Алгоритмы и структуры данных

Лекция 16
Графы. Поиск кратчайших путей. Потоки. Двудольные графы и паросочетания.

Сергей Леонидович Бабичев

Алгоритм Флойда-Уоршалла.

Алгоритм Флойда-Уоршалла

Построение таблиц маршрутизации.

- Известен с 1962 года.
- Определяет кратчайшие пути во взвешенном графе, описанном матрицей смежности.
- В матрице смежности число, находящееся в i-й строке и j-м столбце есть вес связи между ними.
- Изменим представление и будем полагать, что в матрице смежности $C_{ij} = \infty$, если узлы i и j не являются соседями.
- На входе алгоритм принимает модифицированную матрицу смежности, а на выходе эта матрица будет содержать в элементе C_{ij} вес кратчайшего пути из P_i в P_j.
- Допускается наличие путей с отрицательным весом.
- Не должно быть циклов с отрицательной длиной.

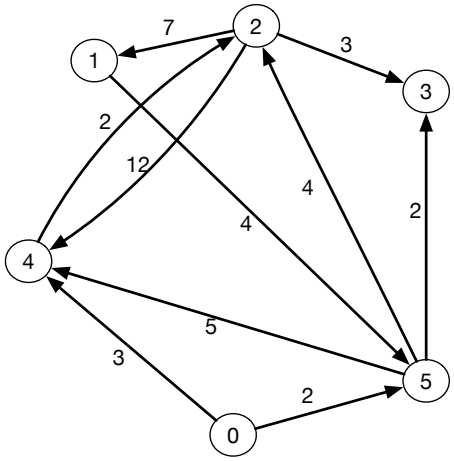
Алгоритм Флойда-Уоршалла

Сам алгоритм может быть описан в рекурсивной форме как

$$D_{ij}^{(k)} = \begin{cases} C_{ij}, & \text{если } k = 0, \\ \min\left(D_{ij}^{(k-1)}, D_{ik}^{(k-1)} + D_{kj}^{(k-1)}\right), & \text{если } k \geqslant 1 \end{cases}$$

Это — задача динамического программирования.

Этапы прохождения алгоритма для графа



Алгоритм Флойда-Уоршалла

Исходная матрица смежности:

$$D^{(0)} = \begin{array}{c|cccccc} & P_0 & P_1 & P_2 & P_3 & P_4 & P_5 \\ \hline P_0 & 0 & \infty & \infty & \infty & 3 & 2 \\ P_1 & \infty & 0 & \infty & \infty & \infty & 4 \\ P_2 & \infty & 7 & 0 & 3 & 12 & \infty \\ P_3 & \infty & \infty & \infty & 0 & \infty & \infty \\ P_4 & \infty & \infty & 2 & \infty & 0 & \infty \\ P_5 & \infty & \infty & 4 & 2 & 5 & 0 \end{array}$$

Начальная матрица $D^{(0)}$ содержит метрики всех наилучших маршрутов единичной длины. Каждая следующая итерация алгоритма добавляет в матрицу $D^{(i+1)}$ элементы, связанные с маршрутами длины i, на единицу большей.

Алгоритм Флойда-Уоршалла

После первой итерации матрицы не изменяются.
После второй итерации получается следующее (красным цветом помечены изменившиеся элементы таблиц):

$D^{(2)} =$

	P_0	P_1	P_2	P_3	P_4	P_5
P_0	0	∞	∞	∞	3	2
P_1	∞	0	∞	∞	∞	4
P_2	∞	7	0	3	12	**11**
P_3	∞	∞	∞	0	∞	∞
P_4	∞	∞	∞	∞	0	∞
P_5	∞	∞	4	2	5	0

Алгоритм Флойда-Уоршалла

$D^{(3)} =$

	P_0	P_1	P_2	P_3	P_4	P_5
P_0	0	∞	∞	∞	3	2
P_1	∞	0	∞	∞	∞	4
P_2	∞	7	0	3	12	11
P_3	∞	∞	∞	0	∞	∞
P_4	∞	∞	∞	∞	0	∞
P_5	∞	**11**	4	2	5	0

Алгоритм Флойда-Уоршалла

Результат четвёртой и пятой итерации совпадает с результатом третьей. Шестая, последняя итерация:

$D^{(6)} =$

	P_0	P_1	P_2	P_3	P_4	P_5
P_0	0	**13**	**6**	**4**	3	2
P_1	∞	0	**8**	**6**	**9**	4
P_2	∞	7	0	3	12	11
P_3	∞	∞	∞	0	∞	∞
P_4	∞	∞	∞	∞	0	∞
P_5	∞	11	4	2	5	0

Поиск максимального потока.

