


ПОТОКИ ПАРОСОЧЕТАНИЯ

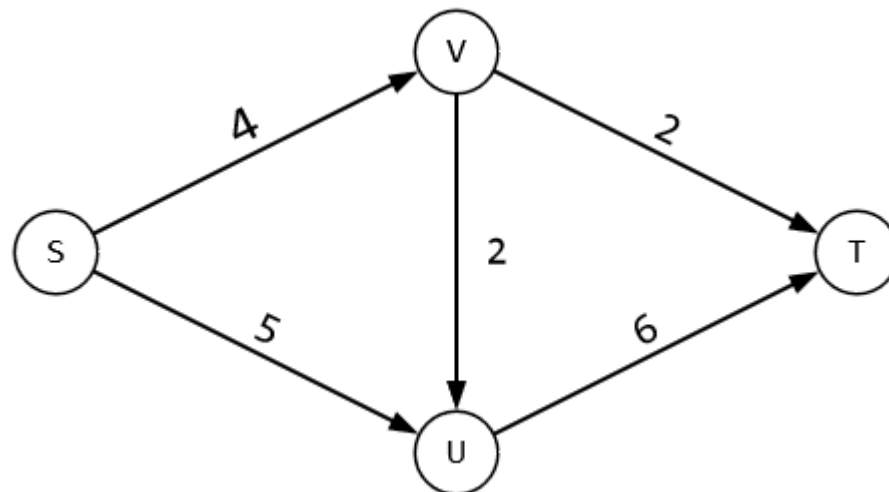


Сеть

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 .



Поток

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

- 1) $f(u, v) = -f(v, u)$ (антисимметричность);
- 2) $f(u, v) \leq c(u, v)$ (ограничение пропускной способности), если ребра нет, то $f(u, v) = 0$
- 3) $\sum_v f(u, v) = 0$ для всех вершин u , кроме s и t (закон сохранения потока).

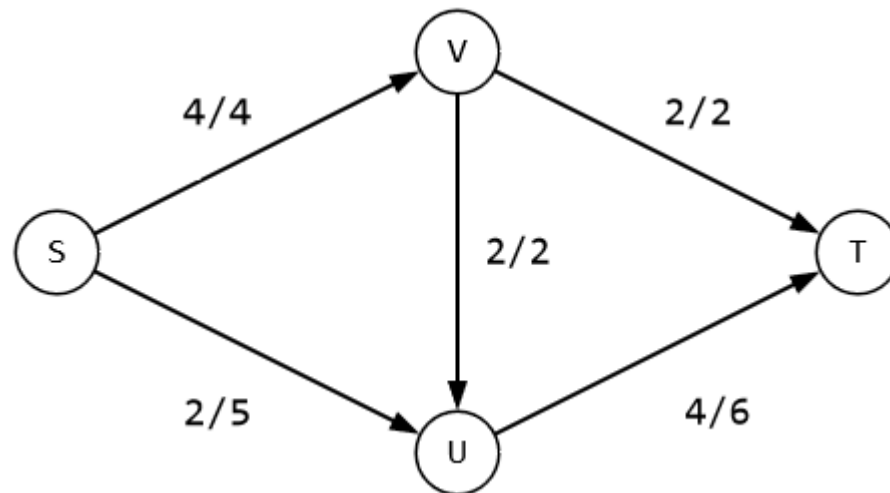
Величина потока f определяется как $|f| = \sum_{v \in V} f(s, v)$.

В данном случае:

Исток S

Сток T

Величина потока $4 + 2 = 6$ ($f(S, V) + f(S, U)$) – сумма потоков выходящих из вершины S)



Задача о максимальном потоке.

Разрез

Разрез — это набор ребер, удаление которых делает невозможным путь от источника к стоку.

