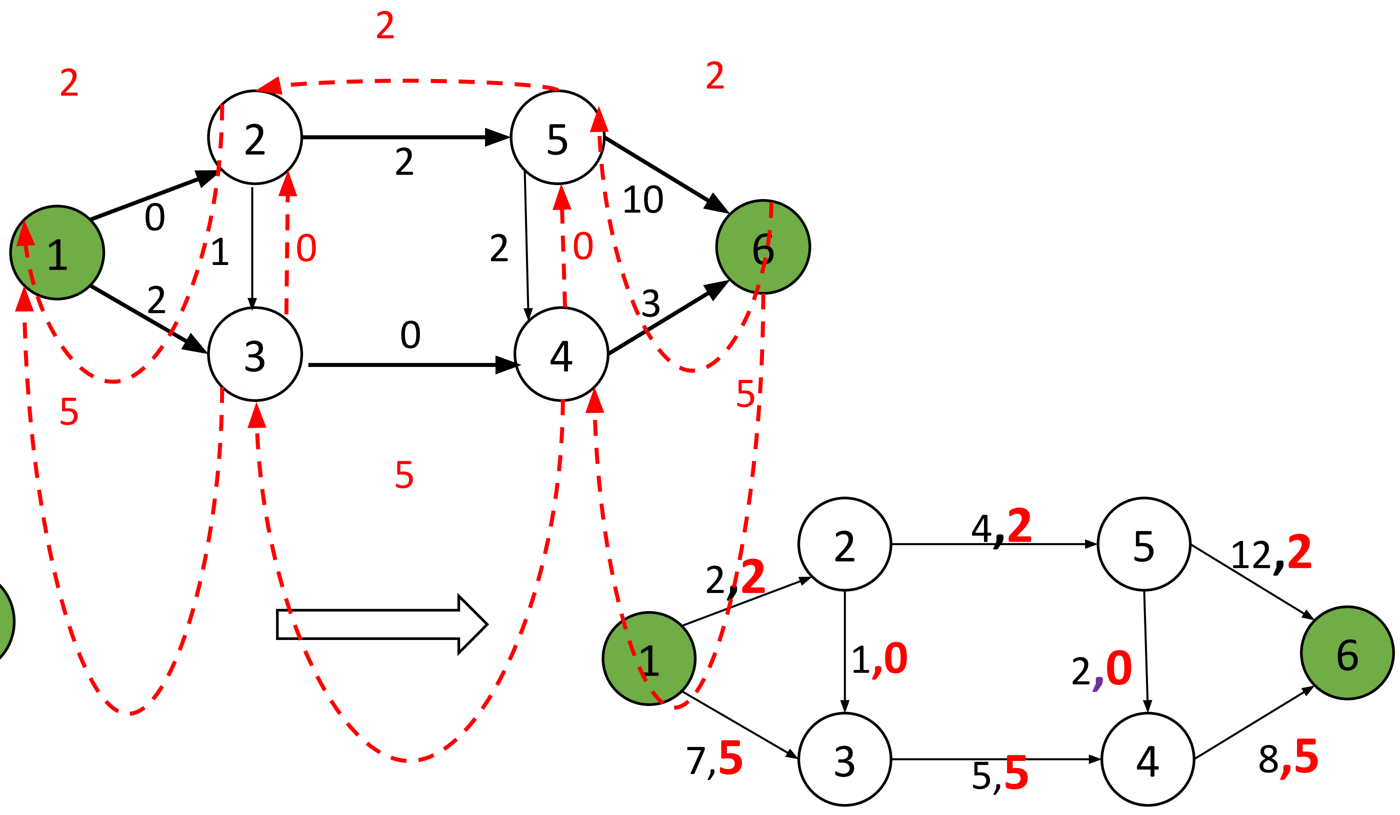
$c_f^{\min}$ =min {c′(1,3), c′(3,4), c′(4,6)}= min { 7, 5, 8}=5

$M(f) = M(f) + c_f^{\min}$=2+5=7

3-я
итерация

$M(f) = 7$



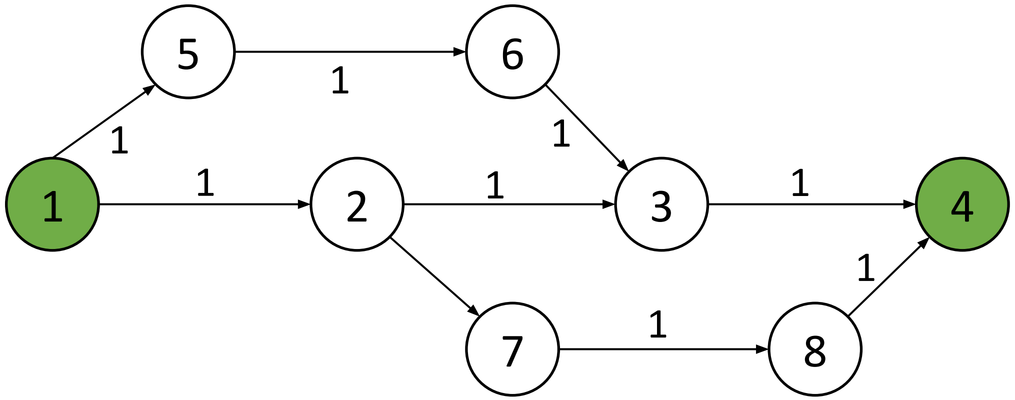


Если для текущего потока *f* в сети остаточных пропускных способностей D_f не существует увеличивающего (s,t)-пути, то величина потока *f* равна пропускной способности разреза (<u>в множество *X* берём вершину *s* и те вершины, до которых удалось дойти из вершины *s* на последней итерации метода Форда-Фалкерсона</u>).

По теореме Форда-Фалкерсона текущий поток f - максимальный.

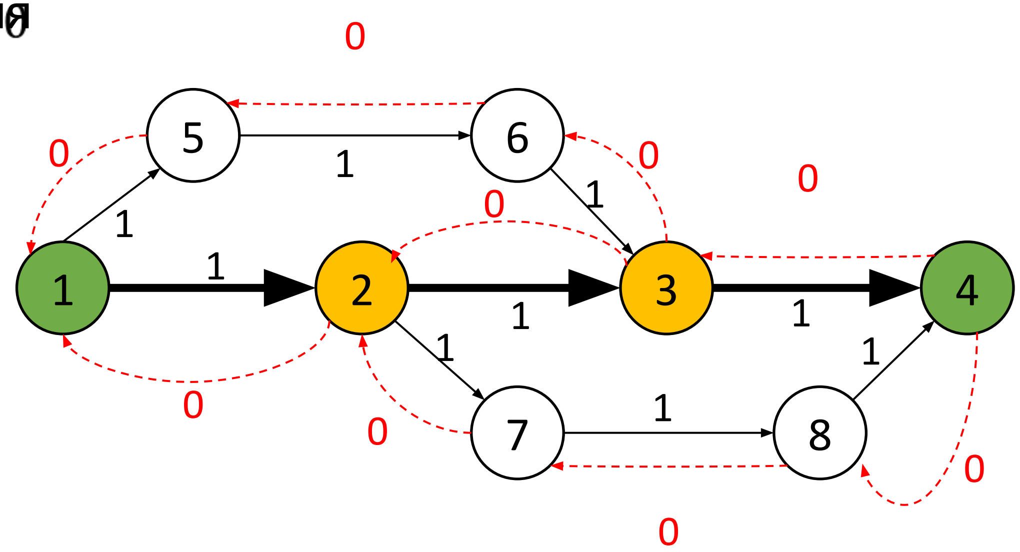
См. доказательство: «Сборник задач по теории алгоритмов : учеб.-метод. пособие» / В. М. Котов [и др.]. – Минск : БГУ, 2017. С. 26-30.

