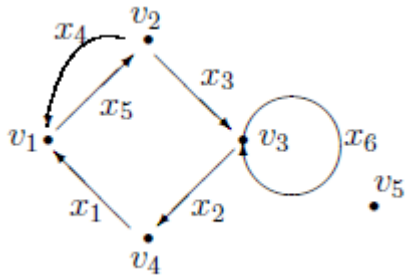
Теория графов

Основные понятия и определения.

Опр. Ориентированный граф $G =< V, X >$ - конечное непустое множество $V \neq \emptyset, V = \{v_1, \dots, v_n\}$ вершин графа и множество $X = \{x_1, \dots x_m\}$ дуг графа, где каждая дуга x_k - упорядоченная пара вершин $x_k =< v_i, v_j >$, v_i - начало дуги, v_j - конец дуги. Дуга x_k исходит из v_i, заходит в v_j. Дуга x_k инцидентна вершинам x_i и x_j. Вершина, которая не имеет инцидентных ей дуг - изолированная. Дуга $< v_i, v_i >$ - петля.

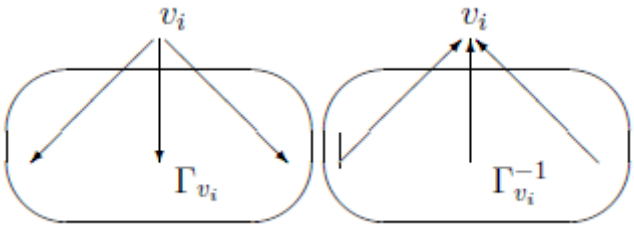


$$\Gamma_{v_i} = \{v_j \mid \exists < v_i, v_j > \in X\}$$

$$\Gamma_{v_2} = \{v_1, v_3\}$$

$$\Gamma_{v_i}^{-1} = \{v_j \mid \exists < v_j, v_i > \in X\}$$

$$\Gamma_{v_1}^{-1} = \{v_2, v_4\}$$



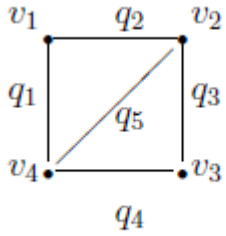
Опр. Последовательность дуг графа, такая что начало следующей дуги совпадает с концом предыдущей дуги - путь. Контур - путь, у которого начало первой дуги совпадает с концом последней (замкнутый путь).

$v_1 \to v_4$ - путь. Описание через дуги: $\{x_5, x_3, x_2\}$, через вершины: $\{v_1, v_2, v_3, v_4\}$.
$v_1 \to v_1$ - контур. Описание через дуги: $\{x_5, x_3, x_2, x_1\}$, через вершины: $\{v_1, v_2, v_3, v_4, v_1\}$.

Опр. Путь (контур) - простой, если все его дуги различны.
Опр. Путь (контур) - элементарный, если все его вершины различны (в контуре - кроме первой и последней).

v_1, v_2, v_1, v_2 - не является ни простым, ни элементарным.

Опр. Неориентированный граф $G=< V, Q >$ - конечное непустое множество $V \neq \emptyset, V = \{v_1, \ldots, v_n\}$ вершин графа и множество $Q = \{q_1, \ldots q_m\}$ ребер графа, где каждое ребро $q_k = (v_i, v_j) = \{v_i, v_j\}$.



Опр. Цепь (маршрут) - последовательность ребер, которую заданием ориентации можно превратить в путь.

Опр. Цикл - замкнутая цепь (введением ориентации можно превратить в контур).
Цепи и циклы аналогично бывают простые и элементарные.

Орграф	**Неор. Граф**
дуга	ребро
путь	цепь (маршрут)
контур	цикл

Граф - иллюстрация бинарных отношений на конечных множествах. Если граф неориентированный, то это симметрическое отношение. Каждому графу соответствует отношение, а каждому отношению соответствует граф.

Матричное задание графов

Способы матричного задания графов:
1. матрица смежности;
2. матрица инцидентности.

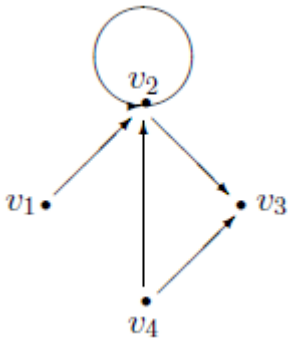
Опр. Матрица смежности ориентированного графа – квадратная матрица порядка n (n - число вершин графа): $A = ||a_{ij}||$ с элементами

$$a_{ij} = \begin{cases} 1, \text{если } \exists < v_i, v_j > \in X \\ 0 \text{ в противном случае} \end{cases}$$

Опр. Матрица смежности неориентированного графа – квадратная (симметрическая) матрица порядка n: $A = ||a_{ij}||$ с элементами

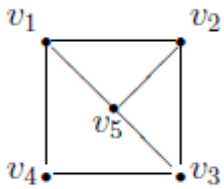
$$a_{ij} = \begin{cases} 1, \text{если } \exists (v_i, v_j) \in Q \\ 0 \text{ в противном случае} \end{cases}$$

Пример 1.



$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Пример 2.



$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

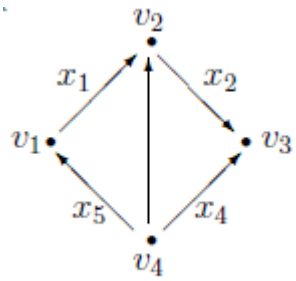
Опр. Матрица инцидентности ориентированного графа – матрица порядка $n \times$ m (n – число вершин графа; m – число его $\frac{\text{ребер}}{\text{дуг}}$) : $B_{n\times m} = \left|\left|b_{ij}\right|\right|$ с элементами

$$b_{ij} = \begin{cases} -1, \text{если дуга } x_j \text{ исходит из } v_i \\ 1, \text{если дуга } x_j \text{ заходит в } v_i \\ 0, \text{если дуга } x_j \text{ не инсцидентна } v_i \end{cases}$$

Опр. Матрица инцидентности неориентированного графа - матрица порядка $n \times$ m: $B_{n\times m} = \left|\left|b_{ij}\right|\right|$ с элементами

$$b_{ij} = \begin{cases} 1, \text{если ребро } q_j \text{ инсцидентно } v_i \\ 0, \text{если ребро } q_j \text{ не инсцидентно } v_i \end{cases}$$

Пример 3.

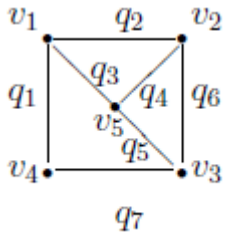


$$B = \begin{pmatrix} & x_1 & x_2 & x_3 & x_4 & x_5 \\ v_1 & -1 & 0 & 0 & 0 & 1 \\ v_2 & 1 & -1 & 1 & 0 & 0 \\ v_3 & 0 & 1 & 0 & 1 & 0 \\ v_4 & 0 & 0 & -1 & -1 & -1 \end{pmatrix}$$

Свойство: сумма строк матрицы инцидентности орграфа равна нулевой строке.

$$rg\ B\ =\ n\ -\ 1$$

Пример 4.



$$B = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Связность в графе

Опр. Неориентированный граф G =< V, Q > - связный, если между любыми его вершинами v_i, v_j существует цепь.