Поиск максимального потока

- Предположим, что мы сидим за рулём автомобиля.
- Дорожная сеть — граф.
- Алгоритм Дейкстры определит, за какое время мы доберёмся до любого пункта назначения.
- Как определить все ли автомобили могут проехать по данному маршруту, или пропускная способность транспортной сети ограничена?
- Москва 9 мая: все хотят попасть в центр на парад, но часть дорог вообще перекрыта, а часть — имеет ограниченную ширину.
- Как узнать максимальное число автомобилей, которые могут проехать в центр за, скажем, один час?
- Требуется найти **максимальный поток** между стартом и финишем, источником и стоком.

Поиск максимального потока

- Толчок: вторая мировая война, Д. Б. Данциг, отдел статистического управления ВВС США.
- Нужна математическая модель, каким образом можно быстро сконцентрировать войска и войсковую инфраструктуру вблизи критических точек на театре военных действий.
- Более общая задача: определения пропускной способности рёбер транспортного графа поставлена им в 1951 году.
- 1955 год: Лестер Форд и Делберт Фалкерсон разработали алгоритм, решающий именно эту задачу.
- Хорошее решение данной задачи критически важно для современных транспортных графов (Москва: более 100000 узлов).

Поиск максимального потока: термины

- **Ёмкость ребра** — максимальное интенсивность потока, проходящего через ребро.
- **Насыщенное ребро** — ребро, по которому проходит максимальный поток.

Поиск максимального потока: алгоритм

Алгоритм ищет максимальны поток в сети из источника (*source*) в сток (*destination*).

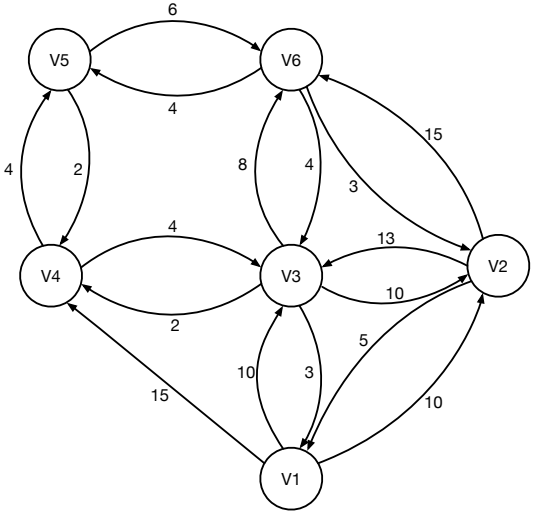
- Каждому ребру ставится в соответствие пара чисел (c, l).
- c — достигнутый до сих пор поток по ребру, вначале он равен нулю, затем это число будет только увеличиваться, пока не достигнет l, ёмкости ребра.
- Если по ребру (u, v) мы пустили прямой поток, пустим такой же в обратном направлении (v, u), добавив, если надо, отсутствующее ребро.
- Алгоритм продолжается, когда на хотя бы одном маршруте из s все рёбра — ненасыщенные, то есть на всех рёбрах $c < l$.

Поиск максимального потока

1. Ищем любой маршрут, содержащий только ненасыщенные рёбра из s в d. Если такого нет, то алгоритм закончен, искомый поток есть сумма потоков всех рёбер, приходящих в d.
2. Мы нашли **дополняющий маршрут**. Определяем значение максимального потока m, который мы можем пропустить по данному маршруту. Он определяется как минимальная из всех возможных разностей ёмкости l и существующего потока c по всем рёбрам маршрута.
3. К каждому из c на маршруте прибавляем m. Хотя бы одно ребро станет насыщенным.
4. Возвращаемся к (1).