Пример 2



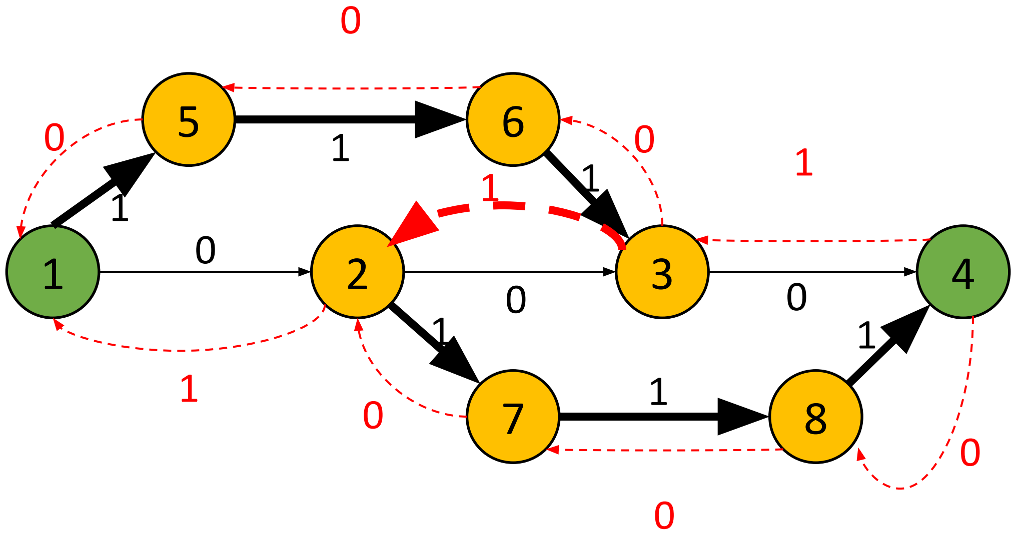
1-я итерация

$M(f) = 0$



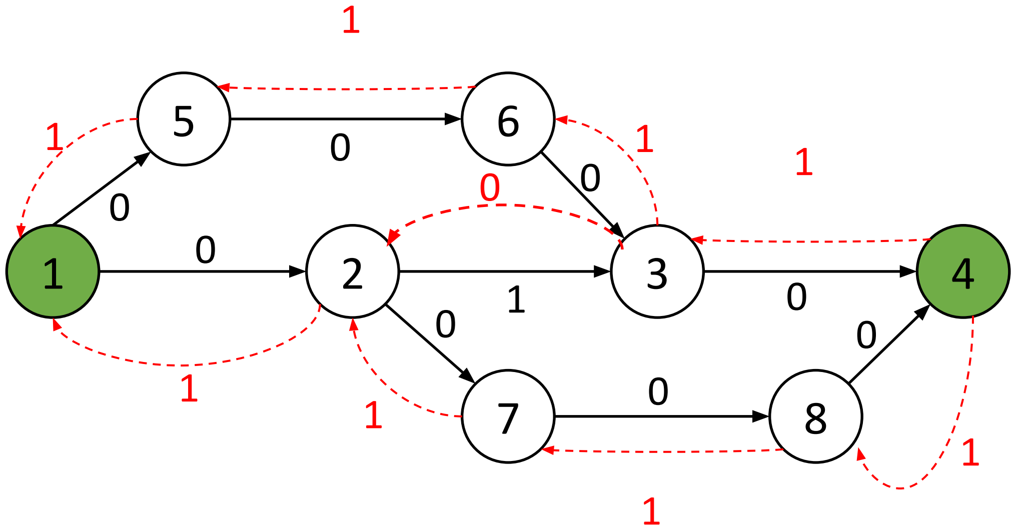
2-я итерация

$M(f) = 1$

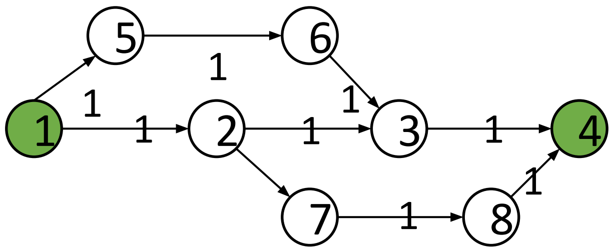


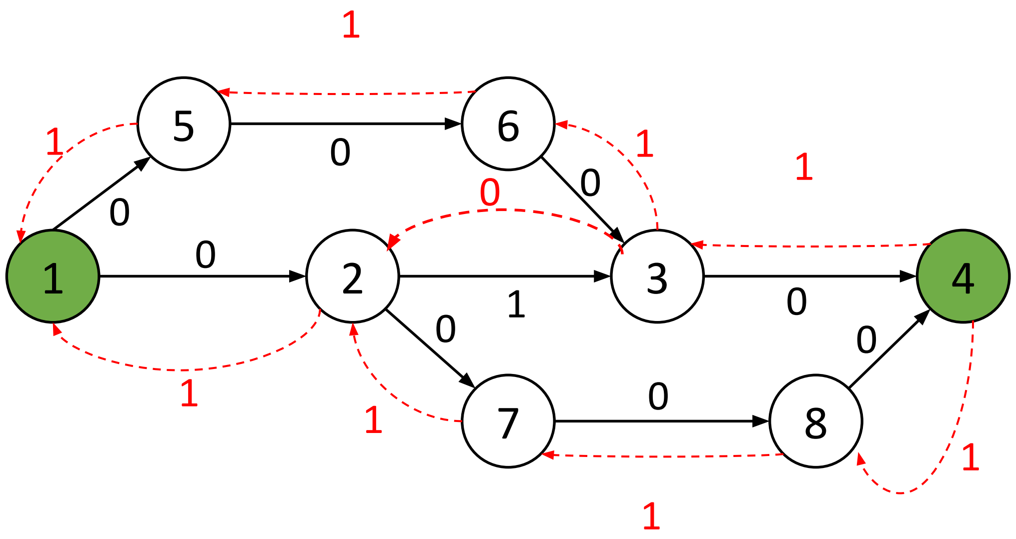
3-я итерация

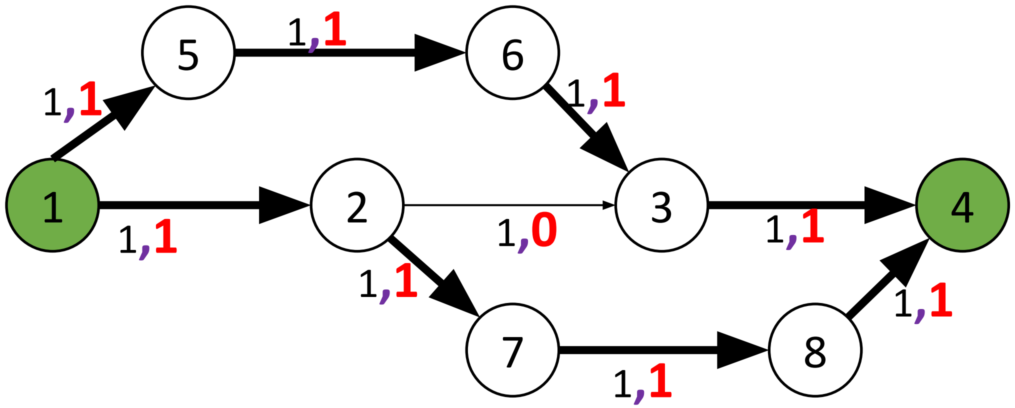
$M(f) = 2$



4-я
итерация







$M(f) = 2$ – максимальный поток в сети

Метод Форда – Фалкерсона

Работает для сетей с целочисленными пропускными способностями дуг.

Время работы: O($M(f^{max})\cdot ?$ **)**,

где $M(f^{max})$ – величина максимального потока

так как поток целочисленный, а в исходной сети (по сделанному ранее предположению) нет кратных дуг, то $M(f^{max}) \le \mathbf{c^{max}\cdot n}$,
где $\mathbf{c^{max}}$ - наибольшая из пропускных стоимость дуг сети.

? – время поиска увеличивающего пути

для поиска увеличивающего пути воспользуемся поиском в глубину (DFS) - **O(n+m)**.

Поэтому, если на итерациях метода Форда-Фалкерсона используется поиск в глубину, то можно выписать следующую оценку

O($c^{max}\cdot n\cdot m$) - псевдополиномиальный алгоритм

Алгоритми Эдмондса – Карпа
полиномиальный алгоритм

Время работы: $O(n \cdot m \cdot (n+m)) = O(n \cdot m^2)$,

$O(n+m)$ – время работы поиска в ширину;
$O(n \cdot m)$ – число итераций алгоритма;

- Поиск увеличивающего пути: поиск в ширину (**BFS**).
- После каждой итерации алгоритма длина **dist(s,v)** (в дугах) наименьшего пути из источника **s** в вершину **v** монотонно не убывает. Так как длина (s,t)-пути не превосходит (n-1), то конечная вершина t может изменять свою метку dist(s,t) не более, чем n раз.
- Назовем **k-этапом** совокупность итераций, на которых длина наименьшего пути сохраняется равной **k**. Эти итерации идут подряд. На k-этапе после каждой итерации алгоритма из новой сети вычёркивается хотя бы одна дуга, построенного на этой итерации увеличивающего пути и она не может появиться вновь, так как она не является обратной дугам следующих увеличивающих путей **k**-этапа. Поэтому число итераций алгоритма на **k**-этапе не превосходит **m**.
- Получаем оценку на число итераций **O(n·m)**.

Метод Форда – Фалкерсона

псевдополиномиальный алгоритм

работает для сетей с целочисленными пропускными способностями дуг

Алгоритми Эдмондса – Карпа

полиномиальный алгоритм

$O(c^{max} \cdot n \cdot m)$

$O(n \cdot m^2)$

Представление сети остаточных пропускных способностей на списках смежности

списки смежности для исходной сети

g

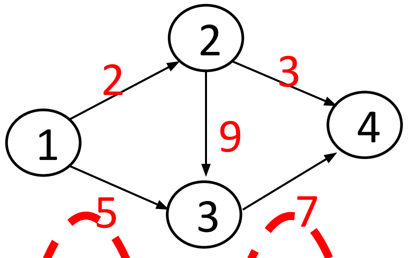
u c_{uv} v

1: [(2,2) , (5,3)]

2: [(9,3) , (3,4)]

3: [(7,4)]

4: [ ]



списки дуг

flow_edges

u c_{uv} f_{uv} v

0: (1,2,0, 2)

1: (2,0,0, 1)

2: (1,5,0, 3)

3: (3,0,0, 1)

4: (2,9,0, 3)

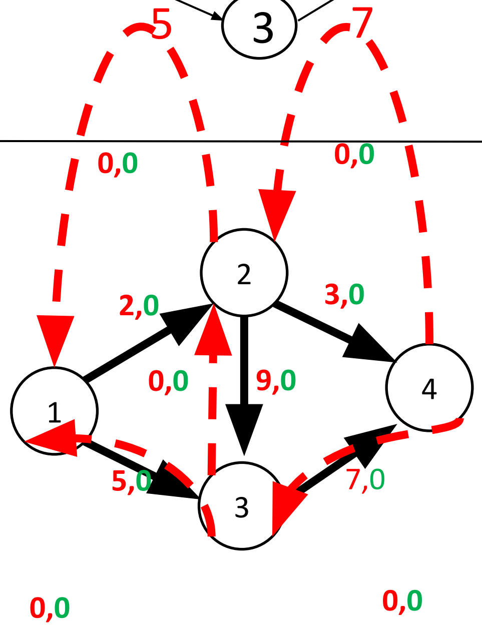
5: (3,0,0, 2)

6: (2,3,0, 4)

7: (4,0,0, 2)

8: (3,7,0, 4)

9: (4,0,0, 3)



остаточная пропускная способность:

$c'_{uv} = f_{uv} - c_{uv}$

списки смежности для остаточной сети

network

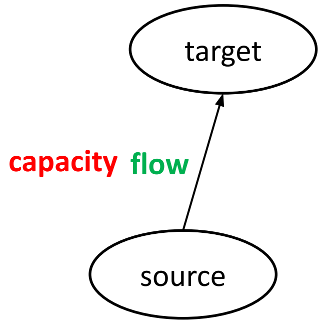
1: [0, 2]

2: [1, 4, 6]

3: [3, 5, 8]

u 4: [7, 9]

```
@dataclass
class Edge:
    source: int
    target: int
    capacity: int
    flow: int
```




```
network = [[] for v in range(n)]
flow_edges = []

def build_network():
    for v in range(n):
        for cu, u in g[v]:
            network[v].append(len(flow_edges))
            flow_edges.append(Edge(source=v, target=u, capacity=cu, flow=0))
            network[u].append(len(flow_edges))
            flow_edges.append(Edge(source=u, target=v, capacity=0, flow=0))
```

Псевдокод функций для работы с сетью

- `build_network` для построения остаточной сети на основе исходного графа
- `source` для получения начальной вершины ребра в остаточной сети
- `target` для получения конечной вершины ребра в остаточной сети
- `available` для получения остаточной пропускной способности ребра
- `flow` для получения величины потока, пропущенного по ребру
- `edges` для получения исходящих из вершины ребер в остаточной сети
- `push` для увеличения потока вдоль ребра в остаточной сети

```
network = [[] for v in range(n)]
flow_edges = []

def build_network():
    for v in range(n):
        for cu, u in g[v]:
            network[v].append(len(flow_edges))
            flow_edges.append(Edge(source=v, target=u, capacity=cu
            network[u].append(len(flow_edges))
            flow_edges.append(Edge(source=u, target=v, capacity=0,
```

https://github.com/larandaA/alg-ds-snippets

```
def edges(v):
    return network[v]

def available(e):
    edge = flow_edges[e]
    return edge.capacity - edge.flow

def flow(e):
    edge = flow_edges[e]
    return edge.flow

def target(e):
    edge = flow_edges[e]
    return edge.target

def source(e):
    edge = flow_edges[e]
    return edge.source
```

```
def push(e, flow):
    edge = flow_edge[e]
    edge.flow += flow

    edge = flow_edge[e ^ 1]
    edge.flow -= flow
```