Опр. Компонента связности графа - максимальный связный подграф.

Опр. Матрица связности неориентированного графа – квадратная (симметрическая) матрица порядка n: $S = ||s_{ij}||$ с элементами

$$s_{ij} = \begin{cases} 1, \text{если существует цепь из } v_i \text{ в } v_j \\ 0 \text{ в противном случае} \end{cases}$$

Опр. Ориентированный граф G =< V, X > - односторонне связный, если для любой пары вершин $v_i, v_j (i \neq j)$ существует путь из v_i в v_j либо из v_j в v_i.

Опр. Ориентированный граф G =< V, X > - сильно связный, если для любой пары вершин $v_i, v_j (i \neq j)$ существуют путь и из v_i в v_j и из v_j в v_i.

Аналогично определяются компоненты односторонней и сильной связной.

Опр. Матрица односторонней связности орграфа - квадратная матрица порядка n: $T = ||t_{ij}||$ с элементами

$$t_{ij} = \begin{cases} 1, \text{если существует путь из } v_i \text{ в } v_j \\ 0 \text{ в противном случае} \end{cases}$$

Опр. Матрица сильной связности орграфа - квадратная матрица порядка n: $\bar{S} = \left|\left|\bar{s}_{ij}\right|\right|$ с элементами

$$\bar{s}_{ij} = \begin{cases} 1, \text{если существует путь из } v_i \text{ в } v_j \text{ и из } v_j \text{ в } v_i \\ 0 \text{ в противном случае} \end{cases}$$

Алгоритм нахождения числа путей длины k.

Число путей длины k из вершины v_i в v_j орграфа G =< V, X > с матрицей смежности $A = ||a_{ij}||$ определяет элемент a_{ij}^{k} матрицы $A^{\mathrm{k}} == ||a_{ij}^{\mathrm{k}}||$.

Докажем индукцией по k.

1. Для $\mathrm{k} = 1$ получаем просто матрицу смежности (в первой степени) $A^{\mathrm{k}} == ||a_{ij}^{\mathrm{k}}||$.
2. Предположим, что справедливо: элемент a_{ij}^{k} матрицы $A^{\mathrm{k}} == ||a_{ij}^{\mathrm{k}}||$ определяет число путей длины k из вершины v_i в v_j (обозначим - $\#P(v_i, v_j, k)$).
3. Докажем справедливость предположения для $\mathrm{k} + 1$.
   $\#P(v_i, v_j, k+1) = \#P(v_i, v_1, k))\cdot \#P(v_1, v_j, 1) + \#P(v_i, v_2, k))\cdot \#P(v_2, v_j, 1) + \ldots$
   $+\#P(v_i, v_n, k))\cdot \#P(v_n, v_j, 1) = a_{i1}^{\mathrm{k}} \cdot a_{1j}^{1} + a_{i2}^{\mathrm{k}} \cdot a_{2j}^{1} + \ldots + a_{in}^{\mathrm{k}} \cdot a_{nj}^{1} = a_{ij}^{\mathrm{k+1}}$

Алгоритмы Уоршала нахождения матрицы S связности неориентированного графа по матрице смежности A

Этим же алгоритмом находится и матрица *T* односторонней связности орграфа.
Так как матрицы булевы, введем операции:

$$C \vee D = ||c_{ij} \vee d_{ij}|| \qquad C\&D = ||c_{ij}\&d_{ij}|| \qquad C * D = ||q_{ij}|| \quad q_{ij} = \bigvee_{k=1}^{n} (c_{ik}\&d_{kj})$$

Первый алгоритм Уоршалла.

$$S = E \vee A \vee A^2 \vee \ldots \vee A^{n-1},$$

$A^k = \underbrace{A * A *\ldots* A}_{k}$; A^k содержит дуги, соединяющие путь из k дуг

Для *T* аналогично: $T = E \vee A \vee A^2 \vee \ldots \vee A^{n-1}$.

Множество дуг - это фактически упорядоченные пары, входящие в отношение ρ: $X=\rho$. Отношение можно задавать матрицей смежности соответствующего графа.
$Tr\rho$ - транзитивное замыкание отношения ρ – наименьшее транзитивное отношение, содержащее ρ: $Tr\rho = \rho \cup \rho^2 \cup \ldots \cup \rho^{n-1}$.

Второй алгоритм Уоршалла (итерационный)

$$S^{(0)}, S^{(1)}, \ldots, S^{(n)} = S$$

0) $S^{(0)} = E \vee A$;

…….

$k)$ $S^{(k)} = ||s_{ij}^{(k)}||, \quad s_{ij}^{(k)} = s_{ij}^{(k-1)} \vee (s_{ik}^{(k-1)} \& s_{kj}^{(k-1)})$

……..

$n)$ $S^{(n)} = S$.

Доказательство.

Обоснуем этот алгоритм. Для этого достаточно доказать, что элемент $s_{ij}^{(k)} = 1$ тогда и только тогда, когда существует цепь (путь) из v_i в v_j, проходящая через вершины $\{v_1, \dots, v_k\}$.

Метод индукции по номеру итерации:
1)$s_{ij}^{(0)} = a_{ij} \vee E_{ii}$ - нулевая индукция справедлива, т.к. нет промежуточных вершин.
2) $s_{ij}^{e+1} \Rightarrow$ существует цепь из v_i в v_j, проходящая через $\{v_1, \dots, v_{e+1}\}$.
3) Докажем, что $s_{ij}^{(0)} = 1$, т.к. существует цепь из v_i в v_j, проходящая через $\{v_1, \dots, v_e\}$.

$$s_{ij}^{(e)} = s_{ij}^{(e-1)} \vee (s_{ie}^{(e-1)} \& s_{ej}^{(e-1)}) = 1$$

1. $s_{ij}^{(e-1)} = 1$.

2. $s_{ie}^{(e-1)} = 1 \Rightarrow v_i \to v_e : \underbrace{v_i, v_{k_1}, \dots, v_{k_p}}_{e(v_i, \dots, v_{e-1})}$ □e

$$s_{ej}^{(e-1)} = 1 \Rightarrow v_e \to v_j : \underbrace{v_e, v_{t_1}, \dots, v_{t_p}}_{e(v_e, \dots, v_j)} v_j$$

Соединим два пути: $v_i, v_{k_1}, \dots, v_{k_p}, v_e, v_{t_1}, \dots v_{t_p}, v_j$ - цепь (путь), из v_i в v_j, проходящая через вершины $\{v_1, \dots, v_{e-1}, v_e\}$.
Если существует путь (цепь) из v_i в v_j, проходящий через вершины $\{v_1, \dots, v_{e-1}, v_e\}$, то $s_{ij}^{(e)} = 1$.
Второй алгоритм имеет вычислительную сложность $O(n^3)$, а первый - $O(n^{n-1})$.

$\bar{S} = \left|\left|\bar{s}_{ij}\right|\right|$ - матрица сильной связности вычисляется через матрицу односторонней связности T по формуле:

$$\bar{S} = T \& T^T.$$

Очевидно, что $\bar{S}$ симметрична.