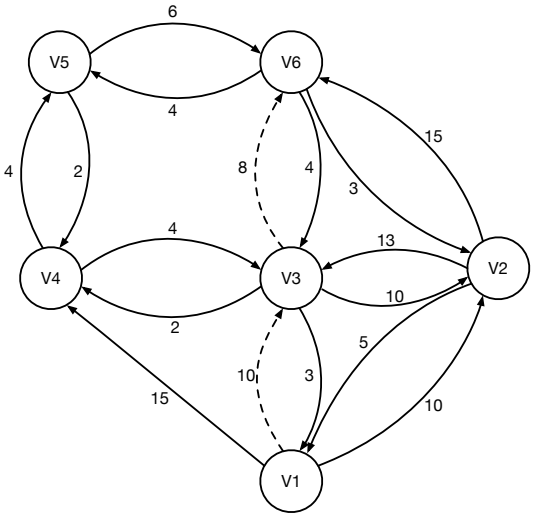
Поиск максимального потока: подопытный граф.

Ищем максимальный поток из V_1 в V_6.



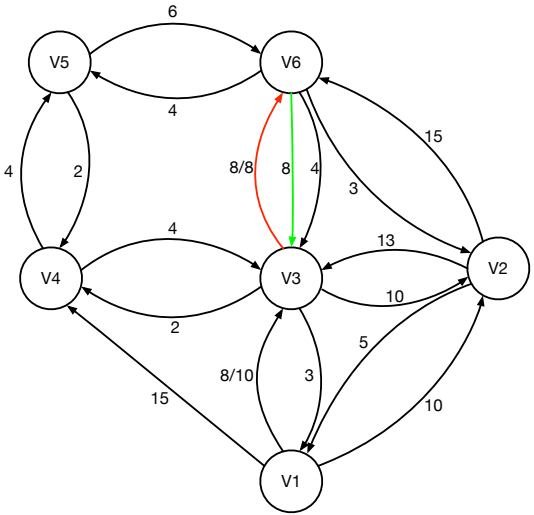
Поиск максимального потока

- Найдём произвольный маршрут из s в d. Пусть $v1 \to v3 \to v6$.
- Наименьшая разница между l и c равна 8, пропускаем поток с интенсивностью 8 по этому маршруту.



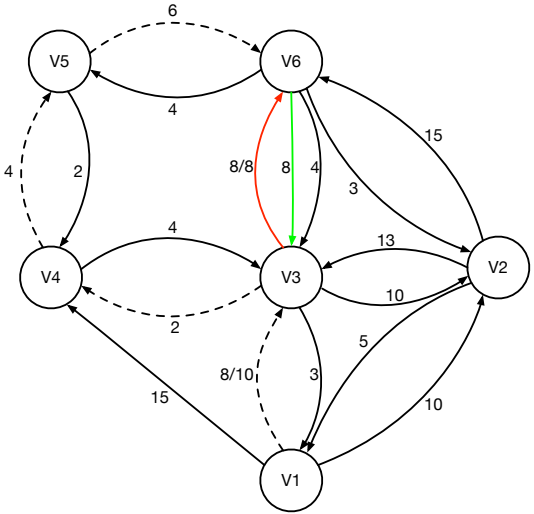
Поиск максимального потока

- Ребро $V_3 \to V_6$ стало насыщенным.
- Добавим обратное ребро $V_6 \to V_3$.



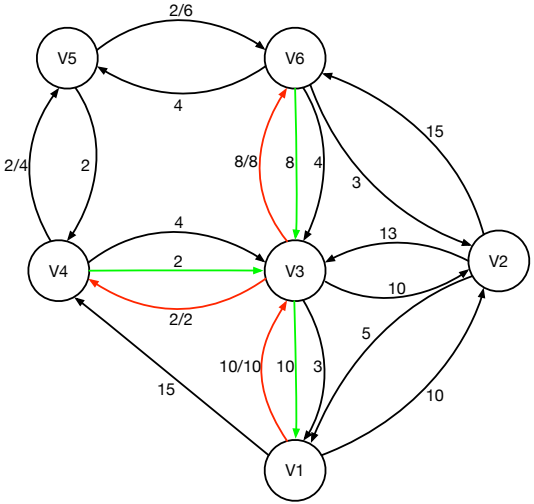
Поиск максимального потока

- Находим новый путь из V_1 в V_6.