Общая схема метода Форда – Фалкерсона

```
def ford_fulkerson(s, t):
    result_flow = 0

    while True:
        for v in range(n):
            visited[v] = False
            pred[v] = None

        find_path(s)
        if not visited[t]:
            break

        path = restore_path(t)
        flow = path_capacity(path)
        push_path(path, flow)
        result_flow += flow

    return result_flow

def max_flow(s, t):
    if s == t:
        return None
    build_network()
    return ford_fulkerson(s, t)
```

dfs

```
def find_path(v):
    visited[v] = True
    for e in edges(v):
        u = target(e)
        if not visited[u] and available(e) > 0:
            pred[u] = e
            find_path(u)
```

bfs

```
def find_path(s):
    q = queue()

    visited[s] = True
    q.enqueue(s)

    while not q.empty():
        v = q.dequeue()

        for e in edges(v):
            u = target(e)
            if not visited[u] and available(e) > 0:
                visited[u] = True
                pred[u] = e
                q.enqueue(u)
```

https://github.com/larandaA/alg-ds-snippets

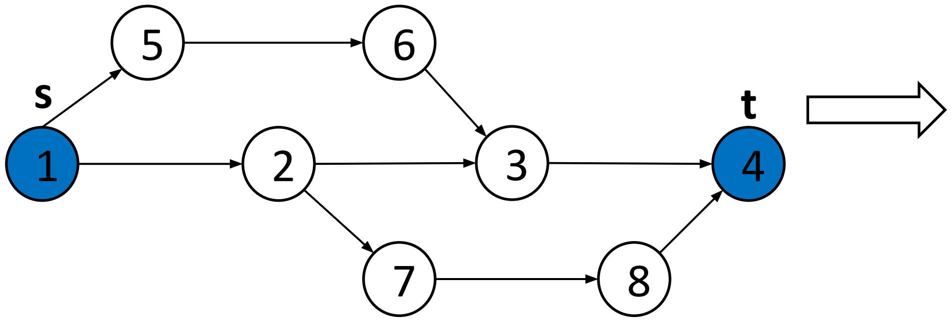
Приложения

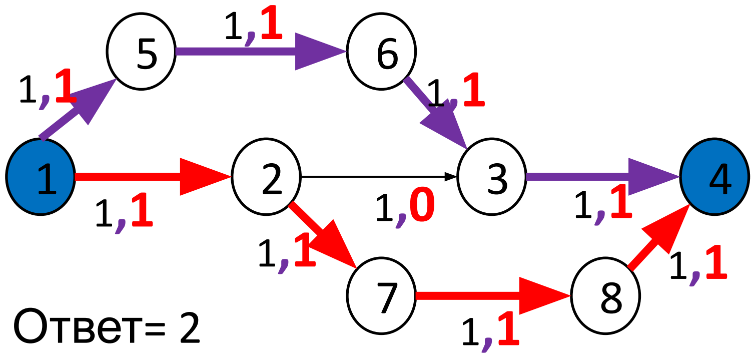
Наибольшее число попарно различных путей

Наибольшее число (s,t)-путей, которые попарно не пересекаются

Задан ориентированый граф, в котором выделены две вершины **s** и **t**.

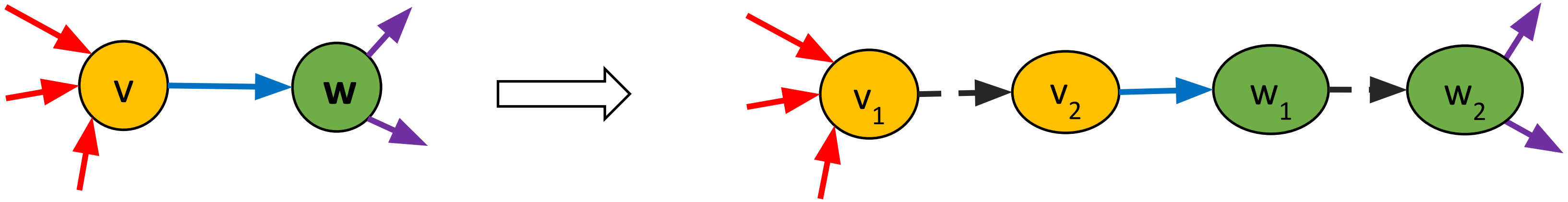
(1) Необходимо найти наибольшее число (**s**,**t**)-путей, которые попарно **не пересекаются по дугам**.





Ответ= 2

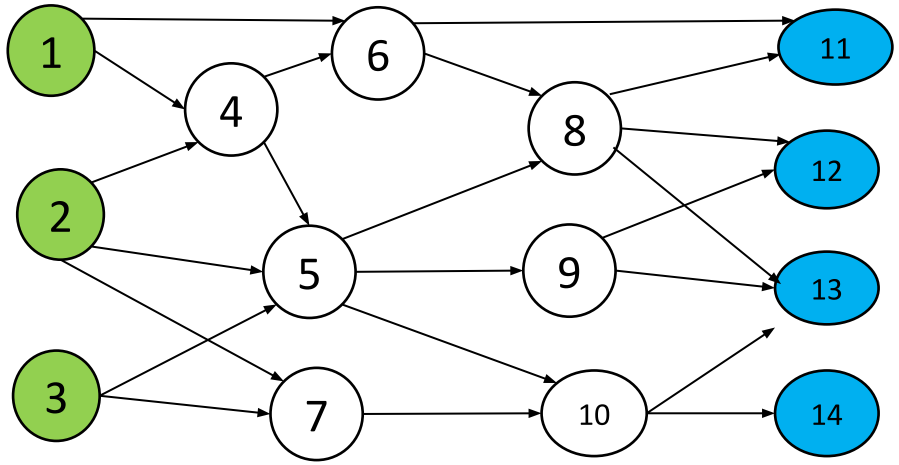
(2) Необходимо найти наибольшее число (**s**,**t**)-путей, которые попарно **не пересекаются по вершинам**.



После преобразования решается задача нахождения наибольшего числа (**s**,**t**)-путей, которые попарно **не пересекаются по дугам** (пропускные способности всех дуг сети полагаются равными 1).

Время работы **O(m·n)**

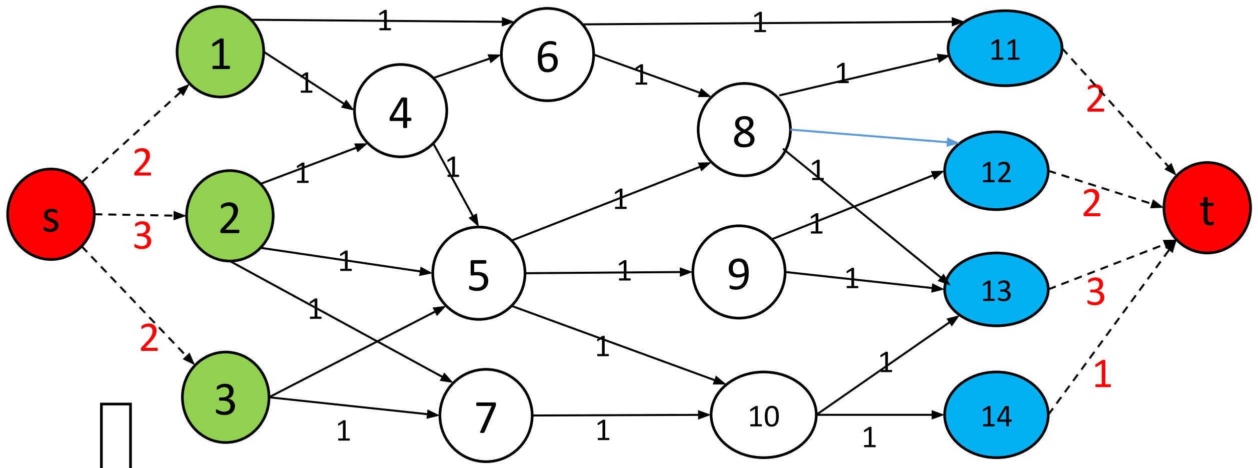
Задан ориентированый граф, в котором выделены два подмножества вершин: $\mathbf{V_1}$ и $\mathbf{V_2}$.
Необходимо найти наибольшее число путей, которые начинаются в $\mathbf{V_1}$ и заканчиваются в $\mathbf{V_2}$ и попарно не пересекаются по дугам.



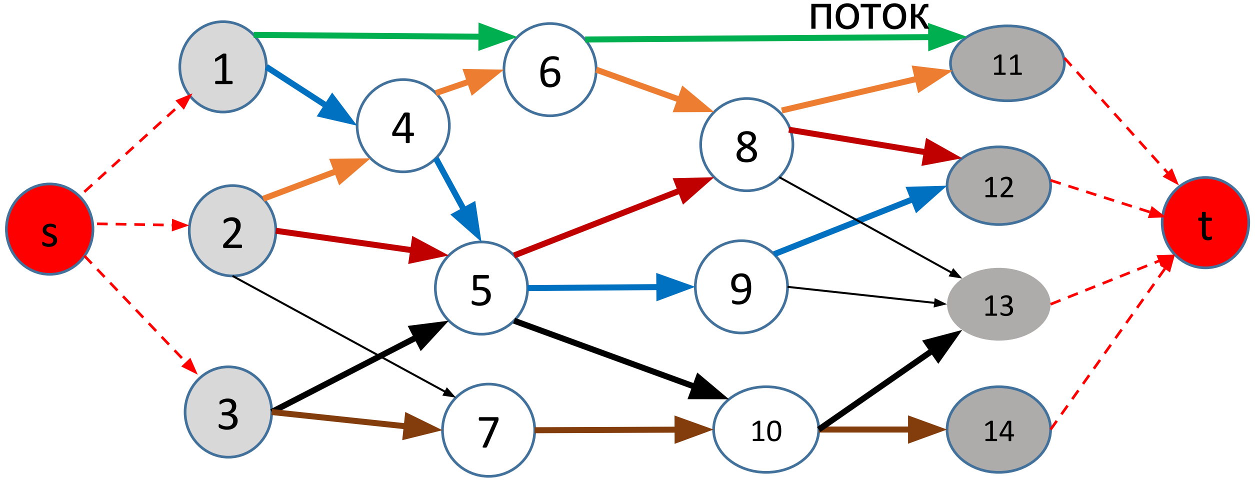
$V_1=\{1, 2, 3\}$

$V_2=\{11, 12, 13, 14\}$

строим сеть



находим максимальный поток



M(f)=6 - решение задачи

Время работы алгоритма:
$O(m^2)$

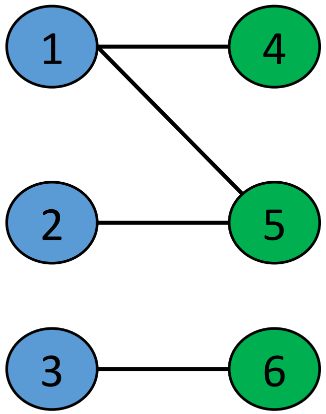
Наибольшее паросочетание в двудольном графе

(*англ*. maximum matching)

Задан двудольный граф. Необходимо найти:

(1) наибольшее паросочетание;

(2) наибольшее паросочетание минимального веса (взвешенный граф).



Паросочетание это некоторое подмножество рёбер графа, в котором никакие два ребра не смежны.

maximum matching – наибольшее паросочетание

maximal matching – максимальное паросочетание

$M_1=\{\{1,5\}, \{3,6\}\}$ максимальное паросочетание

$M_2=\{\{1,4\}, \{2,5\}\}$ паросочетание

$M_3=\{\{1,4\}, \{2,5\}, \{3,6\}\}$ **наибольшее паросочетание** совершенное паросочетание

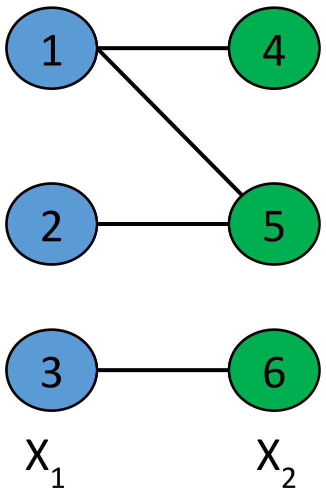
$M_4=\{\{1,5\}, \{1,4\}\}$ НЕ паросочетание

Наибольшее паросочетание в двудольном графе

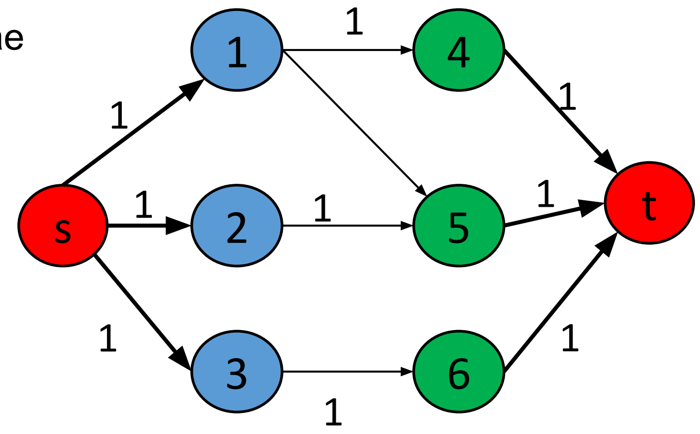
Задан двудольный граф.