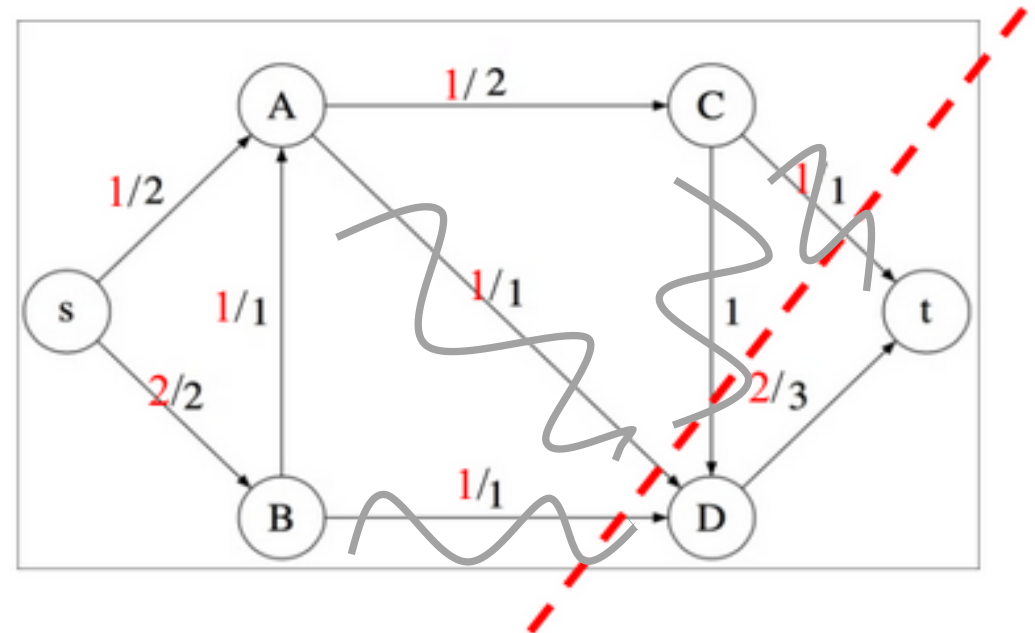
Пропускная способность разреза — сумма пропускных способностей ребер, проходящих через разрез

Поток, проходящий через разрез — сумма значений потока проходящих через разрез

В данном случае:

Поток через разрез = 3

Пропускная способность разреза = 4



Задача о максимальном потоке.

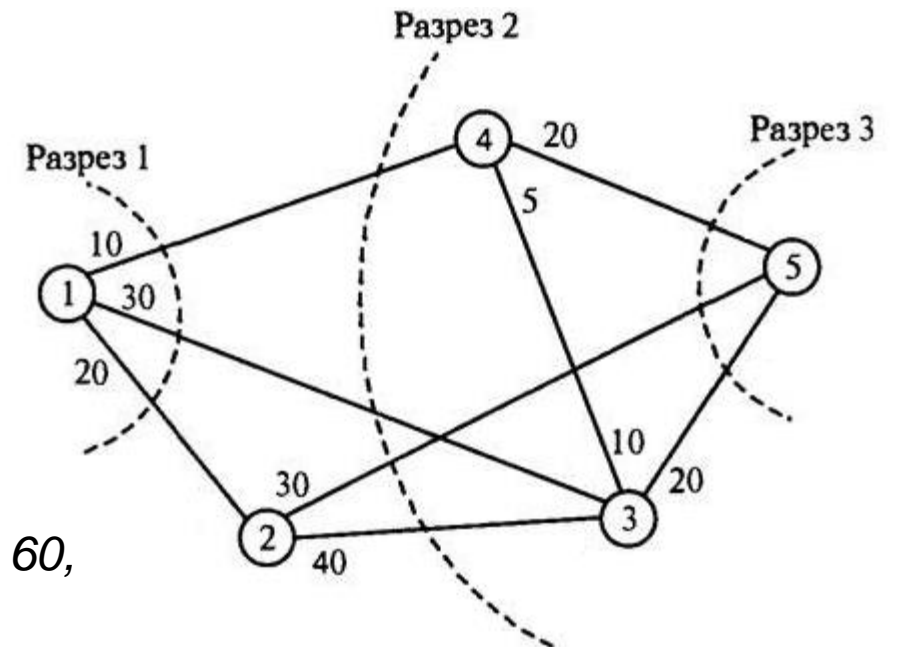
Разрез

Минимальным разрезом (англ. *minimum cut*) называется разрез с минимально возможной пропускной способностью