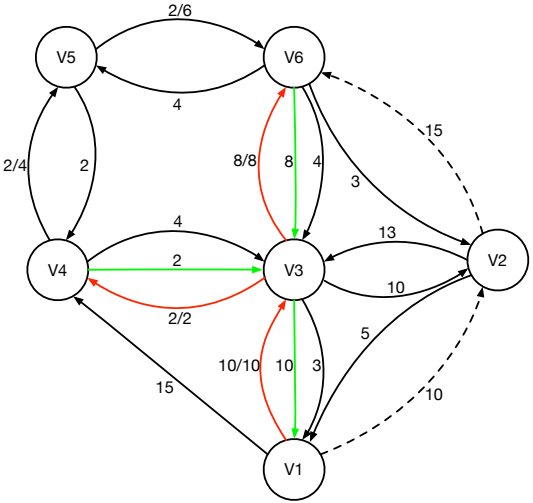
Поиск максимального потока

- Он сделал насыщенными рёбра $V_1 \to V_3$ и $V_3 \to V_4$.
- Добавим обратные рёбра.



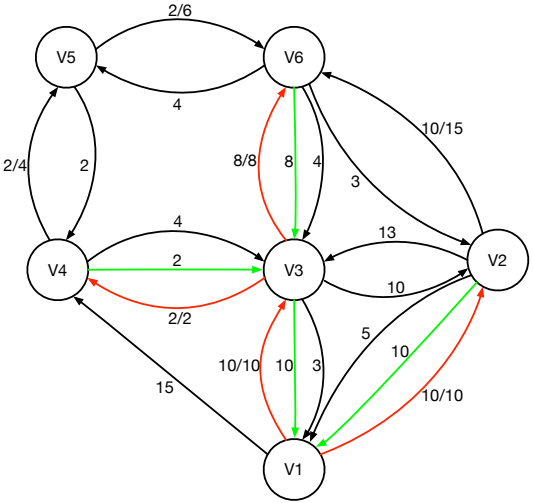
Поиск максимального потока

- Новый поиск дал нам новый путь $V_1 \to V_2 \to V_6$.



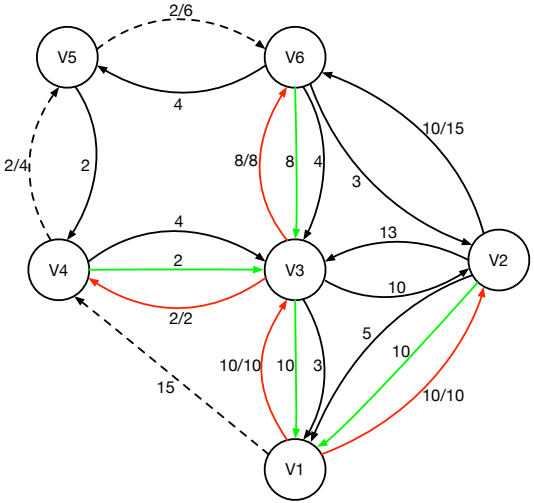
Поиск максимального потока

- Насыщаем рёбра этого пути и добавляем обратные.



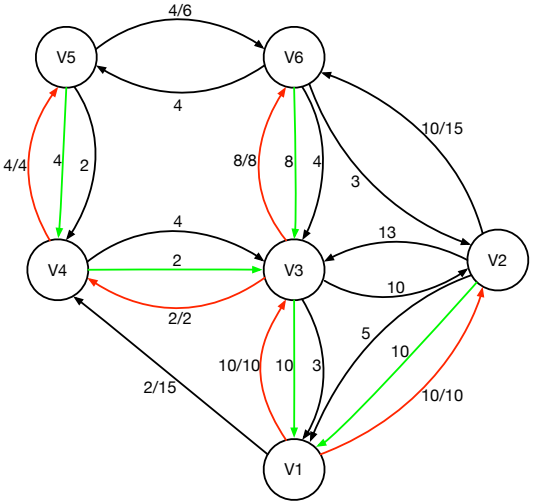
Поиск максимального потока

- Следующий поиск дал нам поток интенсивностью 2.