Известно разбиение на доли.

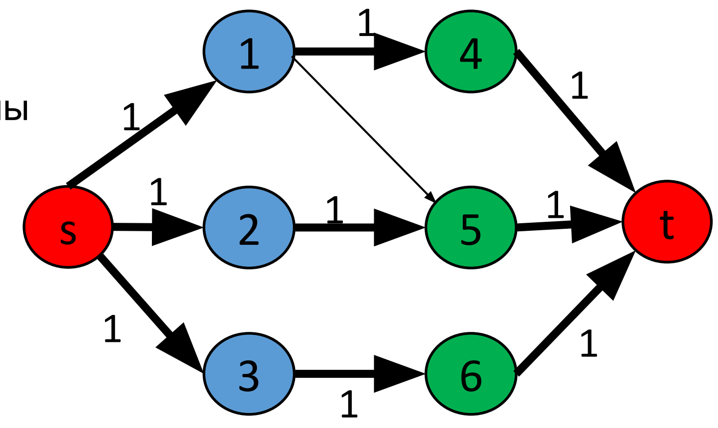
Необходимо найти **наибольшее паросочетание.**



достраиваем до сети



находим максимальный поток



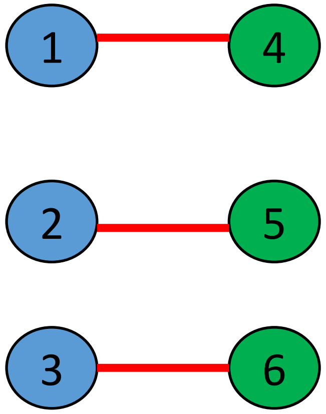
Время работы: **O(n·m)**

время работы bfs,dfs - O(m)

число итераций bfs - min $\{|X_1|, \{|X_2|\}$=O(n)

рёбра двудольного графа, по которым поток равен 1, включаем в наибольшее паросочетание

M={{1,4}, {2,5}, {3,6}}

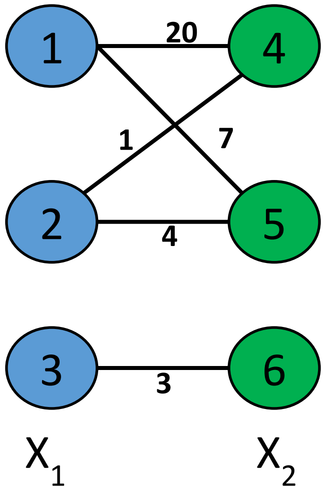


Теория алгоритмов: учеб.-метод. пособие / П. А. Иржавский и др.]. – Минск : БГУ, 2013. С. 77-105.

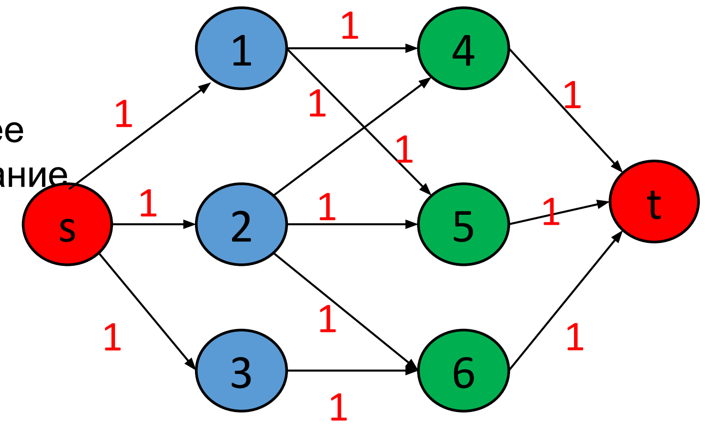
Наибольшее паросочетание минимального веса в двудольном графе

Задан двудольный граф. Каждому ребру которого приписан целочисленный вес c (e)≥0. Известно разбиение вершин двудольного графа на доли.

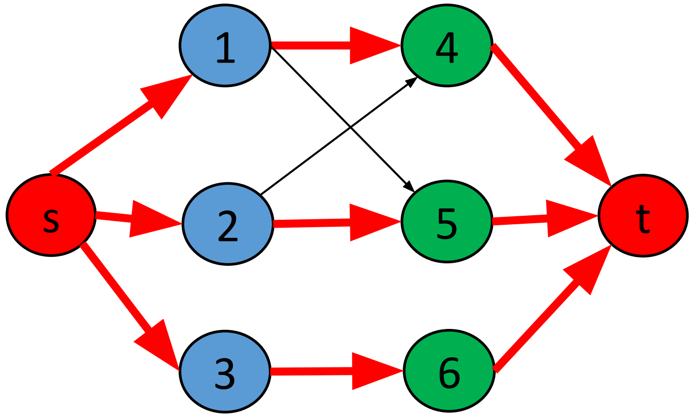
Необходимо найти: **наибольшее паросочетание минимального веса в двудольном графе.**



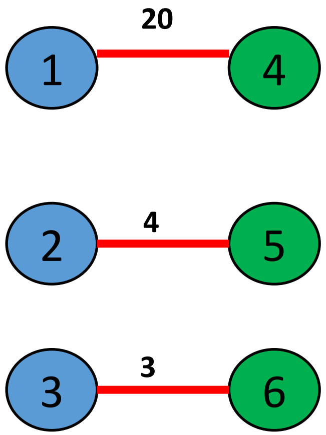
сначала найдём наибольшее паросочетание



строим максимальный поток

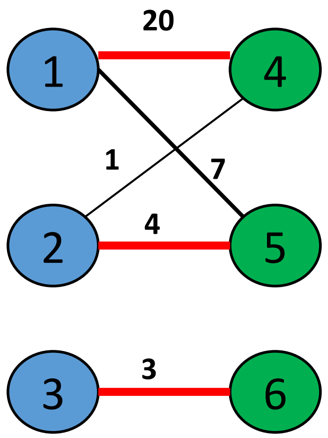


рёбра двудольного графа, по которым поток равен 1, включаем в наибольшее паросочетание



$|M_1|=3$, $c(M_1)=20+4+3=27$

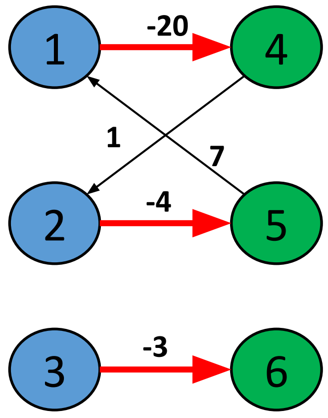
(*продолжение*) наибольшее паросочетание минимального веса



$|M_1|=3$

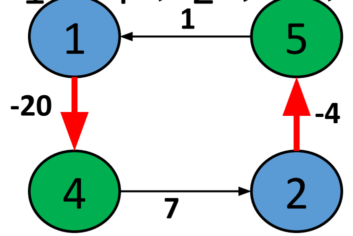
$c(M_1)=20+4+3=27$

ориентируем рёбра графа

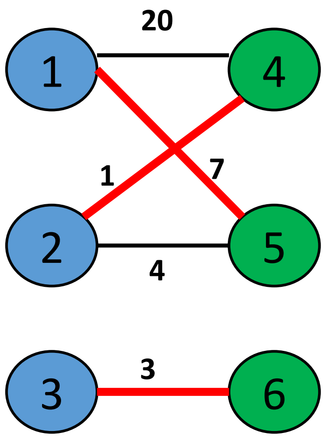


контур отрицательного веса (=-16):

1 -> 4 -> 2 -> 5 -> 1



перестраиваем паросочетание

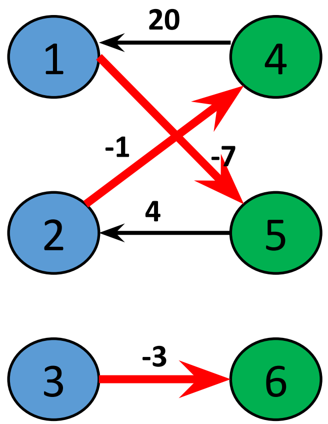


новое паросочетание

$|M_2|=3$,

$c(M_2)=1+7+3=11$

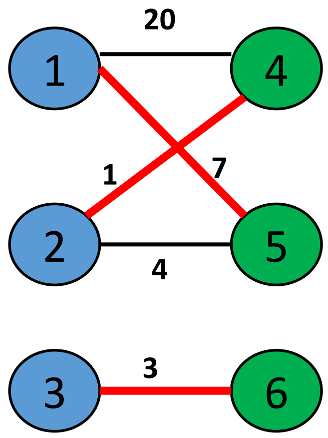
повторяем процесс



контур отрицательного веса: НЕТ

$M_2=\{\{1,5\}, \{2,4\}, \{3,6\}\}$ – наибольшее паросочетание минимального веса

$c(M_2)=11$



Время работы алгоритма

$$O(c^{max} \cdot m \cdot n^2)$$

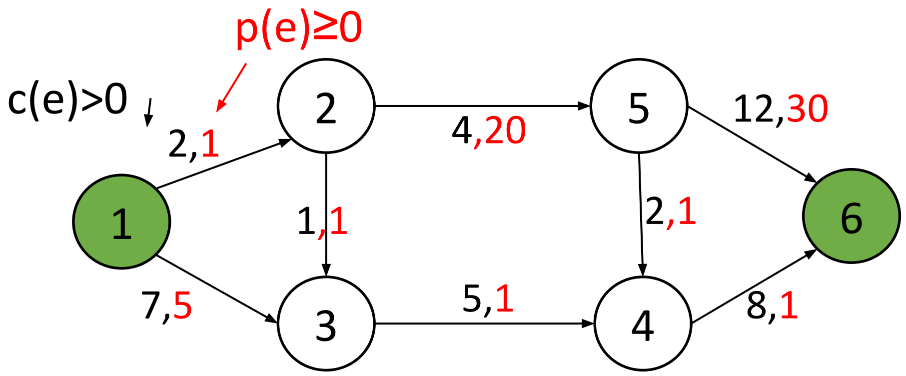
$$O(n \cdot m) \quad + \quad O(c^{max} \cdot n) \quad \cdot \quad (\, O(n \cdot m) + O(m))$$

поиск наибольшего паросочетания в двудольном графе;

максимальный вес наибольшего паросочетания
(в паросочетании не может быть более, чем n рёбер), поэтому данной величиной можно оценить наибольшее число итераций поиска контура отрицательного веса;

поиск контура отрицательного веса алгоритмом Форда –Беллмана и перестройка наибольшего паросочетания вдоль контура отрицательного веса

Максимальный поток минимальной стоимости
(англ. max flow min cost)