Если $f(S, T) = c(S, T)$, то поток - максимален, а разрез - минимален

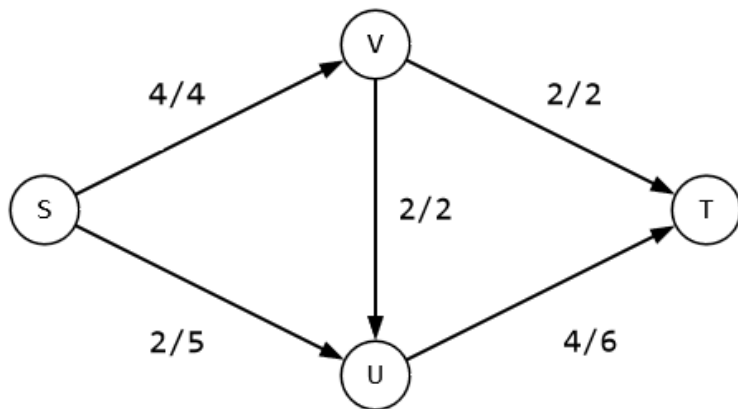
Разрез	"Разрезанные" ребра	Пропускная способность
1	(1,2),(1,3),(1,4)	$10+30+20=60$
2	(1,3),(1,4),(2,3),(2,5)	$30+10+40+30=110$
3	(2,5),(3,5),(4,5)	$30+20+20=70$

Минимальный разрез — 1 с пропускной способностью 60,
Значит максимальная пропускная способность == 60