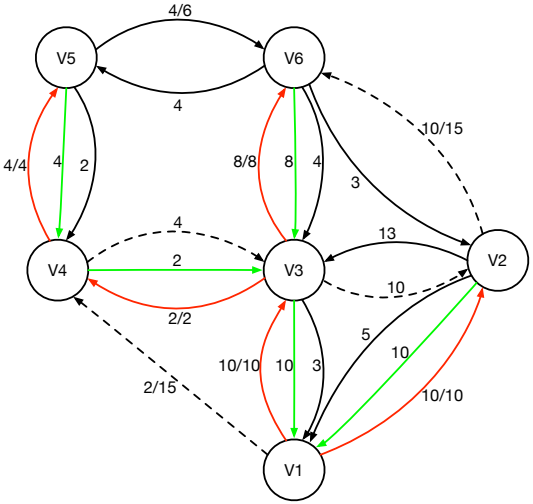
Поиск максимального потока

- Он насытил ребро $V_4 \to V_5$.



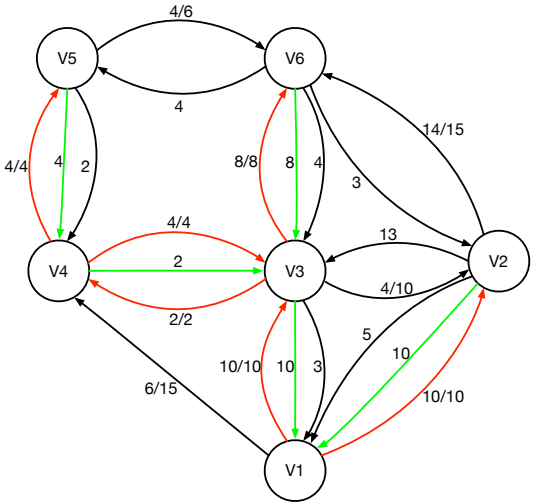
Поиск максимального потока

- Потоки искать всё сложнее, но мы нашли один с интенсивностью 4.



Поиск максимального потока

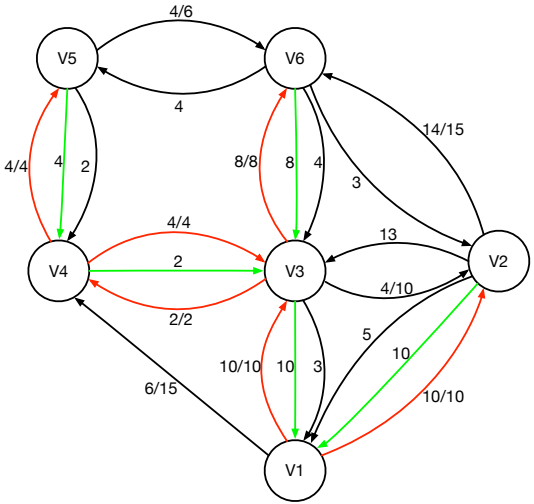
- Мы всё насытили и вот результат:



Максимальный ли это поток? Сейчас он равен 26.

Поиск максимального потока

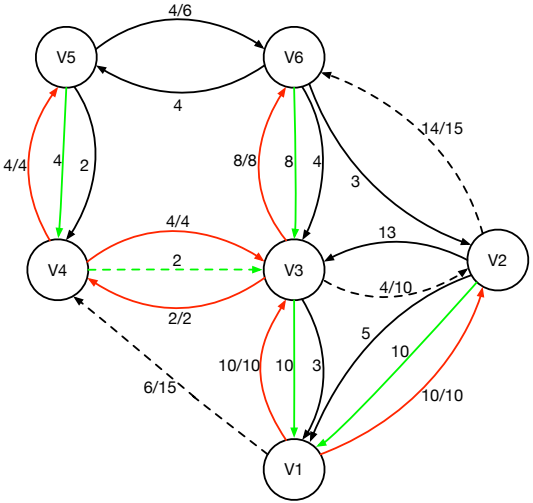
- Мы всё насытили и вот результат:



Максимальный ли это поток? Сейчас он равен 26. Вспомним про обратные рёбра.