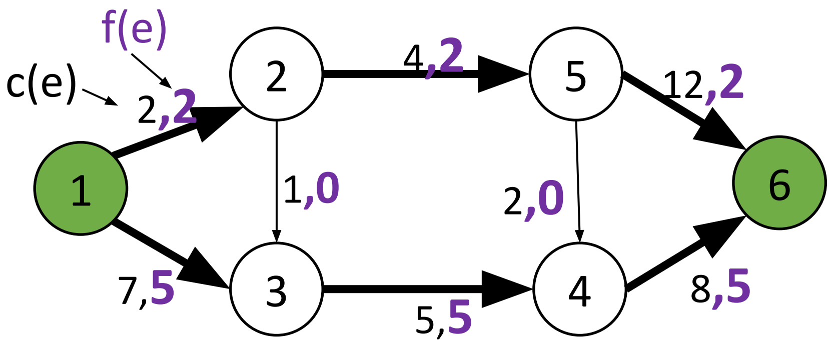


Стоимость потока f:

$$\boxed{P(f) = \sum_{e \in E} f(e) \cdot p(e)}$$

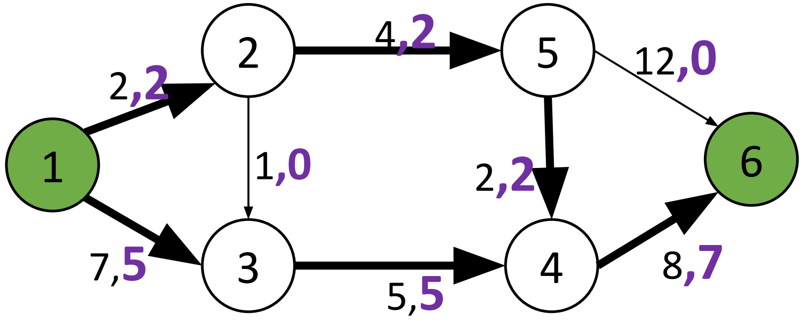
Максимальный поток, но среди всех максимальных потоков его удельная стоимость не является минимальной.

$$M(f) = 7$$



$$P(f) = f(1{,}2) \cdot p(1{,}2) + f(2{,}5) \cdot p(2{,}5) + f(5{,}6) \cdot p(5{,}6) + f(1{,}3) \cdot p(1{,}3) + \\ + f(3{,}4) \cdot p(3{,}4) + f(4{,}6) \cdot p(4{,}6) = 2 \cdot 1 + 2 \cdot 20 + 2 \cdot 30 + 5 \cdot 5 + 5 \cdot 1 + 5 \cdot 1 = 137$$

$$M(f) = 7$$



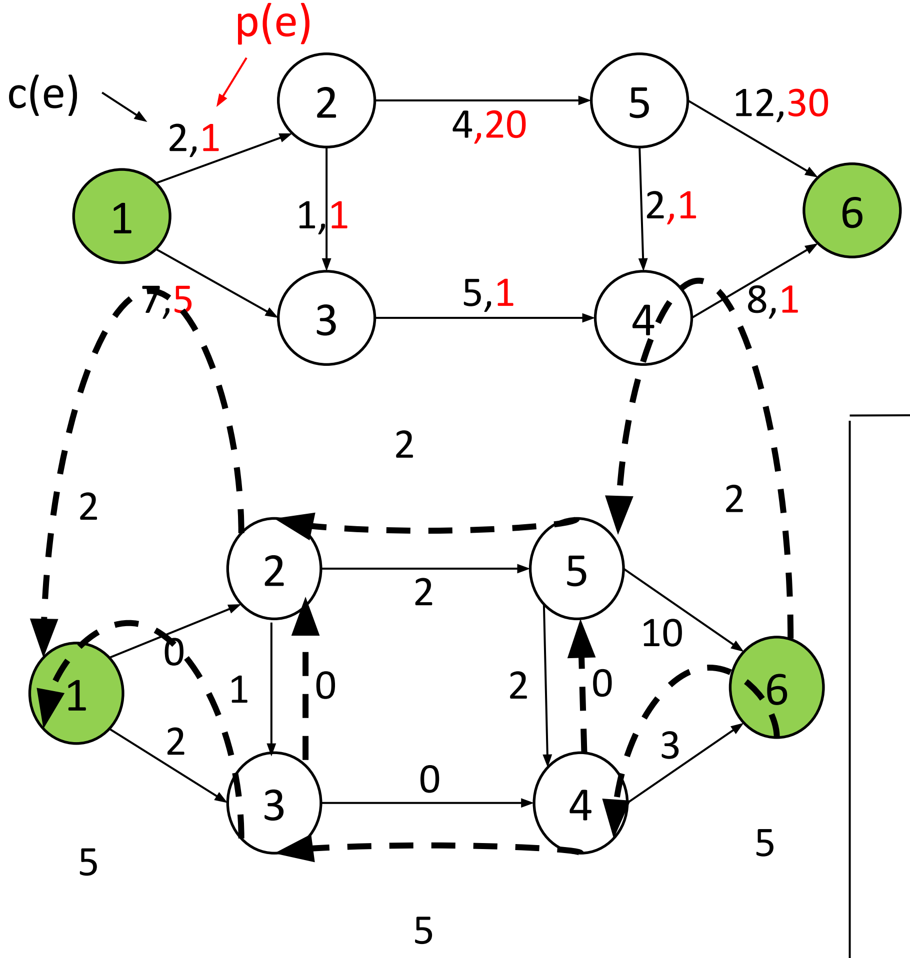
$$P(f) = f(1{,}2) \cdot p(1{,}2) + f(2{,}5) \cdot p(2{,}5) + f(5{,}4) \cdot p(5{,}4) + f(1{,}3) \cdot p(1{,}3) + \\ + f(3{,}4) \cdot p(3{,}4) + f(4{,}6) \cdot p(4{,}6) = 2 \cdot 1 + 2 \cdot 20 + 2 \cdot 1 + 5 \cdot 5 + 5 \cdot 1 + 7 \cdot 1 = 81$$

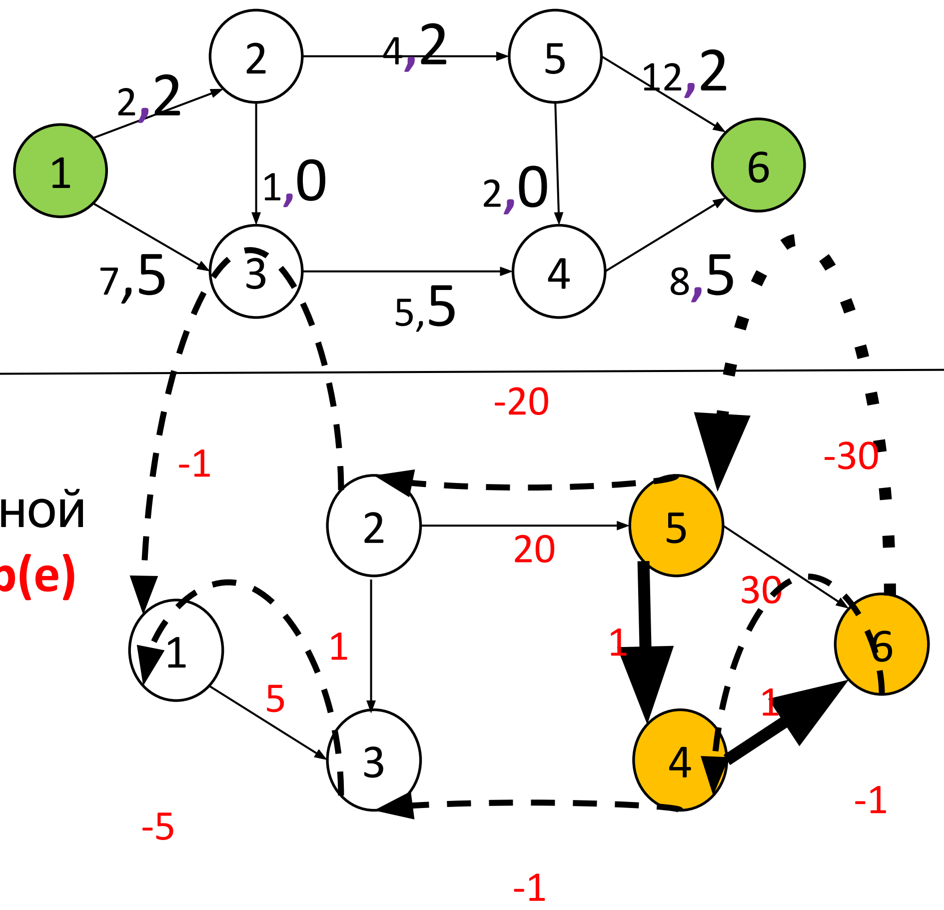
Максимальный поток минимальной стоимости

Метод устранения отрицательных циклов

исходная сеть

максимальный поток





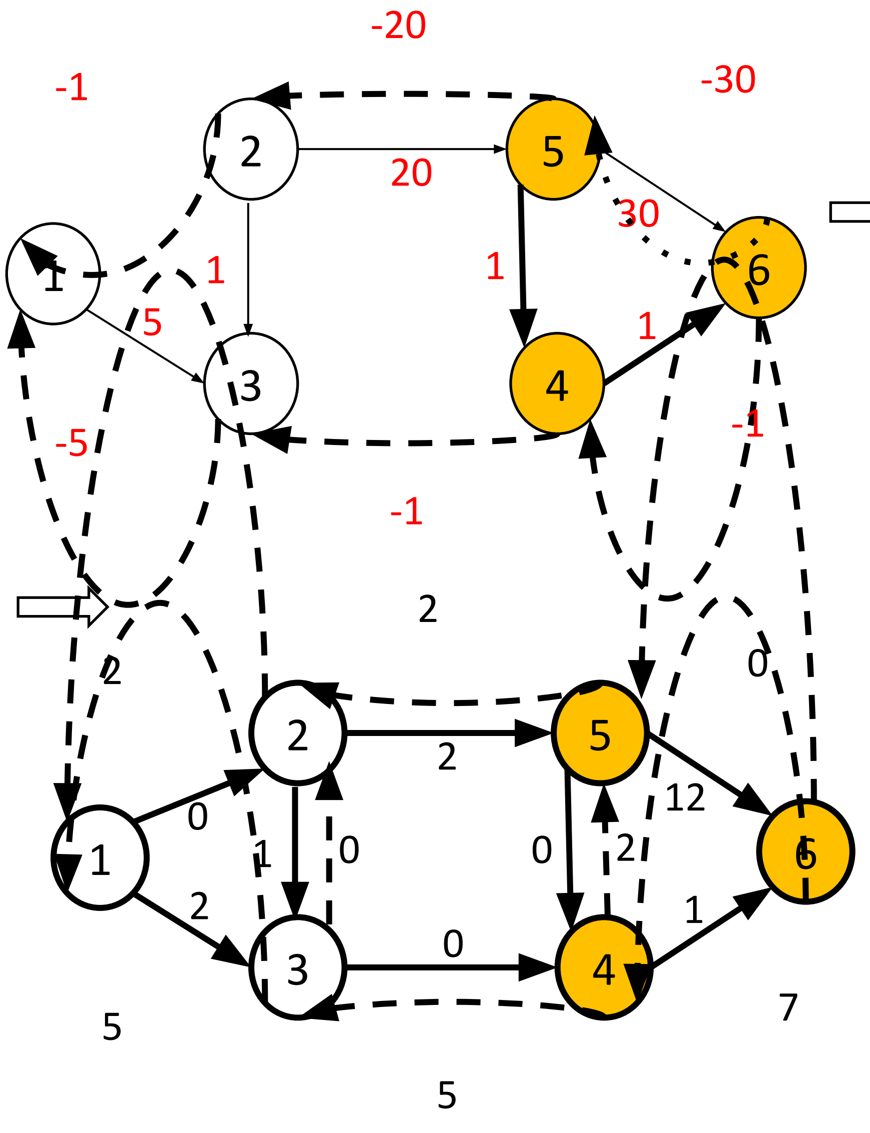
дуге ***e*** исходной сети **p(e)**

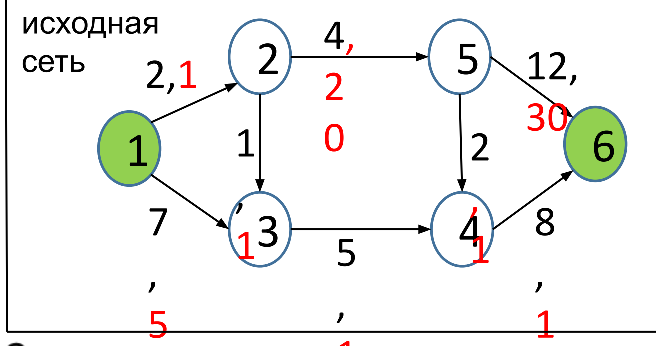
обратной дуге $\bar{e}$ ставим **−p(e)**