Все дуги контуров графа определяются по формуле

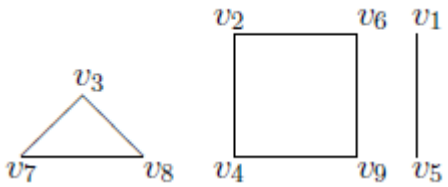
$$K = \bar{S} \& A.$$

Алгоритм компонент связности неориентированного графа по матрице связности S (компонент сильной связности орграфа)

1. В матрице S обнуляем столбцы (можно строки), у которых в первой строке стоят единицы. Получаем матрицу S_1. Соответствующие единиам первой строки номера вершины принадлежат первой компоненте связности, $k = 1$.
2. Если $S_1 \not\equiv (0)$, то $k = k + 1$. Находим не нулевую строку S_1. Пусть ее номер i_1. Соответствующие единиам i_1 строки номера вершин принадлежат второй компоненте связности. В матрице S_1 обнуляем столбцы (строки), у которых в строке i_1 стоят единицы.. Получаем матрицу S_2.

Процесс оканчивается, когда матрица $S_t = (0)$. В этом случае все вершины графа будут принадлежать какой-нибудь компоненте связности.

Пример 1.



$$S = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$k = 1;\ \{v_1, v_5\}$$

$$S_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

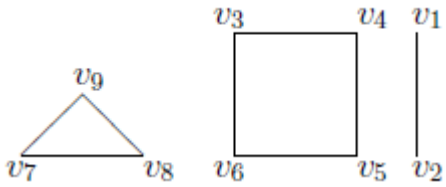
$$k = 2;\ \{v_2, v_4, v_6, v_9\}$$

$$S_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$k = 3;\ \{v_3, v_7, v_8\}$$

$$S_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

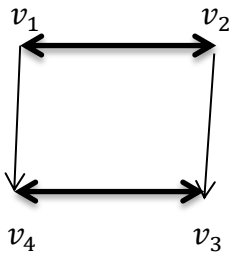
Пример 2. Для данной матрицы связности компоненты связности очевидны.



$$S = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Пример 3. По матрице смежности A найдем:
- матрицу односторонней связности,
- матрицу сильной связности,
- матрицу контуров,
- компоненты сильной связности.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$



$$T = E \vee A \vee A^2 \vee A^3$$

$$T = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix} \vee \begin{pmatrix}0&1&0&1\\1&0&1&0\\0&0&0&1\\0&0&1&0\end{pmatrix} \vee \begin{pmatrix}1&0&1&0\\0&1&0&1\\0&0&1&0\\0&0&0&1\end{pmatrix} \vee \begin{pmatrix}0&1&0&1\\1&0&1&0\\0&0&0&1\\0&0&1&0\end{pmatrix}$$

$$T = \begin{pmatrix}1&1&1&1\\1&1&1&1\\0&0&1&1\\0&0&1&1\end{pmatrix}.$$

$$\bar{S} = T\&T^T = \begin{pmatrix}1&1&1&1\\1&1&1&1\\0&0&1&1\\0&0&1&1\end{pmatrix} \& \begin{pmatrix}1&1&0&0\\1&1&0&0\\1&1&1&1\\1&1&1&1\end{pmatrix} = \begin{pmatrix}1&1&0&0\\1&1&0&0\\0&0&1&1\\0&0&1&1\end{pmatrix}.$$

Следовательно, две компоненты сильной связности:

$$K = \bar{S}\&\mathrm{A} = \begin{pmatrix}1&1&0&0\\1&1&0&0\\0&0&1&1\\0&0&1&1\end{pmatrix} \& \begin{pmatrix}0&1&0&1\\1&0&1&0\\0&0&0&1\\0&0&1&0\end{pmatrix} = \begin{pmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}.$$

Цепи и пути в графах

Длина пути (цепи) равна количеству дуг (ребер), содержащихся в нем, причем считаем столько раз, сколько встречается в пути (цепи).

Опр. Граф называется нагруженным, если каждому ребру (v_i, v_j) (дуге) ставится в соответствие число $l_{ij} \geq 0$ (длина/вес).

$l_{ij} = l(v_i, v_j)$.

Длина пути в нагруженном графе сумме длин дуг, входящих в этот путь: $L = \sum l_{ij}$

Алгоритмы поиска цепей (путей)

1) Алгоритм Тэрри поиска цепи (маршрута) в неориентированном графе;
2) Алгоритм ’фронта волны’ нахождения кратчайшего пути в орграфе;
3) Алгоритм нахождения минимального пути в нагруженном орграфе.

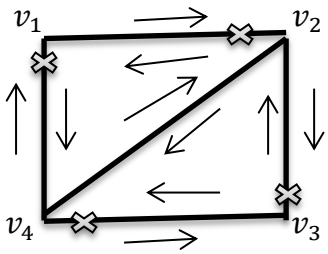
Алгоритм Тэрри

1. Помечаем направление, в котором проходим ребро.
2. По каждому ребру можно идти не более раза в каждом направлении, т.е. не более 2-х раз в разных направлениях.

3. Помечаем ребро q_i, по которому в вершину v_j зашли первый раз.

4. По помеченному ребру q_i можно идти в противоположном направлении, если другой возможности нет.

Пример. Задача о поливальной машине. Полить все улицы и вернуться на базу -v_1.



v_1- v_2- v_3- v_4- v_2- v_4- v_1- v_4- v_3- v_2- v_1.

Построение кратчайшего пути

Опр. Путь из вершины v_i в v_j называется кратчайшим, если он содержит наименьшее количество дуг по сравнению со всеми путями из v_i в v_j.

Будем рассматривать орграф без петель. Найдем кратчайший путь из вершины v_1в вершину v_t.

Алгоритм 'фронта волны'

0) Помечаем вершину v_1 индексом 0; $v_1 \in w_0(v_1)$ – фронт волны нулевого уровня.

1) Помечаем вершины из $\Gamma w_0(v_1) = \Gamma v_1$ единицей. $\Gamma v_1 = w_1(v_1)$

.....

k) Помечаем не помеченные ранее вершины из $\Gamma w_{k-1}(v_1)$ индекс k, они принадлежат $w_k(v_1) \subseteq \Gamma\left(w_{k-1}(v_1)\right)$.

Если через k шагов мы дошли до вершины v_t (до конца), то длина кратчайшего пути равна *k*. Если через *n-1* шаг мы не дошли до v_t, то пути из v_i в v_t не существует.