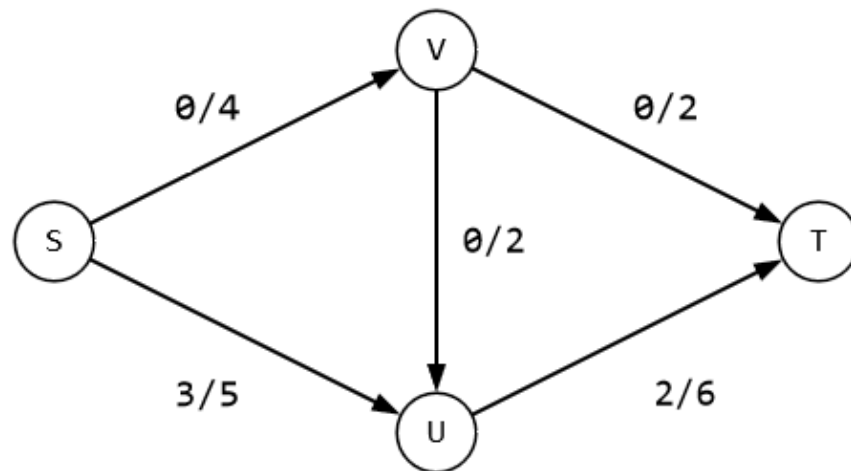


Остаточная сеть. Принцип построения

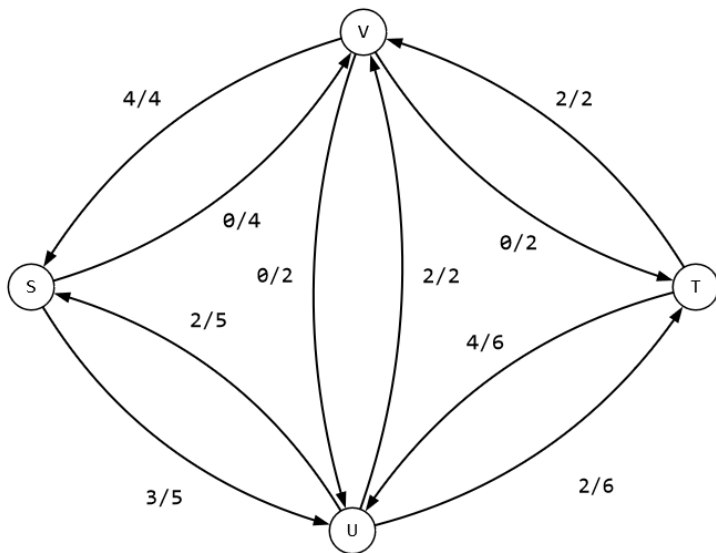
1 шаг



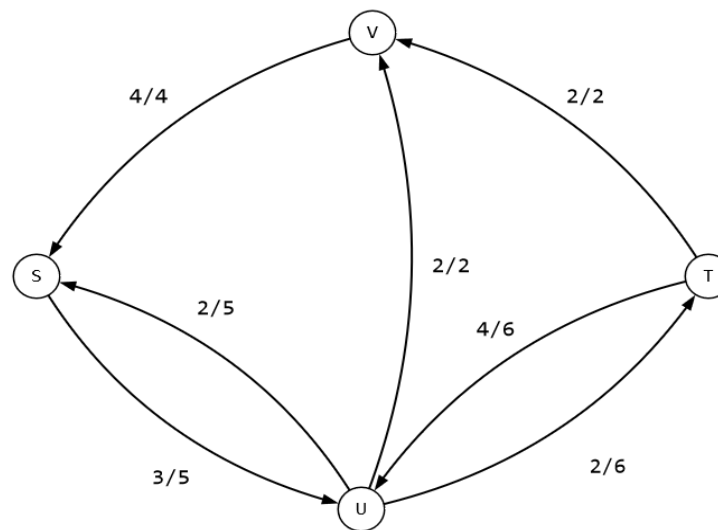
2 шаг



3 шаг



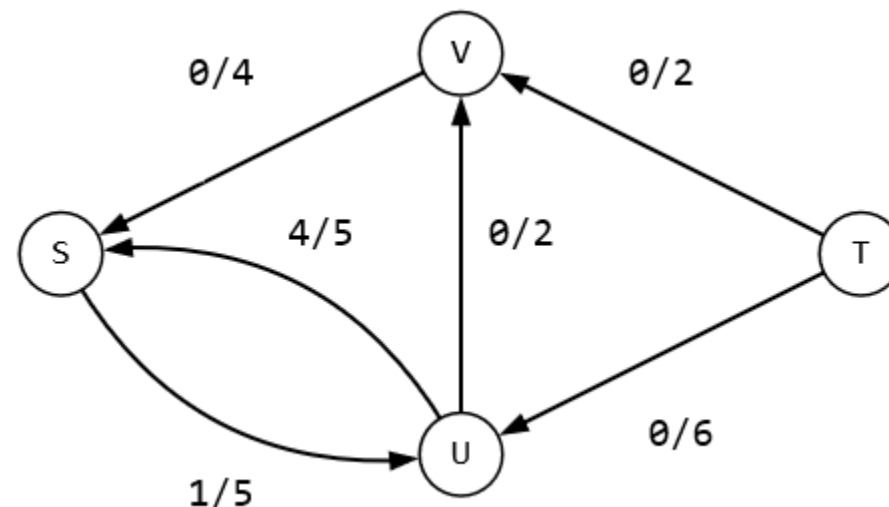
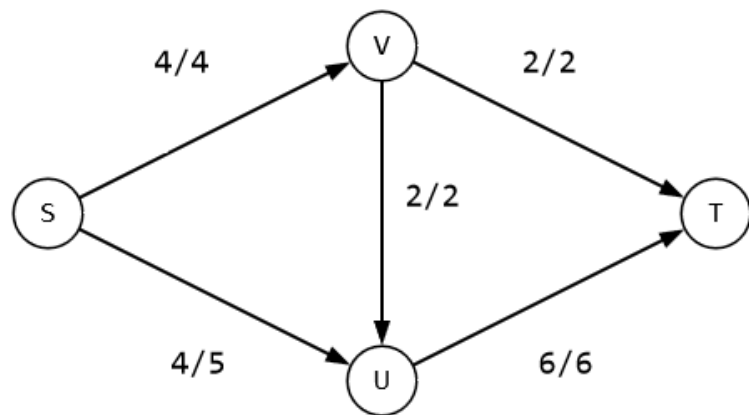
4 шаг



Задача о максимальном потоке. Теорема Форда-Фалкерсона

По данной теореме, для любого максимального потока, для его остаточной сети не будет существовать пути из истока в сток