сеть остаточных пропускных способностей на последней итерации алгоритма построения максимального потока

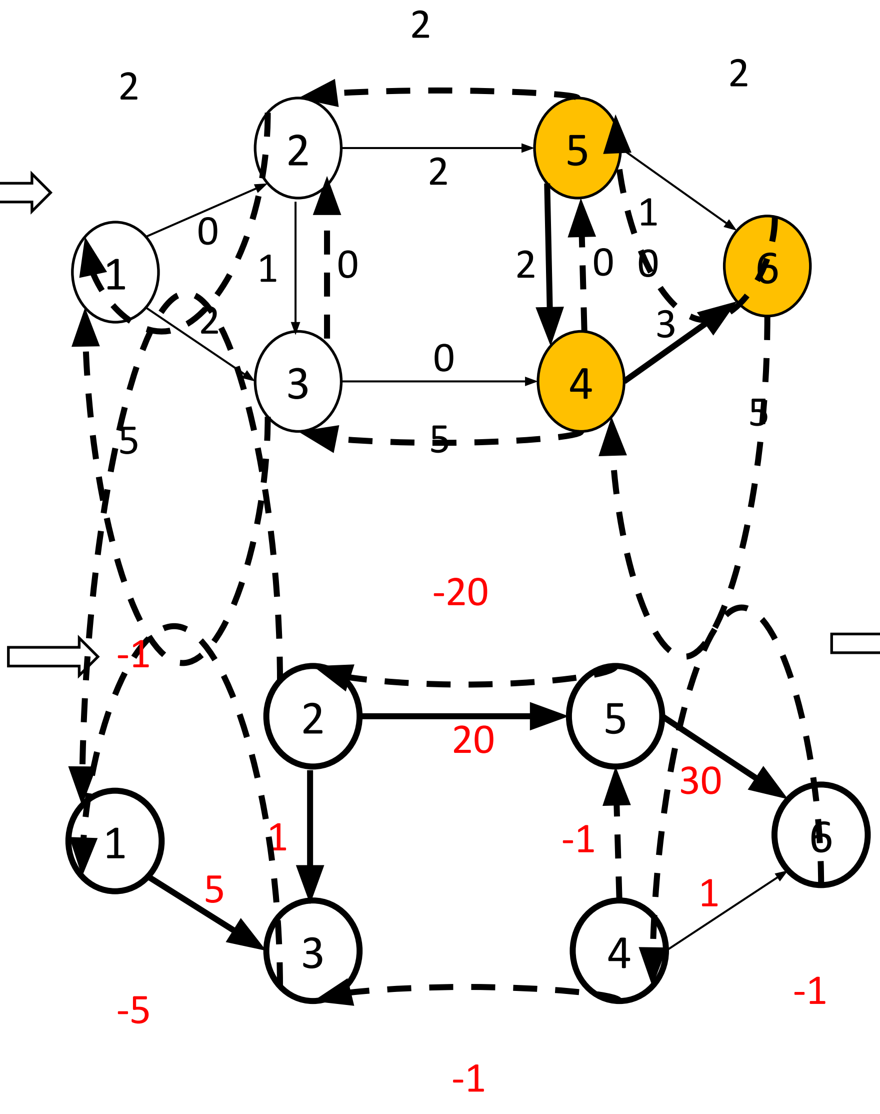
C={ (6,5),(5,4),(4,6)} – контур отрицательной стоимости **-28**; перераспределяя вдоль контура 1 единицу потока, получим поток той же величины, но стоимость которого меньше на **|p'(C)|** единиц, чем у текущего потока;

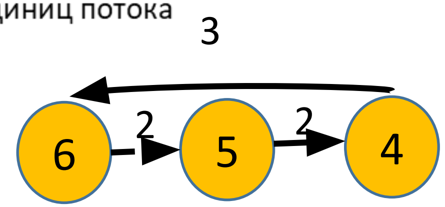
берём сеть остаточных пропускных способностей последней итерации алгоритма построения максимального потока и вдоль контура отрицательного веса перераспределяем поток

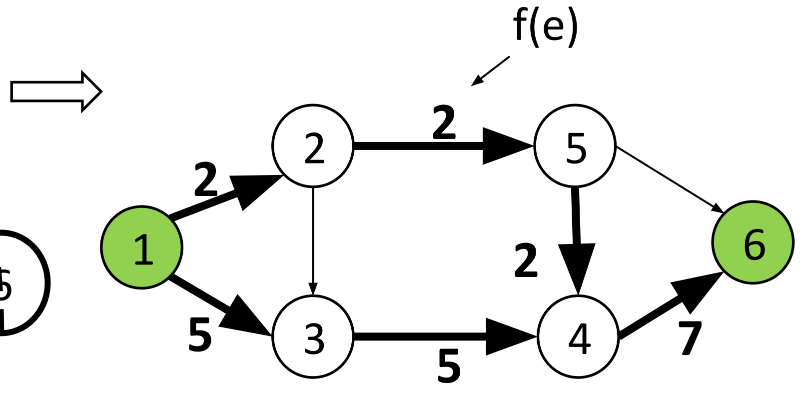




$c_f^{\min} = 2$ количество перераспределяемых единиц потока







$P(f) = 137 - 2 \cdot 28$
$= 81$

нет контура отрицательной стоимости

$M(f) = 7$
$P(f) = 81$

ФПМИ БГУ

Время работы: $O(c^{max} \cdot p^{max} \cdot m \cdot n^2 + n \cdot m^2)$

$O(n \cdot m^2)$ – поиск максимального потока, например, алгоритмом Эдмондса – Карпа

+

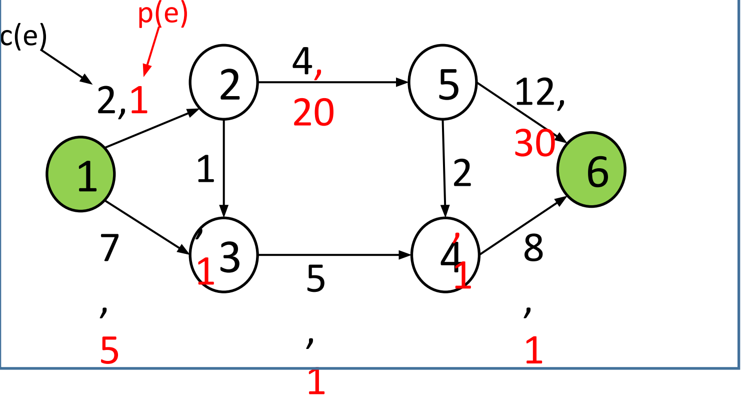
$O(c^{max} \cdot n \cdot p^{max}) \cdot O(n \cdot m)$

наибольшая возможная удельная стоимость максимального потока (по предположению сеть целочисленная и нет кратных дуг)

поиск циклов отрицательной удельной стоимости, например, алгоритмом Форда – Беллмана

Максимальный поток минимальной стоимости

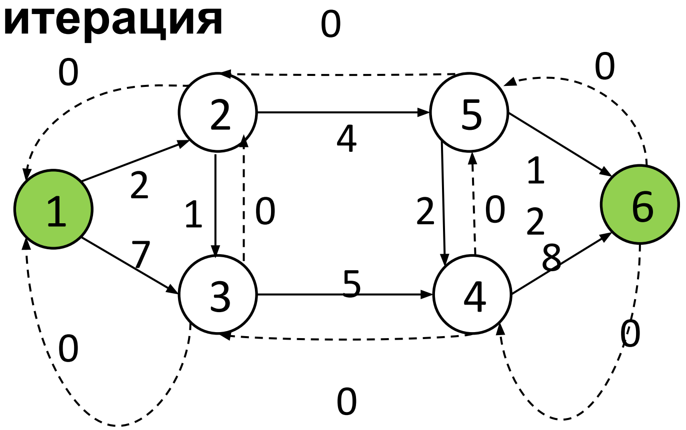
Метод минимальных путей

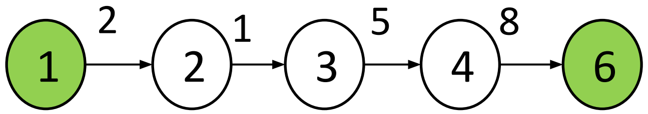


Исходная сеть

$M(f) = 0$
$P(f) = 0$

1-я итерация

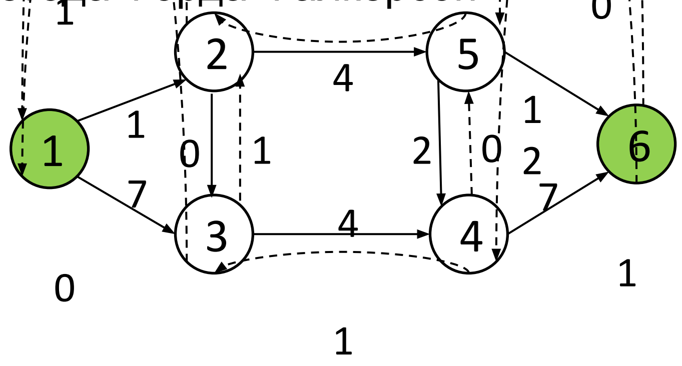




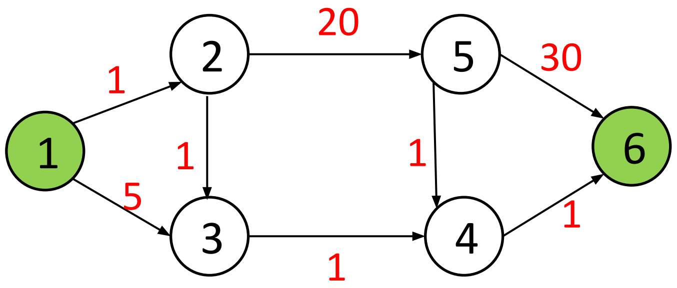
$c_f^{\min} = 1$

сеть остаточных пропускных способностей

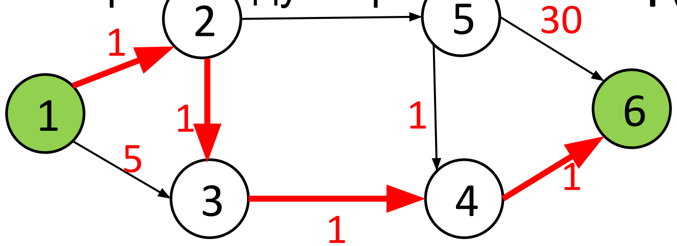
вдоль данного пути увеличиваем текущий поток, как и на итерациях метода Форда-Фалкерсон