Предположим, на k шаге мы дошли до вершины v_t. Найдем все вершины кратчайшего пути, начиная с последней v_t.

$$v_1 = v_{i_0}, v_{i_1}, v_{i_2}, \ldots\ldots, v_{i_k} = v_t$$

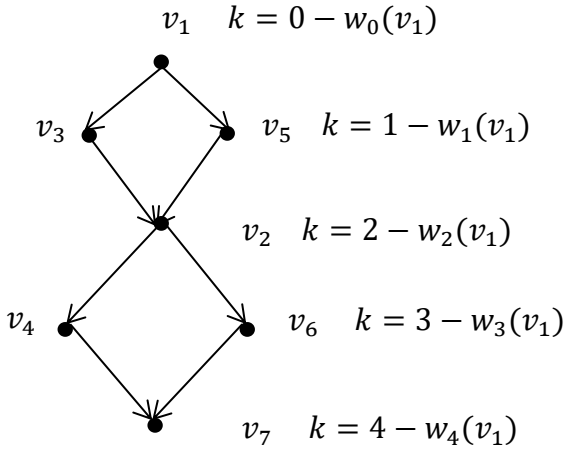
1. $v_t = v_{i_k}$
2. $v_{i_{k-1}} \in w_{k-1}(v_1) \cap \Gamma^{-1} v_{i_k}$
3. $v_{i_{k-2}} \in w_{k-2}(v_1) \cap \Gamma^{-1} v_{i_{k-1}}$

……

k. $v_1 = v_{i_0} \in w_0(v_1) \cap \Gamma^{-1} v_{i_1}$

Пример. Задана матрица смежности орграфа. Най ти кратчайший путь из вершины v_1 в вершину v_7.

$$A=\begin{pmatrix}0&0&1&0&1&0&0\\1&0&0&1&0&1&0\\0&1&0&0&0&0&0\\0&0&0&0&1&0&1\\0&1&0&0&0&0&0\\0&0&1&0&1&0&1\\1&1&0&0&0&0&0\end{pmatrix}$$



Найдем вершины кратчайших путей.

1. $v_{i_4}=v_7$
2. $v_{i_3}\in(w_3(v_1)\cap\Gamma^{-1}v_7)=\{v_6,v_4\}\cap\{v_6,v_4\}=\{v_6,v_4\}$
3. $v_{i_2}\in(w_2(v_1)\cap\Gamma^{-1}v_4)=\{v_2\}\cap\{v_2\}=\{v_2\}$
   $v_{i_2}\in(w_2(v_1)\cap\Gamma^{-1}v_6)=\{v_2\}\cap\{v_2\}=\{v_2\}$
4. $v_{i_1}\in(w_1(v_1)\cap\Gamma^{-1}v_2)=\{v_3,v_5\}\cap\{v_3,v_5,v_7\}=\{v_3,v_5\}$
5. $v_{i_0}\in(w_0(v_1)\cap\Gamma^{-1}v_3)=\{v_1\}\cap\{v_1,v_6\}=\{v_1\}$
   $v_{i_0}\in(w_0(v_1)\cap\Gamma^{-1}v_5)=\{v_1\}\cap\{v_1,v_3,v_6\}=\{v_1\}$

Кратчайшие пути – длина равна 4:

1) v_1,v_3,v_2,v_4,v_7.
2) v_1,v_3,v_2,v_6,v_7.
3) v_1,v_5,v_2,v_4,v_7.
4) v_1,v_5,v_2,v_6,v_7.

Минимальный путь в нагруженном графе

Опр. Нагруженным называется граф, в котором каждой дуге $<v_i,v_j>\in X$ (каждому ребру) ставится в соответствие число $l_{ij}\geq 0$, называемое весом или длинной дуги (ребра).

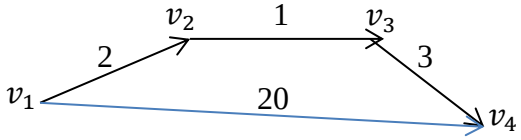
Опр. Матрица весов нагруженного графа - квадратная матрица порядка n (n - число вершин) с элементами $C=||C_{ij}||$.

$$C_{ij}=\begin{cases}l_{ij},\text{если }\exists\text{ дуга }<v_i,v_j>\\ \infty\text{ в противном случае}\end{cases}$$

Опр. Длина пути в нагруженном орграфе - сумма длин его дуг:

$$L=\sum_{\substack{\text{по всем}\\ \text{дугам пути}}} l_{ij}$$

Опр. Путь из v_i в v_j называется минимальным, если его длина наименьшая по сравнению со всеми путями из v_i в v_j.



Кратчайший путь $v_1 - v_4$ содержит одну дугу, её длина $L = 20$.

Минимальный путь содержит три дуги, он не является кратчайшим: $v_1 - v_2 - v_3 - v_4$. Его длина $L = 2 + 1 + 3 = 6$.

Алгоритм нахождения минимального пути в нагруженном графе

$\lambda_i^{(k)}$ – длина минимального пути из вершины v_1 в v_i, содержащего не более k дуг.

$\lambda_i^0, \dots, \lambda_i^{(n-1)}$ i=1,...,n (n-r) вершин

1) Положим $\lambda_i^{(0)} = \infty$ i=2,...,n

   $\lambda_j^{(0)} = 0$, j=1,....,n

   …

   k-1) $\lambda_i^{(k)} = min(\lambda_j^{(k-1)} + C_{ji})$ $1 \le j \le n$

$\lambda_i^{(n-1)}$ - длина минимального пути из v_1 в v_i.

Найдем вершины минимального пути $v_1 = v_{i_0}, v_{i_1}, \dots, v_{i_k} = v_t$:

$$\lambda_{i_{k-1}}^{(k-1)} + C_{i_{k-1}i_k} = \lambda_{i_k}^{(k)}$$

$$\lambda_{i_{k-2}}^{(k-2)} + C_{i_{k-2}i_{k-1}} = \lambda_{i_{k-1}}^{(k-1)}$$

$$\dots\dots$$

$$\lambda_{j_0}^{(0)} + C_{i_0 i_1} = \lambda_{i_1}^{(1)}$$

Индексы по столбцам соответствуют номерам вершин минимального пути.

Может существовать несколько путей, тогда перебором находим все так же, как и в алгоритме «фронта волны».

Пример.

$$\boldsymbol{C} = \begin{pmatrix} \infty & \infty & \mathbf{1} & \mathbf{5} & \mathbf{4} & \mathbf{20} \\ \infty & \infty & \infty & \infty & \infty & \mathbf{2} \\ \infty & \mathbf{7} & \infty & \infty & \mathbf{2} & \infty \\ \infty & \mathbf{3} & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \mathbf{1} & \infty & \mathbf{8} \\ \infty & \infty & \infty & \infty & \mathbf{10} & \infty \end{pmatrix}$$

$\boldsymbol{\lambda_i^{(0)}}$	$\boldsymbol{\lambda_i^{(1)}}$	$\boldsymbol{\lambda_i^{(2)}}$	$\boldsymbol{\lambda_i^{(3)}}$	$\boldsymbol{\lambda_i^{(4)}}$	$\boldsymbol{\lambda_i^{(5)}}$
0	**0**	**0**	**0**	**0**	**0 в v_1**
∞	∞	**8**	**8**	**7**	**7 в v_2**
∞	**1**	**1**	**1**	**1**	**1 в v_3**
∞	**5**	**5**	**4**	**4**	**4 в v_4**
∞	**4**	**3**	**3**	**3**	**3 в v_5**
∞	**20**	**12**	**10**	**10**	**9 в v_6**

Последний столбец таблицы - длины минимальных путей из вершины v_1 во все вершины графа.

Минимальный путь из вершины v_1 в v_6 единственный: $v_1 - v_3 - v_5 - v_4 - v_2 - v_6$.

Восстановил последовательность вершин этого пути, выписав следующие соотношения:

$$\lambda_2^{(4)} + C_{26} = \lambda_6^{(5)}$$

$$\lambda_4^{(3)} + C_{42} = \lambda_2^{(4)}$$

$$\lambda_5^{(2)} + C_{54} = \lambda_4^{(3)}$$

$$\lambda_3^{(1)} + C_{35} = \lambda_5^{(2)}$$

$$\lambda_1^{(0)} + C_{13} = \lambda_3^{(1)}$$

Индексы по столбцам соответствуют номерам вершин минимального пути.