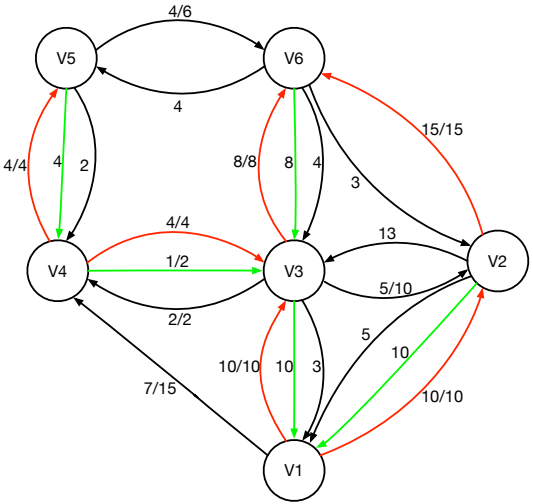
Поиск максимального потока

- Пропустим единицу потока по маршруту $V_1 \to V_4 \to V_3 \to V_2 \to V_6$.



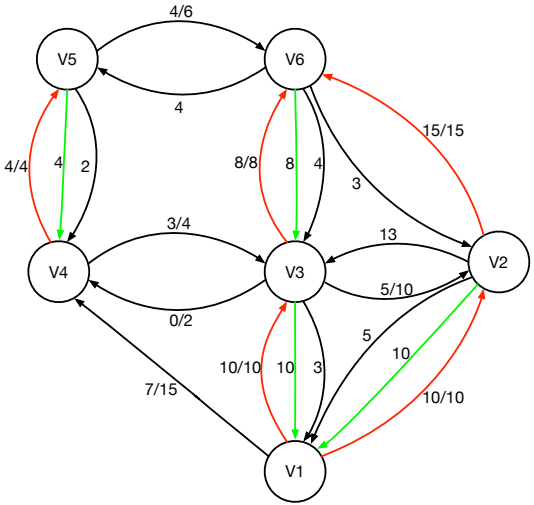
Поиск максимального потока

- Но ведь у нас уже есть две единицы потока из V_3 в V_4.
- Объединим потоки из V_3 в V_4 (пять единиц) и из V_4 в V_3 (две единицы).



Поиск максимального потока

- Больше ничего никуда не добавить — алгоритм завершён.



Из V_1 выходит 27 единиц из 35 возможных, в V_6 приходит 27 из 29 возможных. Что можно расширить, чтобы увеличить пропускную способность?

Планарность

Планарные графы

- Простой (не имеющий петель и кратных ребер) граф называется **планарным**, если его можно изобразить на плоскости без пересечения рёбер. [1].
- **Гранями** планарного графа называются циклы, которые не могут быть разбиты на более мелкие циклы.
- Граф называется **n-регулярным**, если каждая его вершина имеет степень n.

[1] Верно даже более сильное утверждение: планарный граф можно изобразить на плоскости так, что его ребра — непересекающиеся *отрезки*; соответствующее утверждение — *теорема Фари*.

Изоморфизм графов

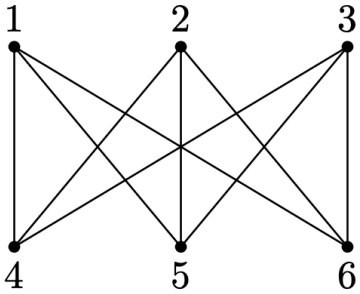
- Граф G **изоморфен** графу H, если существует биективное отображение f вершин графа G на вершины графа H, сохраняющие отношение смежности.
- Отношение изоморфизма графов — отношение эквивалентности.
- Введём отношения R так: два графа находятся в отношении R, если один можно свести к другому в отношении изоморфизма, заменой вершины степени 2 на ребро между смежными ей вершинами или добавление вершины степени 2 на ребро. Тогда отношением **гомеоморфизма** является транзитивное замыкание отношения R.

Формула Эйлера

Для любого связного планарного графа имеет место равенство $|V| - |E| + |F| = 1$, где $|V|$ — число вершин, $|E|$ — число рёбер, а $|F|$ — число граней этого графа.

- **Задача**. Есть три дома и три магазина — продуктовый, аптека и хозяйственный. Можно ли от каждого дома проложить тропинку до каждого из магазинов так, чтобы тропинки не пересекались?