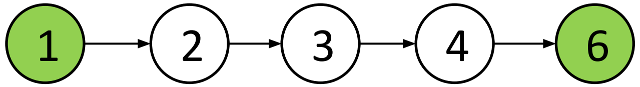
$M(f) = 1$
$P(f) = 4$

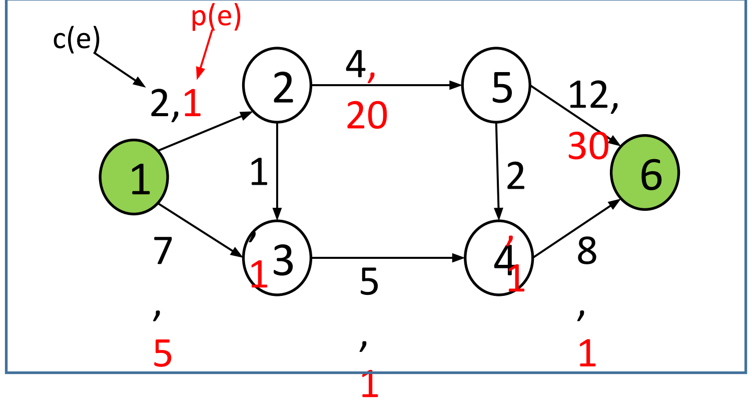


оставляем дуги, для которых остаточная пропускная способность >0;
дуге e исходной сети приписываем **p(e)**,
а её обратной дуге приписываем **–p(e)**

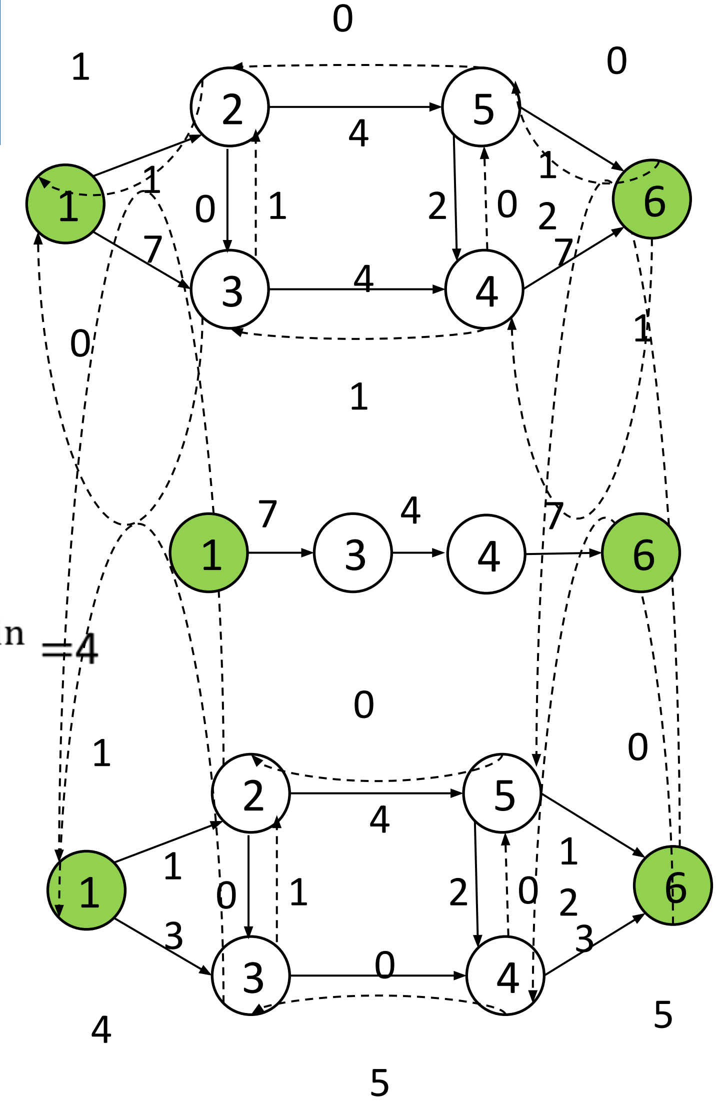


в сети могут быть дуги отрицательного веса, но нет отрицательных контуров; находим кратчайший по удельной стоимости (1,6)-путь:





2-я итерация

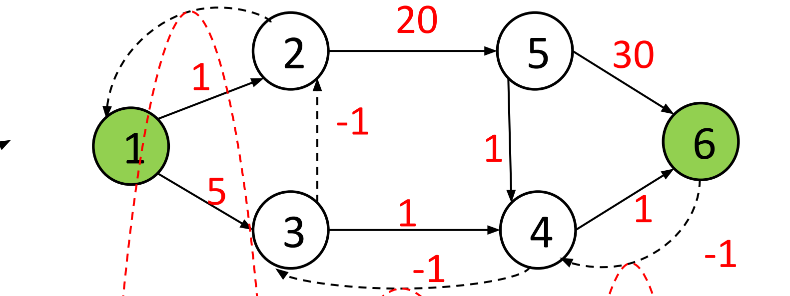


сеть остаточных пропускных способностей

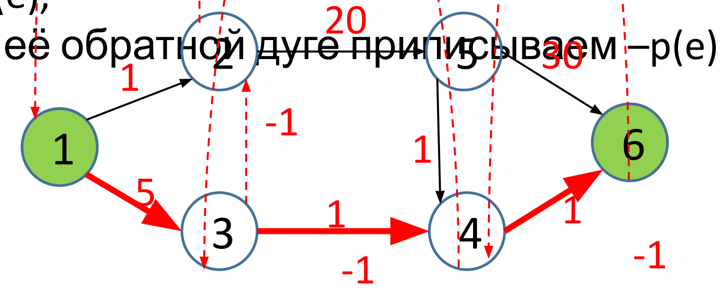
вдоль данного пути увеличиваем текущий поток

$$c_f^{\min} = 4$$

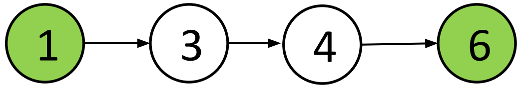
$$M(f) = 5$$
$$P(f) = 32$$

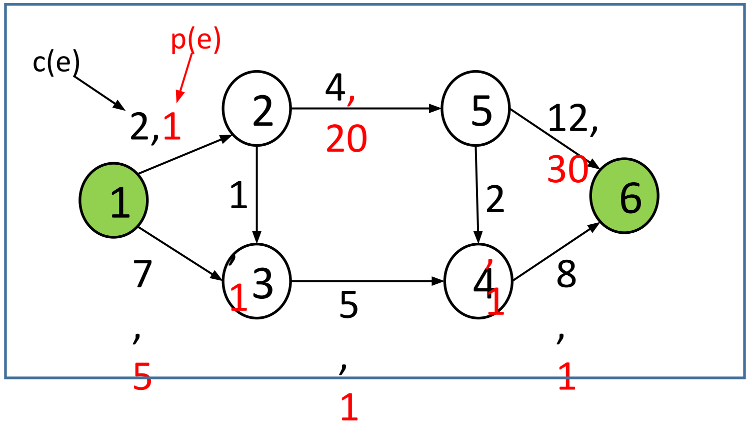


оставляем дуги, для которых остаточная
пропускная способность >0;
дуге e исходной сети приписываем p(e),
а её обратной дуге приписываем –p(e)



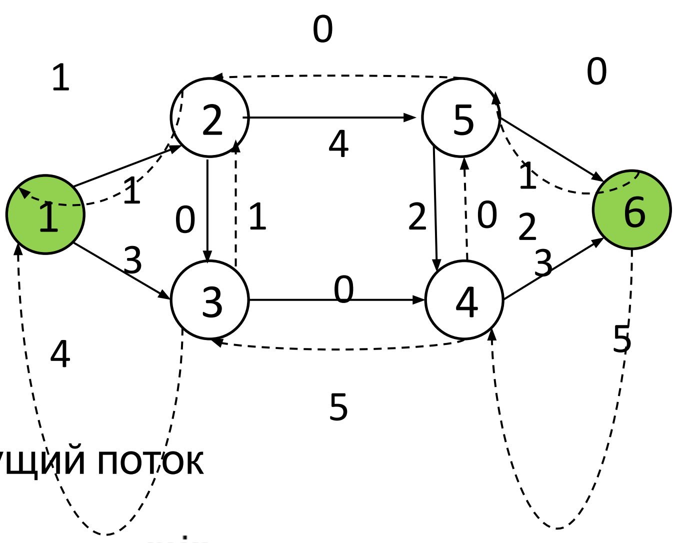
находим кратчайший по удельной стоимости (1,6)-путь:





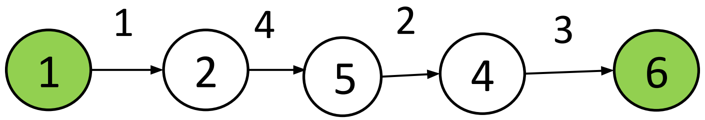
сеть остаточных пропускных способностей

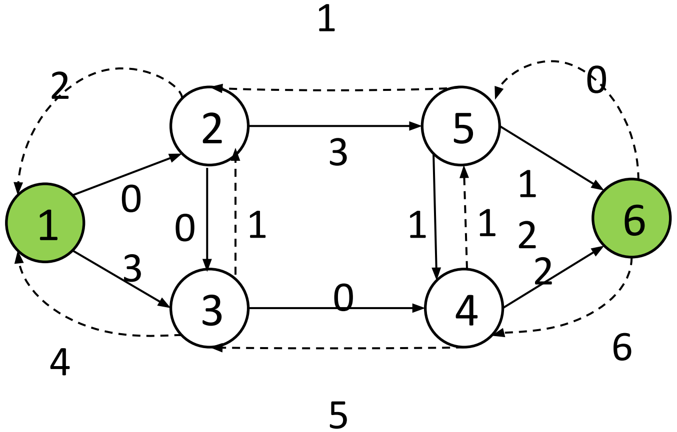
3-я итерация



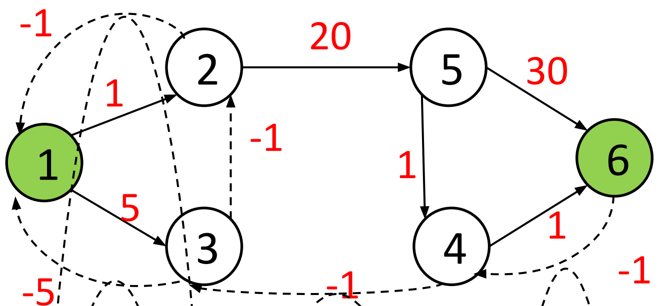
$c_f^{\min} = 1$

вдоль данного пути увеличиваем текущий поток

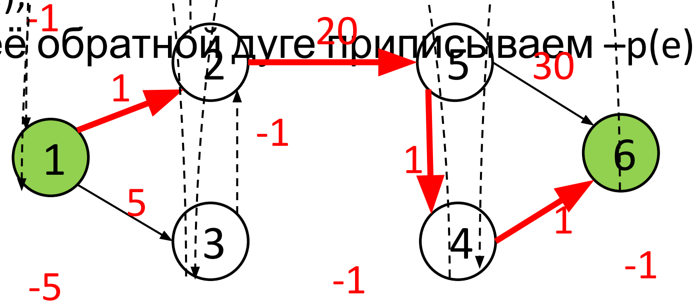




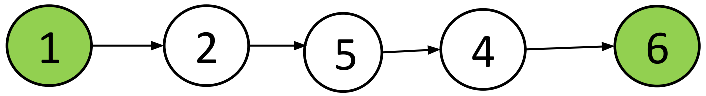
$M(f) = 6$
$P(f) = 55$

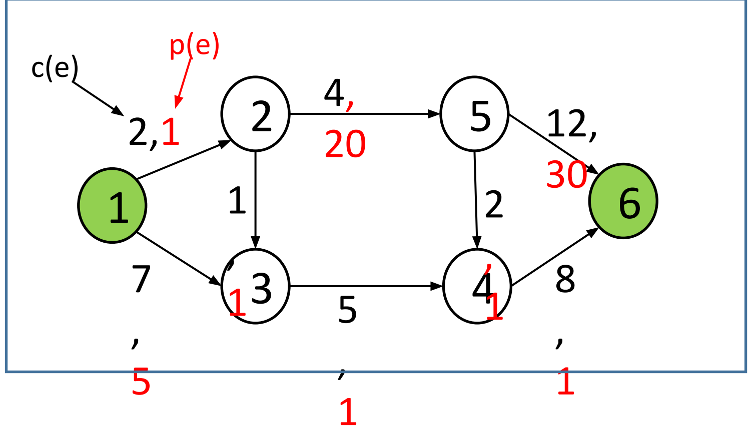


оставляем дуги, для которых остаточная
пропускная способность >0;
дуге e исходной сети приписываем p(e),
а её обратной дуге приписываем –p(e)