Устойчивые подмножества в графе.

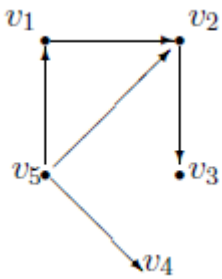
В этом разделе будем рассматривать орграфы без петель

Внутренне устойчивые подмножества.

Опр. Подмножество S вершин графа $G = \langle V, X \rangle$ называется внутренне устойчивое, если для любой вершины $v_i \in S \quad S \cap \Gamma v_i = \emptyset$

Опр. Максимальное внутренне устойчивое подмножество графа - подмножество, не являющееся собственным подмножеством никакого другого внутренне устойчивого подмножества этого графа.

.



Максимальное внутренне устойчивое $\{v_1, v_3, v_4\}$
$\{v_2, v_4\}$
$\{v_3, v_5\}$

Максимально количество вершин в максимальном внутренне устойчивом подмножестве – число внутренней устойчивости.

Метод Магу нахождения внутренне устойчивых подмножеств графа.

Вводим предикаты:

$$v_i = \text{И}\ (1) \Leftrightarrow v_i \epsilon S$$

$$\alpha(v_i, v_j) = \alpha_{ij} = \text{И}\ (1) \Leftrightarrow v_j \epsilon \Gamma v_i (< v_i, v_j > \epsilon X)$$

α_{ij} - элементы матрицы смежности графа (И-1, Л-0).

Из определения внутренне устойчивого подмножества графа следует:

$$\forall v_i, v_j \left(v_j \epsilon \Gamma v_i \text{ или } v_i \epsilon \Gamma v_j\right) \Longrightarrow (v_i \notin S \text{ или } v_j \notin S)$$

Запишем истинную формулу на языке логики предикатов:

$$(\forall v_i)(\forall v_j)\left(\left(\alpha_{ij} \vee \alpha_{ji}\right) \supset \left(\overline{v_i} \vee \overline{v_j}\right)\right) = \text{И}$$

$$\left(\alpha_{ij} \vee \alpha_{ji}\right) \supset \left(\overline{v_i} \vee \overline{v_j}\right) = \neg\left(\alpha_{ij} \vee \alpha_{ji}\right) \vee \left(\overline{v_i} \vee \overline{v_j}\right) = \left(\overline{\alpha_{ij}} \,\&\, \overline{\alpha_{ji}}\right) \vee \left(\overline{v_i} \vee \overline{v_j}\right)$$
$$= \left(\overline{\alpha_{ij}} \vee \overline{v_i} \vee \overline{v_j}\right) \,\&\, \left(\overline{\alpha_{ji}} \vee \overline{v_i} \vee \overline{v_j}\right)$$

В силу конечности множества вершин запишем:

$$F = \mathop{\&}_{i=1}^{n} \mathop{\&}_{j=1}^{n} \left(\overline{\alpha_{ij}} \vee \overline{v_i} \vee \overline{v_j}\right) = \text{И} \qquad (*)$$

Приведем формулу F к сокращенной ДНФ:

$$F = \bigvee_{\substack{\text{по всем конъюнкциям} \\ \text{сокращенной ДНФ}}} (\overline{v}_{i_1} \& \overline{v}_{i_2} \& \ldots \& \overline{v}_{i_k}) = \text{И}$$

Тогда элементарной конъюнкции $(\overline{v}_{i_1} \& \overline{v}_{i_2} \& \ldots \& \overline{v}_{i_k})$ соответствует внутренне устойчивое подмножество $V \backslash \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$.

Покажем, что формула F позволяет найти все **максимальные** внутренне устойчивые подмножества.

Докажем от противного. Пусть внутренне устойчивое подмножество $V \backslash \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ не является максимальным. Тогда можно добавить еще вершину, например v_{i_1} и подмножество $V \backslash \{v_{i_2}, \ldots, v_{i_k}\}$ также будет внутренне устойчивым. Но в этом случае в сокращенной ДНФ будут находится одновременно две конъюнкции:

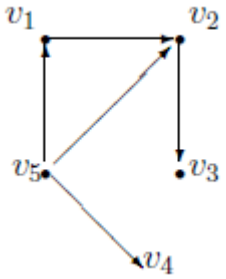
$$\left(\overline{v}_{i_1} \& \overline{v}_{i_2} \& \ldots \& \overline{v}_{i_k}\right) \vee \left(\overline{v}_{i_2} \& \ldots \& \overline{v}_{i_k}\right) \equiv \left(\overline{v}_{i_2} \& \ldots \& \overline{v}_{i_k}\right)$$

Тождество, справедливое по закону поглощения, показывает, что в сокращенной ДНФ может содержаться только одна из этих конъюнкций, соответствующая максимальному внутренне устойчивому подмножеству.

Внешне устойчивые подмножества.

Опр. Подмножество T вершин графа $G =< V, X >, T \subseteq V, V = \{v_1, \ldots, v_n\}$ - внешне устойчивое, если для $\forall v_i \notin T$ выполняется $T \cap \Gamma v_i \neq \emptyset$

Опр. Минимальное внешне устойчивое подмножество графа - внешне устойчивое подмножество такое, что никакое другое внешне устойчивое подмножество не является его собственным подмножеством.



Минимальное внешне устойчивое $\{v_2, v_3, v_4\}$
$\{v_1, v_3, v_4\}$

<u>Метод Магу нахождения внешне устойчивых подмножеств графа.</u>

Вводим предикаты:

$$v_i = \text{И}\ (1) \Leftrightarrow v_i \epsilon S$$

$$\alpha(v_i, v_j) = \alpha_{ij} = \text{И}\ (1) \Leftrightarrow v_j \epsilon \Gamma v_i (< v_i, v_j > \epsilon X)$$

α_{ij} - коэффициенты матрицы смежности графа (И-1, Л-0).

Для нахождения внешне устойчивых подмножеств положим $\alpha_{ii} = \text{И}$

Из определения внешне устойчивого подмножества графа следует:

$$\forall v_i \left(v_i \epsilon T \text{ или } \exists v_j \colon v_j \epsilon T \text{ и } v_j \epsilon \Gamma v_i\right)$$

Запишем истинную формулу на языке логики предикатов:

$$(\forall v_i)\left(v_i \vee (\exists v_j)(\alpha_{ij} \& v_j)\right) = \text{И}$$

В силу конечности множества вершин запишем:

$$F = \&_{i=1}^{n} \vee_{j=1}^{n} \left(\alpha_{ij} \& v_j\right) = \text{И} \qquad (**)$$

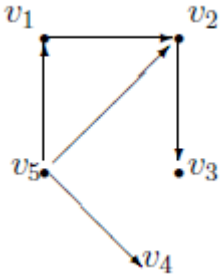
Приведем формулу F к сокращенной ДНФ:

$$F = \bigvee_{\substack{\text{по всем конъюнкциям} \\ \text{сокращенной ДНФ}}} (v_{i_1} \& v_{i_2} \& \ldots \& v_{i_k}) = \text{И}$$

Тогда элементарной конъюнкции $(v_{i_1} \& v_{i_2} \& \ldots \& v_{i_k})$ соответствует внутренне устойчивое подмножество $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$.