Пусть дома и магазины — это вершины графа, а тропинки — это рёбра. Пусть вершины под номерами 1, 2 и 3 — это дома, а 4, 5 и 6 — магазины. Количество вершин — 6, количество рёбер — 9. Для того чтобы граф был планарным, необходимо, чтобы у него было количество граней, равное $|F| = |E| + 1 - |V| = 9 + 1 - 6 = 4$. Но можно насчитать по крайней мере 5 простых циклов: $1 - 4 - 2 - 5 - 1$; $1 - 4 - 3 - 6 - 1$; $2 - 5 - 3 - 6 - 2$; $1 - 6 - 2 - 4 - 1$; $2 - 6 - 3 - 4 - 2$.



Значит, граф $K_{3,3}$ не является планарным.

- **Задача.** Можно ли между каждыми двумя из 5 городов провести дорогу так, чтобы эти дороги не пересекались?

- **Задача.** Можно ли между каждыми двумя из 5 городов провести дорогу так, чтобы эти дороги не пересекались?
- Нужно проверить планарность графа K_5. Количество вершин $|V| = 5$, рёбер $|E| = C_5^2 = 10$. Грани — простые циклы. В графе не менее 10 простых циклов, образуемых тройками вершин. По формуле Эйлера $|F| = |E| + 1 - |V| = 10 + 1 - 5 = 6 < 10$. Граф не планарный.

Теорема Куратовского-Понтрягина

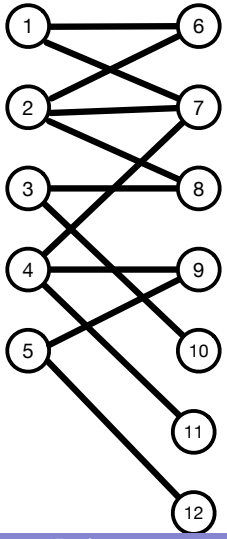
Theorem (Куратовского-Понтрягина)

Граф планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных K_5 и $K_{3,3}$.

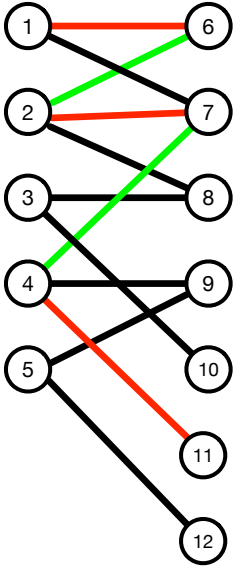
Двудольные графы. Паросочетания.

Двудольные графы

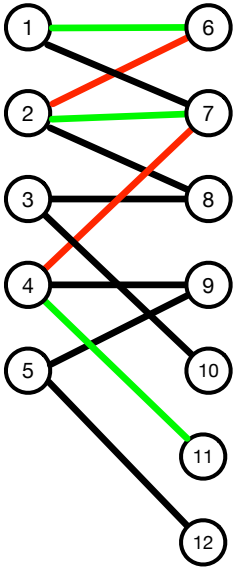
- Граф называется **двудольным**, если все вершины в нём можно разбить на два множества (доли) так, чтобы все рёбра находились между вершинами различных множеств.



- **Теорема о чётных циклах.** Граф является двудольным тогда и только тогда, когда в нём не содержатся циклы нечётной длины.
- **Паросочетание** в двудольном графе — набор попарно несмежных рёбер.
- **Максимальное паросочетание** — наибольшее по множеству паросочетание.
- **Насыщенные вершины** — вершины, принадлежащие ровно одному ребру в паросочетании.
- **Цепь** длины k — простой путь, содержащий ровно k рёбер.
- **Чередующаяся цепь** относительно паросочетания — цепь, в которой рёбра поочерёдно то принадлежат паросочетанию, то не принадлежат.
- **Увеличивающая цепь** относительно паросочетания: чередующаяся цепь, начало и конец которых не принадлежит паросочетанию.



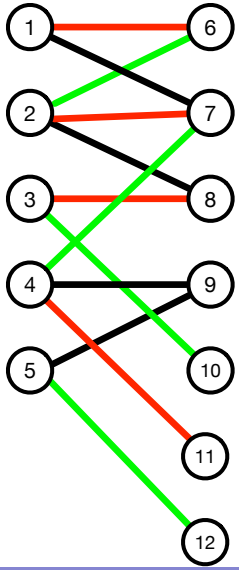
Чередующаяся цепь $1 \to 6 \to 2 \to 7 \to 4 \to 11$ на паросочетании $(2, 6), (4, 7)$. Она же увеличивающая.