Построим сеть с максимальным потоком и его остаточную сеть



Алгоритм Форда-Фалкерсона

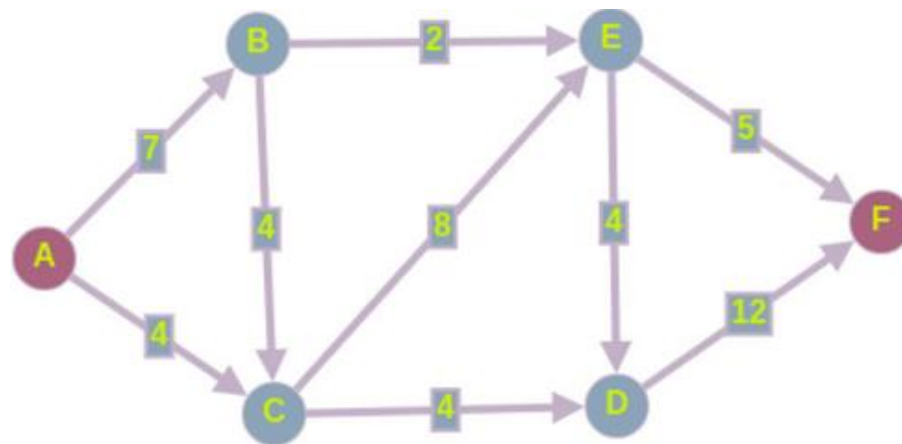
1. Обнуляем все потоки. Остаточная сеть изначально совпадает с исходной сетью.
2. В остаточной сети находим любой путь из источника в сток. Если такого пути нет, останавливаемся.
3. Пускаем через найденный путь (дополняющий путь) максимально возможный поток:

1. На найденном пути в остаточной сети ищем ребро с минимальной пропускной способностью C_{min} .

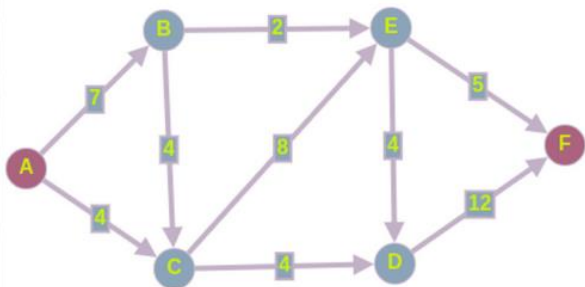
2. Для каждого ребра на найденном пути увеличиваем поток на C_{min} , а в противоположном ему — уменьшаем на C_{min} .

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



Исходный граф

$$6 + 4 = 10$$

$$5 + 5 = 10$$

поток через сеть?

10



4-2

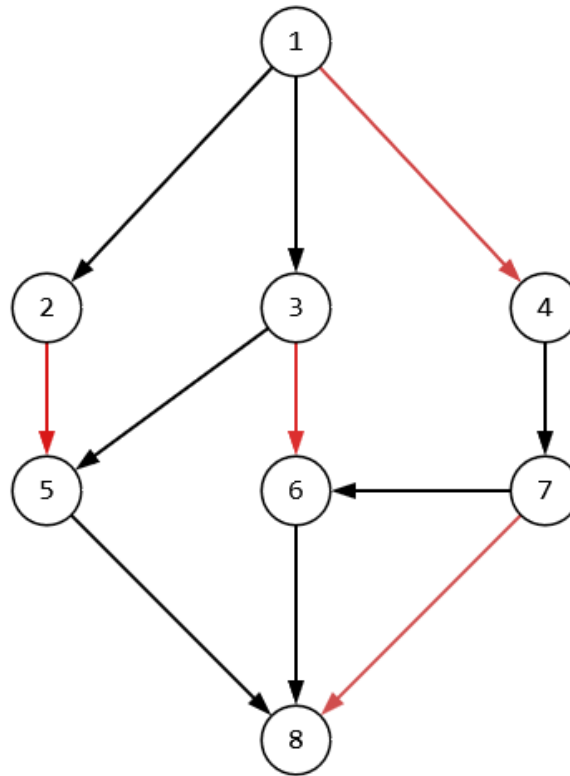
```
int dfs(int u, int Cmin):    // Cmin - пропускная способность в текущем подпотоке
    if u = t
        return Cmin
    visited[u] = true
    for v in u.children
        auto uv = edge(u, v)
        if not visited[v] and uv.f < uv.c
            int delta = dfs(v, min(Cmin, uv.c - uv.f))
            if delta > 0
                uv.f += delta
                uv.backEdge.f -= delta
            return delta
    return 0
```

Оценка по времени

$O(VE)$

Паросочетания

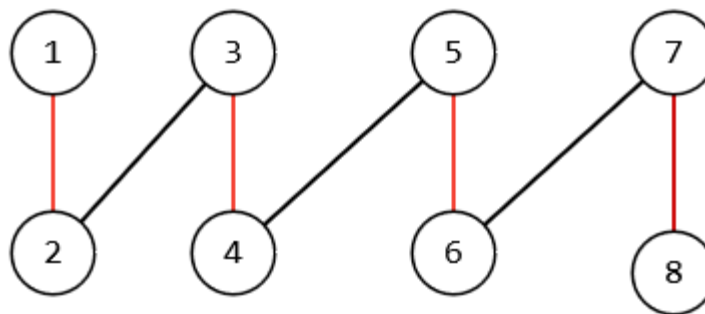
Паросочетание (англ. matching) M — произвольное множество рёбер графа такое, что никакие два ребра не имеют общей вершины.