Аналогично внутренне устойчивым подмножествам можно показать, что этот метод позволяет найти все **минимальные** внешне устойчивые подмножества.

Пример



$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

1. Найдем максимальные внутренне устойчивые подмножества по формуле $(*)$.

$$F = (\overline{v_1} \vee \overline{v_2})\&(\overline{v_2} \vee \overline{v_3})\&(\overline{v_5} \vee \overline{v_1})\&(\overline{v_5} \vee \overline{v_2})\&(\overline{v_5} \vee \overline{v_4}) = \big(\overline{v_2} \vee (\overline{v_1}\&\overline{v_3})\big)\&$$
$$\&\big(\overline{v_5} \vee (\overline{v_1}\&\overline{v_2}\&\overline{v_4})\big) = (\overline{v_2}\&\overline{v_5}) \vee (\overline{v_1}\&\overline{v_2}\&\overline{v_4}) \vee (\overline{v_1}\&\overline{v_3}\&\overline{v_5}) \vee (\overline{v_1}\&\overline{v_2}\&\overline{v_3}\&\overline{v_4}) =$$
$$= (\overline{v_2}\&\overline{v_5}) \vee (\overline{v_1}\&\overline{v_2}\&\overline{v_4}) \vee (\overline{v_1}\&\overline{v_3}\&\overline{v_5})$$
$$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$$
$$\{v_1, v_3, v_4\} \qquad \{v_3, v_5\} \qquad \{v_2, v_4\}$$

Максимальные внутренне устойчивые подмножества графа

2. Найдем минимальные внешне устойчивые подмножества по формуле $(**)$.

$$A \vee E = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Из каждой строчки выписываем дизъюнкции - все единицы и переменные без отрицания.

$$(v_1 \vee v_2)\&(v_2 \vee v_3)\&v_3\&v_4\&(v_1 \vee v_2 \vee v_4 \vee v_5) = (v_1 \vee v_2)\&v_3\&v_4 =$$

$$= (v_1\&v_3\&v_4) \vee (v_2\&v_3\&v_4)$$
$$\downarrow \qquad\qquad \downarrow$$
$$\{v_1, v_3, v_4\} \qquad \{v_2, v_3, v_4\}$$

Минимальные внешне устойчивые подмножества графа.

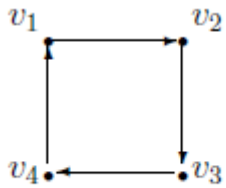
Ядро графа - подмножество $\{v_1, v_3, v_4\}$, являющееся одновременно внутренне и внешне устойчивым.

Опр. Ядром называется подмножество вершин графа N, одновременно являющееся внешне и внутренне устойчивым.

$$\forall v_i \in N \quad N \cap \Gamma v_i = \emptyset$$
$$\forall v_i \notin N \quad N \cap \Gamma v_i \neq \emptyset$$

Граф может не обладать ядром или обладать несколькими ядрами/

Пример.
Граф обладает двумя ядрами.



Внутренне устойчивые подмножества: $\{v_1\},\{v_2\}$

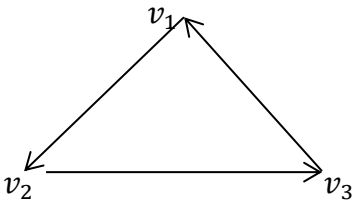
$$\{v_3\},\{v_4\},\{v_1,v_3\}$$

$$\{v_2,v_4\}$$

Внешне устойчивые подмножества: $\{v_1,v_2,v_3,v_4\}$
$$\{v_1,v_3\}$$

$$\{v_2,v_4\}$$

2 ядра: $\{v_2,v_4\}$ и $\{v_1,v_3\}$.
Граф не обладает ядром

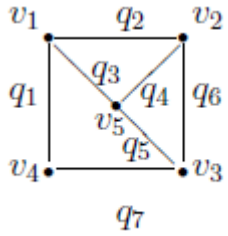


Внутренне устойчивые подмножества: $\{v_1\},\{v_2\},\{v_3\}$.
Внешне устойчивые подмножества: $\{v_1,v_2\},\{v_1,v_3\},\{v_2,v_4\}$

Теорема. Для того чтобы подмножество N⊆ V вершин графа было ядром, необходимо и достаточно, чтобы оно было максимальным внутренне и минимальным внешне устойчивым подмножеством.
Доказательство.
Достаточность очевидна.
Докажем необходимость.
Пусть N не является максимальным внутренне устойчивым. Тогда можно добавить еще одну вершину, например v_i, и всё равно $N' = N \cup \{v_i\}$ будет внутренне устойчивым. Из внешней устойчивости N следует, что $N \cap \Gamma v_i \neq \emptyset$, но $N \subseteq N' \Rightarrow N' \cap \Gamma v_i \neq \emptyset$ что противоречит предположению о внутренней устойчивости N′ .
Предположим теперь, что N не является минимальным внешне устойчивым. Уберем из него тогда хотя бы одну вершину, получим: $N' = N\backslash\{v_i\}$. Из внешней устойчивости N' получим $N' \cap \Gamma_{v_i} \neq \emptyset$, но $N' \subseteq N \Rightarrow N \cap \Gamma v_i \neq \emptyset$, что противоречит внутренней устойчивости N.

Устойчивость в неориентированных графах

Опр. Внутренне устойчивое подмножество графа - *S*, в котором никаких две вершины не смежны. Т- внешне устойчивое подмножество графа *G = (V, Q)*, если $\forall v_i \in V, v_i \notin T\ \exists v_j \in T: (v_i, v_j) \in Q$. *Ядро* - одновременно внутренне и внешне устойчивое подмножество вершин графа.



Максимальные внутренне устойчивые подмножества: $\{v_1, v_3\}$, $\{v_2, v_4\}$,$\{v_4, v_5\}$. Они также являются и минимально внешне устойчивыми, а, следовательно, и ядрами. Но в данном графе есть еще минимальные внешне устойчивые подмножества, не являющиеся внутренне устойчивыми, например $\{v_1, v_4\}$.

Для нахождения максимальных внутренне и минимальных внешне устойчивых подмножеств графа также используется алгоритм Магу: вводится ориентация на ребрах и в ту, и в другую сторону.

Задача.

Как расставить часовых в тюрьме, чтобы их было минимальное количество и все коридоры просматривались?

Найти минимальные внешне устойчивые подмножества графа. если их несколько, выбрать то, которое с наименьшим числом вершин.

Функции на графах. Разбиение графа на уровни.

Рассматриваем орграфы без петель.