Чередующаяся цепь $1 \to 6 \to 2 \to 7 \to 4 \to 11$ на паросочетании $(1,6),(2,7),(4,11)$.

Теорема Бержа

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

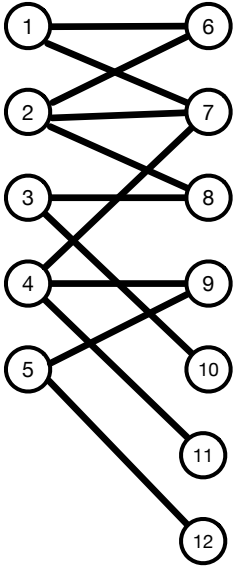


Алгоритм Куна

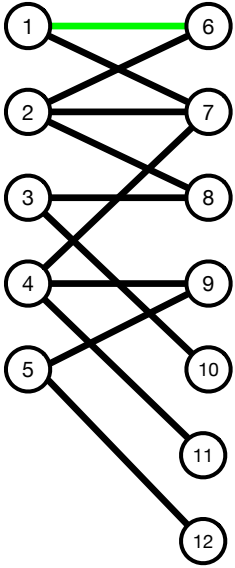
1. Возьмём пустое паросочетание.
2. Добавим ребро. Если невозможно — конец алгоритма.
3. Проверяем, не образовалось ли увеличивающихся цепей.
4. Если образовалось, то производим операцию чередования.
5. Идём к п. 2.

Алгоритм Куна: поиск увеличивающих цепей

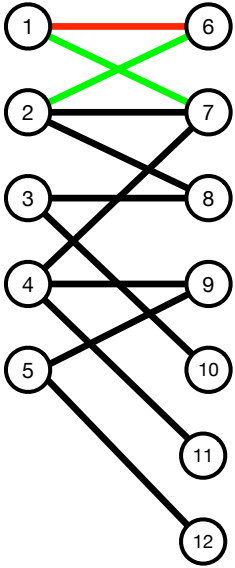
- Ищем ненасыщенную вершину первой доли v.
- От неё запускаем поиск увеличивающей цепи.
  - Если в смежных найдётся ненасыщенная вершина u — увеличивающая сеть найдена, включаем ребро в паросочетания и выходим.
  - Для насыщенной вершины u есть ребро в w. Переходим в него и рекурсивно ищем увеличивающую цепь из w.
  - Если какой-либо из поисков из v нашёл увеличивающую цепь, насыщаем вершину v нужным ребром.



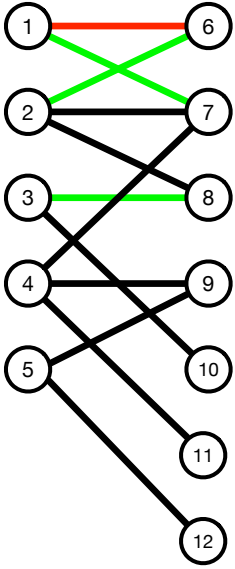
Вершина 1 — ненасыщенная, запускаем обход с неё, выбирая вершину 6, создав ребро и насытив вершину 6.



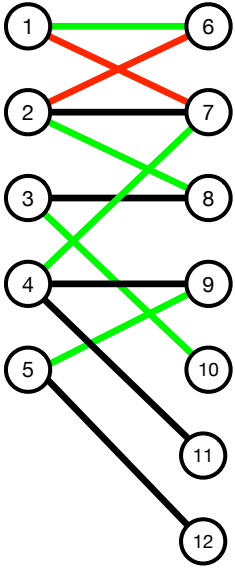
Обход от ненасыщенной вершины 2 приводит к насыщенной вершине 6. У неё есть пара 1, и, если рекурсия с 1 даст увеличение цепи, фиксируем результат.



Начинаем обход с 3. Первый сосед не насыщен — фиксируем новую пару.



Обход с 4 приводит в 7, насыщенную вершину. Рекурсия продолжается от насыщенной 1 и приводит к рекурсивной 2, от которой есть ненасыщенный сосед.


1
6
2
7
3
8
4
9
5
10
11
12