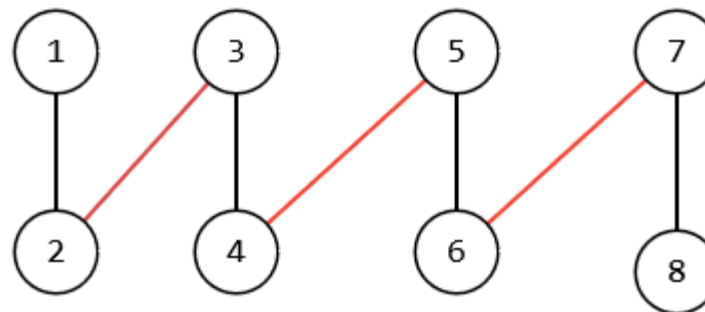
Задача о максимальном паросочетании

Алгоритм Куна

Чередующейся цепью (в двудольном графе, относительно некоторого паросочетания) назовём цепь, в которой рёбра поочередно принадлежат/не принадлежат паросочетанию.



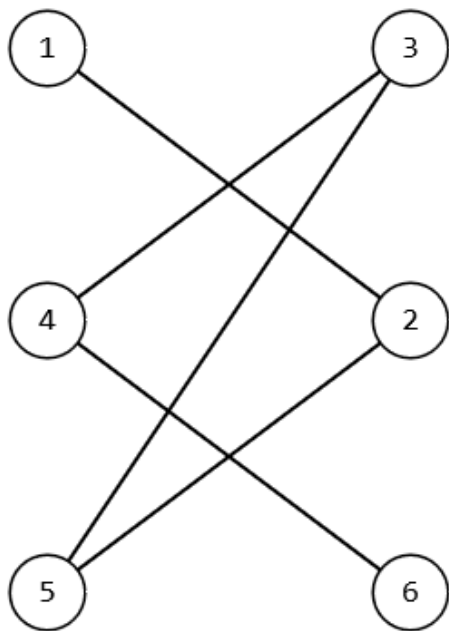
Увеличивающей цепью (в двудольном графе, относительно некоторого паросочетания) назовём чередующуюся цепь, у которой начальная и конечная вершины не принадлежат паросочетанию.



Алгоритм Куна

Алгоритм Куна — сначала возьмём пустое паросочетание, а потом — пока в графе удаётся найти увеличивающую цепь, — будем выполнять чередование паросочетания вдоль этой цепи, и повторять процесс поиска увеличивающей цепи. Как только такую цепь найти не удалось — процесс останавливаем, — текущее паросочетание и есть максимальное.

Проще говоря: Алгоритм Куна просматривает все вершины графа по очереди, запуская из каждой обход, пытающийся найти увеличивающую цепь, начинающуюся в этой вершине.