Определение. Уровнями $N_0, N_1,..., N_k$ графа $G = (V, X)$ без контуров называются следующие непустые множества вершин графа

$$N_0 = \{v_i \mid v_i \in V, \Gamma v_i = \varnothing\};$$

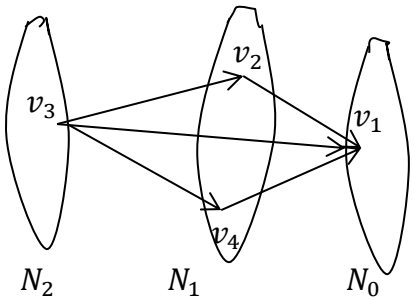
$$N_1 = \{v_i \mid v_i \in V \setminus N_0, \Gamma v_i \subseteq N_0\};$$

……

$$N_k = \{v_i \mid v_i \in V \setminus \bigcup_{j=0}^{k-1} N_j, \Gamma v_i \subseteq \bigcup_{j=0}^{k-1} N_j\}.$$

Для последнего уровня $\Gamma^{-1} N_k = \emptyset$.

Пример.



Уровни $N_0, N_1, \ldots, N_k$ образуют разбиение вершин графа: $V = \bigcup_{j=0}^{k} N_j$

$$N_i \cap N_j = \emptyset,\ i \neq j;\ i, j = 0, \ldots, k$$

Теорема. Если граф не содержит контуров, то его можно разбить на уровни, и обратно, если граф можно разбить на уровни, то он не содержит контуров.

Алгоритм Демукрона разбиения графа без контуров на уровни

0) $l^{O} = \begin{pmatrix} l_1^0 \\ l_2^0 \\ \vdots \\ \vdots \\ l_n^0 \end{pmatrix}$

Где l_i^0 - сумма единиц в i-той строке матрицы смежности. Если l_i^0=0, то вершина $v_i \in N_0$. В этом случае обнуляем i-й столбец матрицы смежности A.

1) $l^1 = \begin{pmatrix} l_1^1 \\ l_2^1 \\ \vdots \\ \vdots \\ l_n^1 \end{pmatrix}$

Где l_i^1 равно сумме единиц в i-той строке матрицы A или $*$, если l_i^0=0.
Если l_i^1=0, то вершина $v_i \in N_1$. В этом случае обнуляем i-й столбец матрицы A.
…….

k) $l^k = \begin{pmatrix} l_1^k \\ l_2^k \\ \vdots \\ \vdots \\ l_n^k \end{pmatrix}$

Где l_i^{k} равно сумме единиц в i-той строке матрицы A или $*$, если $l_i^{\mathrm{k}-1}$=0 или $l_i^{\mathrm{k}-1}$=$*$.
Если l_i^{k}=0, то вершина $v_i \in N_{\mathrm{k}}$. В этом случае обнуляем i-й столбец матрицы A.

Алгоритм заканчивает работу, когда в матрице A все элементы равны нулю.

Пример

Матрица смежности графа имеет вид

$$A=\begin{pmatrix}0&0&0&0&0\\1&0&0&0&0\\1&1&0&0&1\\1&1&1&0&1\\1&1&0&0&0\end{pmatrix}$$

$l^{(0)}$	$l^{(1)}$	$l^{(2)}$	$l^{(3)}$	$l^{(4)}$
0	*	*	*	*
1	0	*	*	*
2	2	1	0	*
3	3	2	1	0
1	1	0	*	*
$v_1 \in N_0$	$v_2 \in N_1$	$v_5 \in N_2$	$v_3 \in N_3$	$v_4 \in N_4$

Используя алгоритм Демукрона, разобьем граф на уровни. Согласно алгоритму, просуммируем элементы каждой строки матрицы смежности. Получим вектор $l^{(0)}$. .

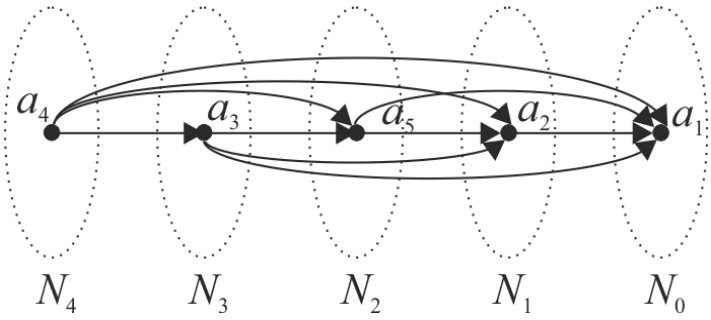
$v_1 \in N_0$

Обнулим первый столбец матрицы A

$A=\begin{pmatrix}0&0&0&0&0\\0&0&0&0&0\\0&1&0&0&1\\0&1&1&0&1\\0&1&0&0&0\end{pmatrix}$. Найдем вектор $l^{(1)}$, компоненты которого равны сумме единиц каждой строки полученной матрицы/ Элемент $l_1^1 = *$, так как вершина $v_1 \in N_0$. Элемент $l_2^0 = 0$, следовательно, вершина $v_2 \in N_1$.

Аналогично обнуляем соответствующие столбцы матрицы A, а затем вычисляем последовательно векторы l^2, l^3, l^4.

$$A=\begin{pmatrix}0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&1\\0&0&1&0&1\\0&0&0&0&0\end{pmatrix},\quad A=\begin{pmatrix}0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&1&0&0\\0&0&0&0&0\end{pmatrix},\quad A=\begin{pmatrix}0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{pmatrix}.$$



Функции на графах

Опр. Порядковая функция графа – функция, определенная на множестве вершин со значениями во множестве $N_0(N \cup \{0\})$, которая ставит в соответствие каждой вершине номер уровня графа:

$$O: V \to N \cup \{0\}$$

Порядковая функция существует только для графов без контуров.

Опр. Функция Гранди орграфа – функция, определенная на множестве вершин со значениями в множестве N ∪ {0}

$$g: V \to N \cup \{0\}$$

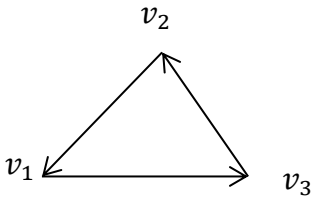
определяемая следующим образом:

$g(v_i)$ =min из $\{N \cup \{0\} \backslash g_(v_j) | v_j \in \Gamma_{v_i}\}$

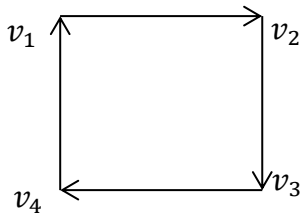
$g(v_i) = 0$, если $\Gamma_{v_i} = \emptyset$.

Функция Гранди может не существовать в графе, имеющем контуры нечетной длины.

У графа с контурами может быть несколько различных функций Гранди.



Функции Гранди не существует.



Две функции Гранди: <0,1,0,1> и <1,0,1,0>.

Теорема. Для графов без контуров существует, и притом единственная, функция Гранди.

Доказательство конструктивное, то есть приведем алгоритм построения функции Гранди.

Граф без контуров можно разбить на уровни $N_0, \ldots, N_k$.

0) Функция Гранди для вершин уровня N_0 по определению равна нулю: $g(v_i) = 0$
1) Для вершин уровня N_1: $g(v_i) = 1$

...

k) Дуги, исходящие из вершин уровня N_k, по определению заходят в вершины уровней

$N_0, \ldots, N_{k-1}$, для которых значения функции Гранди уже определены. Тогда для вершины $v_i \in N_k$ - строим функцию по определению функции Гранди..