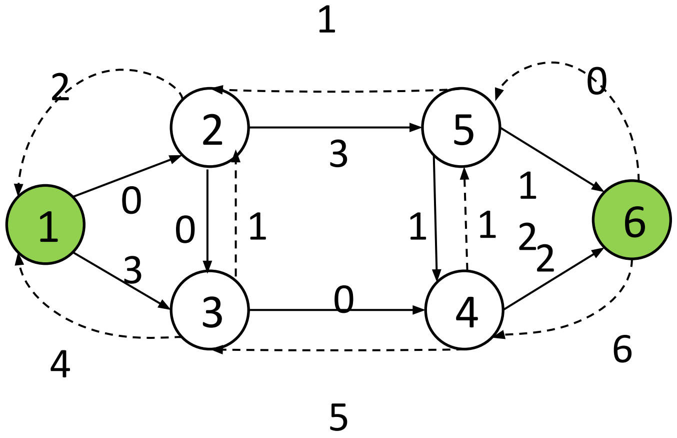
находим кратчайший по удельной стоимости (1,6)-путь:



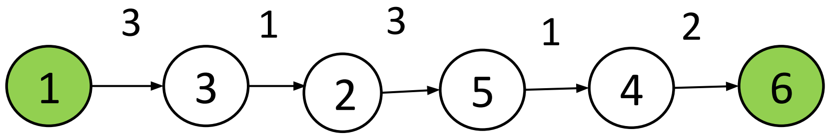


сеть остаточных
пропускных способностей

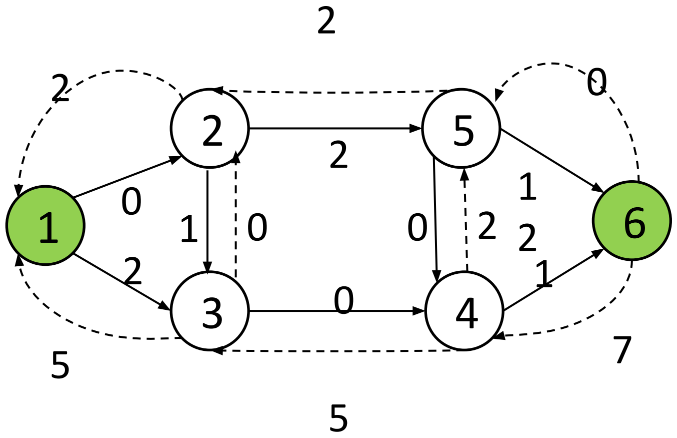
4-я итерация



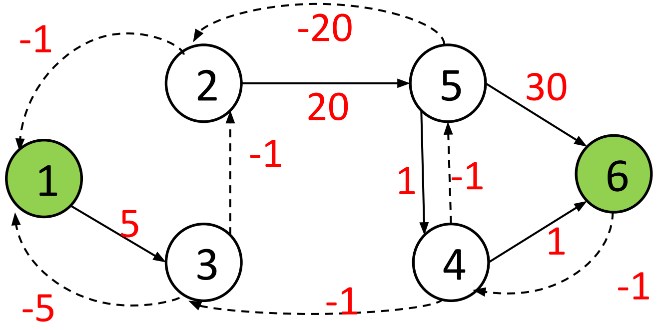
вдоль данного пути увеличиваем текущий поток



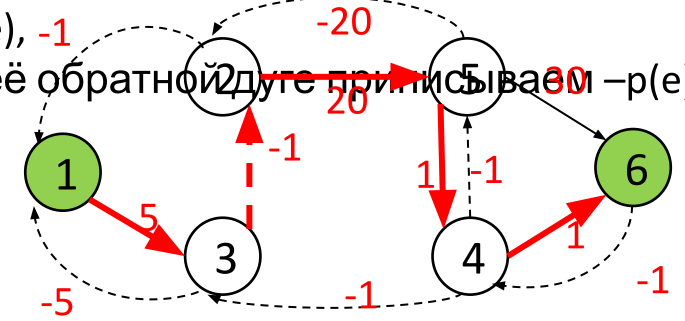
$$c_f^{\min} = 1$$



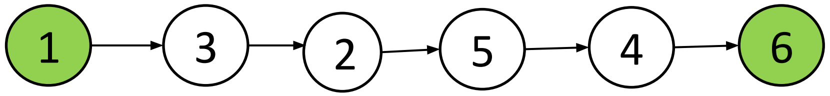
$$M(f) = 7$$
$$P(f) = 81$$



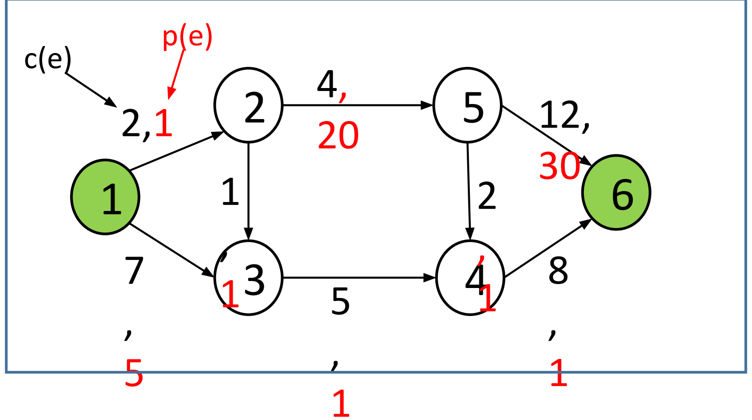
оставляем дуги, для которых
остаточная
пропускная способность >0;
дуге e исходной сети приписываем
p(e), -1
а её обратной дуге приписываем –p(e)



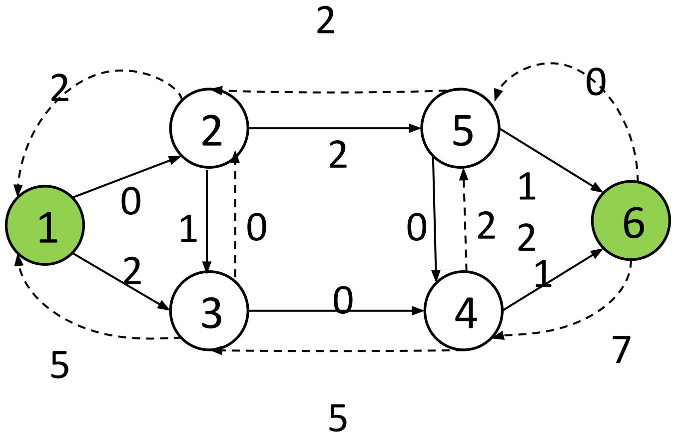
находим кратчайший по удельной
стоимости (1,6)-путь:

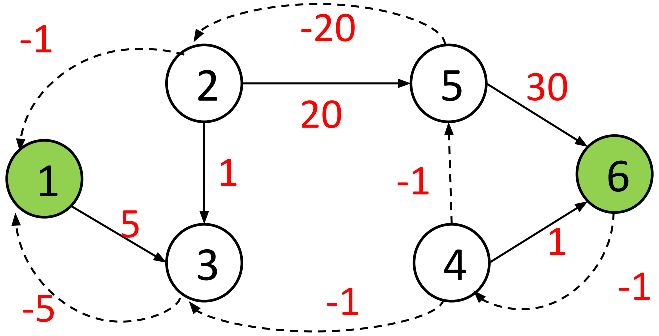


5-я итерация



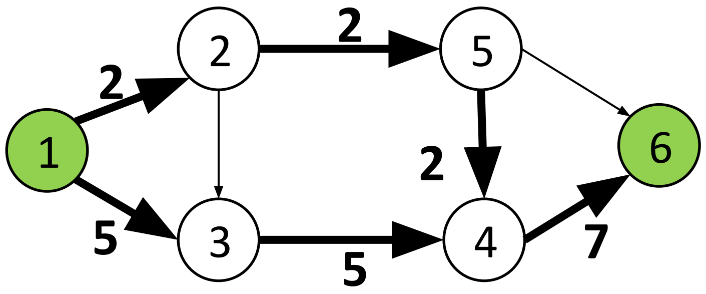
сеть остаточных
пропускных способностей





оставляем дуги, для которых остаточная
пропускная способность >0;
дуге e исходной сети приписываем p(e),
а её обратной дуге приписываем –p(e)

НЕТ (1,6)-пути



$M(f) = 7$

$$P(f) = f(1{,}2)\cdot p(1{,}2) + f(2{,}5)\cdot p(2{,}5) + f(5{,}4)\cdot p(5{,}4) + f(1{,}3)\cdot p(1{,}3) + f(3{,}4)\cdot p(3{,}4) + f(4{,}6)\cdot p(4{,}6) = 2\cdot 1 + 2\cdot 20 + 2\cdot 1 + 5\cdot 5 + 5\cdot 1 + 7\cdot 1 = 81$$

Время работы

$O(c^{max}\cdot m\cdot n^2)$	$O(p^{max}\cdot m^2\cdot n^2)$
$O(c^{max}\cdot n) \quad \cdot \quad (O(n\cdot m) + O(m))$	$O(p^{max}\cdot n\cdot m) \quad \cdot \quad (O(n\cdot m) + O(m))$
так как сеть целочисленная, нет кратных дуг и на каждой итерации метода минимальных путей строится поток большей величины, то число итераций алгоритма ограничено сверху наибольшей возможной величиной потока конкретной индивидуальной задачи, т.е. $c^{max}\cdot n$. время работы алгоритма Форда – Беллмана; модификация потока вдоль найденного увеличивающего пути.	число шагов запуска алгоритма нахождения кратчайшего пути (доказательство оценки выполняется по той же схеме, как и в алгоритме Эдмондса – Карпа) время работы алгоритма Форда – Беллмана; модификация потока вдоль найденного увеличивающего пути.

Максимальный поток минимальной стоимости

Метод
устранения отрицательных
циклов

$$O(c^{max} \cdot p^{max} \cdot m \cdot n^2 + n \cdot m^2)$$

Метод
минимальных путей

$$O(c^{max} \cdot m \cdot n^2)$$

$$O(p^{max} \cdot m^2 \cdot n^2)$$

оба алгоритма –
псевдополиномиальные

Общие задачи в iRunner для закрепления навыков

[0.11 Максимальный поток в сети (простая версия)]()
[0.12 Максимальный поток в сети (большие ограничения, по желанию)]()

БЕЛОРУССКИЙ
ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ

Первая часть курс лекций по теории алгоритмов завершена

Никогда не останавливайтесь, расширяйте и углубляйте свои знания – это того стоит!