Алгоритм Куна

```
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])
```

Оценка по времени

$O(VE)$

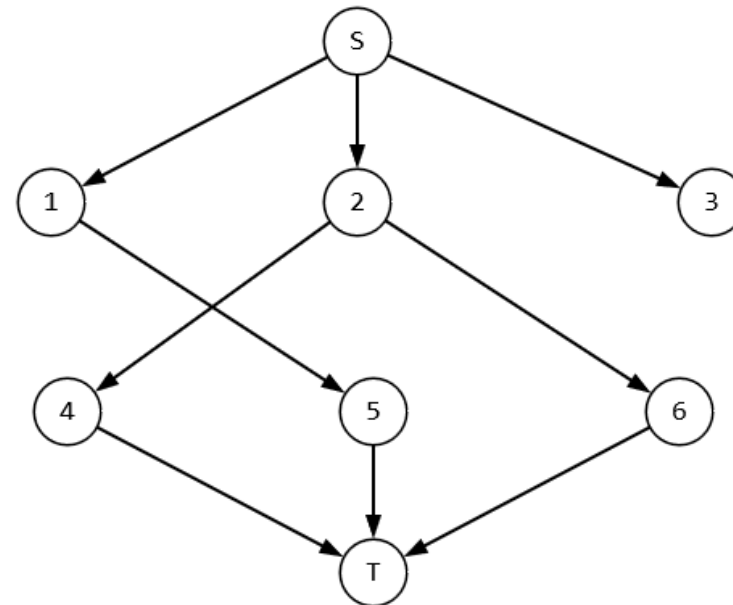
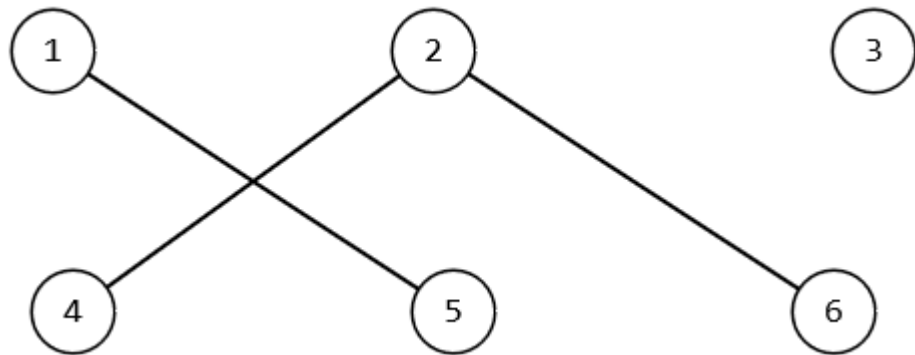
Задача о максимальном паросочетании

Алгоритм Форда-Фалкерсона

Для неориентированного двудольного графа построим относительно изначального, граф такой что:

У нового графа есть исток(все ребра которого ведут в первую долю) и сток(все ребра, входящие в него принадлежат второй доле)

Сделаем ребра ориентированными так, что ребра из истока направлены в первую долю, ребра первой доли направлены во вторую, а ребра второй доли направлены в сток.

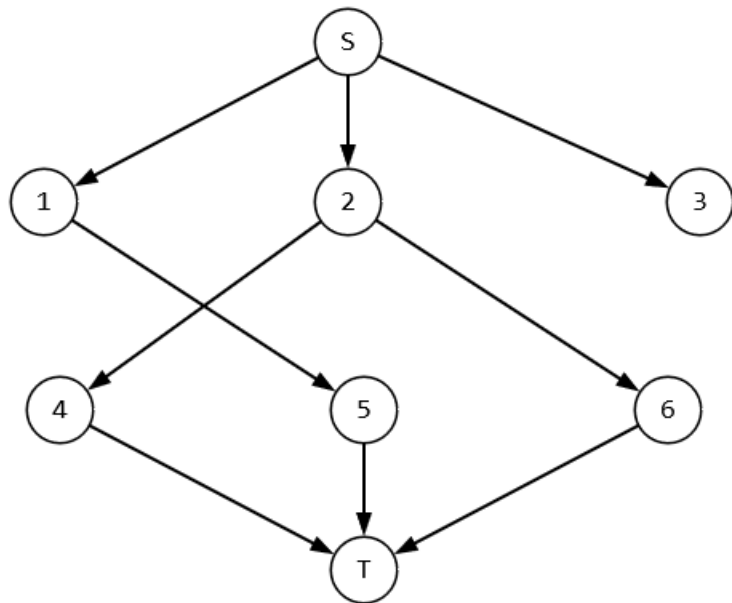


Задача о максимальном паросочетании

Алгоритм Форда-Фалкерсона

Изначально текущее паросочетание пусто. На каждом шаге алгоритма будем поддерживать следующий инвариант: в текущее найденное паросочетание входят те и только те ребра, которые направлены из первой доли во вторую

1. Ищем в графе путь из s в t поиском в глубину.
2. Если путь найден, перезаписываем текущее паросочетание. Далее инвертируем все рёбра на пути и удаляем ребра, принадлежащие текущему паросочетанию.
3. Если путь не был найден, значит текущее паросочетание является максимальным, и алгоритм завершает работу. Иначе переходим к пункту 1.



Задача о максимальном паросочетании

Алгоритм Форда-Фалкерсона

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

```
bool dfs(x):  
    if vis[x]  
        return false  
    vis[x] = true  
    for  $(x, y) \in E$   
        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
```

Оценка по времени
 $O(VE)$