Теорема. Вершины, в которых функция Гранди равна нулю, являются ядром графа.
Доказательство:
Пусть $N = \{v_i | g(v_i) = 0\}$.
Вершины, в которых функция Гранди равна нулю, по определению не могут быть смежными, следовательно, $N -$ внутренне устойчивое множество:

$$g(v_i) = 0 \text{ и } g(v_j) = 0 \Rightarrow \nexists < v_i, v_j > \text{ и } \nexists < v_j, v_i >.$$

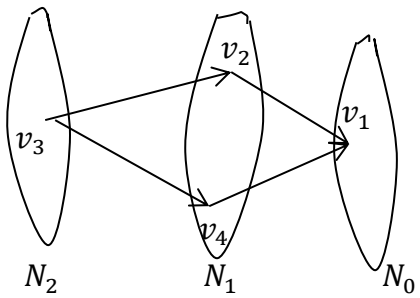
Если $\exists v_i \notin N$, то $\exists < v_i, v_j >$ - дуга графа, причем $v_j \in N$, $g(v_j) = 0$ (по определению функции Гранди). Следовательно, $N -$внешне устойчивое множество.

Утверждение. Для транзитивного графа без контуров порядковая функция совпадает с функцией Гранди.
Доказательство. Для вершин уровней N_0 и N_1 значения этих двух функций совпадают. Из вершины уровня N_2 дуга исходит в вершину уровня N_1, а из вершины N_1 в вершину уровня N_0. Следовательно, из вершины уровня N_2 дуга также исходит в вешину уровня N_0 (из транзитивности). В этом случае по определению значения функции Гранди в вершинах уровня N_2 равно двум и, следовательно, совпадает со значением порядковой функции. Аналогично доказывается для всех следующих уровней.

Утверждение. Ядро транзитивного графа без контуров полностью принадлежит уровню N_0.
Доказательство следует из предыдущих теоремы и утверждения.

Пример.

1.

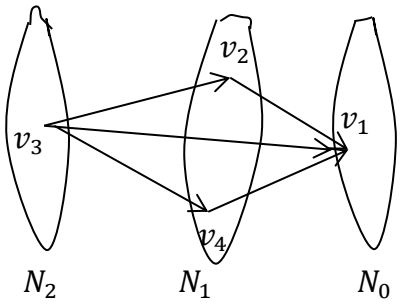


Порядковая функция: $O(v_1) = 0; O(v_2) = 1; O(v_4) = 1; O(v_3) = 2.$

Функция Гранди: $g(v_1) = 0; g(v_2) = 1; g(v_4) = 1; g(v_3) = 0.$

Порядковая функция и функция Гранди не совпадают

2.Граф транзитивный порядковая функция совпадает с функцией Гранди



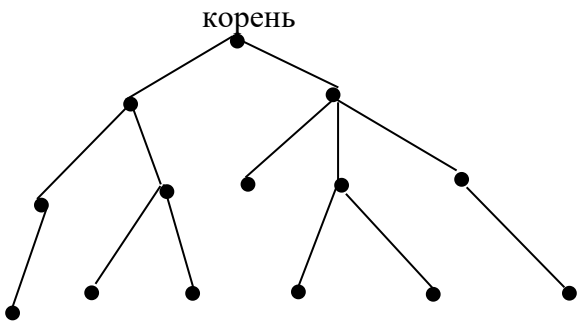
Порядковая функция: $O(v_1) = 0; O(v_2) = 1; O(v_4) = 1; O(v_3) = 2.$

Функция Гранди: $g(v_1) = 0; g(v_2) = 1; g(v_4) = 1; g(v_3) = 2.$

Ядро $-\{v_1\}$ полностью принадлежит уровню N_0.

Деревья и циклы

Опр. 1. Дерево - связный граф без циклов.



Опр. 2. Дерево - связный граф, у которого число вершин на единицу больше числа ребер.

Опр. 3. Дерево – граф, любые две вершины которого соединены единственной простой цепью.

Опр. 4. Дерево - граф без циклов, но при добавлении любого одного ребра получим ровно один простой цикл.

Вершина, инцидентная только одному ребру - висячая.

Докажем эквивалентность определений 1 и 3.

1 ⇒ 3: Пусть существует цикл $v_i, v_{i_1}, \dots, v_{i_k}, v_j, \dots, v_{j_k}, v_i$. Тогда вершины v_i и v_j соединяют две цепи: $v_i, v_{i_1}, \dots, v_{i_k}, v_j$ и $v_j, \dots, v_{j_k}, v_i$.

3 ⇒ 1: Пусть существуют две цепи ⇒ строим цикл (см. выше)

Опр. Остовным деревом графа $G = (V, Q)$ называется подграф $D = (V, \bar{Q})$, содержащий все вершины графа G и являющийся деревом.

$G = (V, Q)$ $D = (V, \bar{Q})$

<u>Алгоритм нахождения остовного дерева связного графа</u>

1) $D_1 = (v_{i_1}, \emptyset)$
2) $D_2 = (\{v_{i_1}, v_{i_2}\}, \{v_{i_1}, v_{i_2}\})$ – добавляем к D_1 вершину v_{i_2}, смежную с v_{i_1}, и соединяем их ребром.

…

k) $D_k = D_{k-1} + \{v_{i_k}\} + \{v_{i_j}, v_{i_k}\}, j = 1, \ldots, k-1$ – добавляем новую вершину v_{i_k}, смежную хотя бы с одной из ранее выбранных и соответствующую дугу.

Продолжаем процесс, пока ни выберем все *n* вершин графа.

Обоснование: 1) так как каждый раз новая вершина связана с одной из предыдущих - граф связный, 2) без циклов, так как каждая новая вершина висячая.

У графа может быть несколько остовных деревьев.

Рассмотрим нагруженный неориентированный граф $G = (V, Q)$, в котором каждому ребру (v_i, v_j) ставится в соответствие вес $l_{ij} \geq 0$.

Опр. Остовным деревом минимального веса называется остовное дерево с наименьшей суммой весов его ребер.

$$L(D) = l_i j \rightarrow min$$

Алгоритм нахождения остовного дерева минимальной длины(веса):

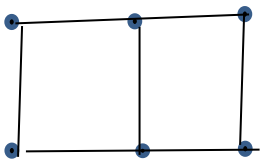
1)Все вершины l

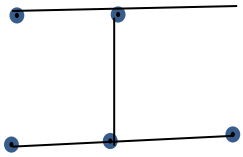
2)Добавляем ребра с минимальным весом, чтобы не было циклов.

3)Если уже дерево, процесс окончен, иначе пункт 2.

Обоснование алгоритма. Предположим, что существует остовное дерево S меньшей длины, чем D: $L(S) < L(D)$. В D найдем ребро $q_{d_1} \in D,\ q_{d_1} \notin S$. Добавим его в S, получим ровно один простой цикл. Удалим из этого цикла ребро $q_{s_1} \in S,\ q_{s_1} \notin D$. Получим дерево S_1. Так как по построению вес $l(q_{d_1}) < l(q_{s_1})$, то $L(S) \geq L(S_1)$. И т.д. Заменим все ребра дерева S на ребра дерева D. Получим цепочку $L(S) \geq L(S_1) \ldots \geq L(D)$. Противоречит предположению: $L(S) < L(D)$.

У графа может быть несколько остовных деревьев минимальной длины.





<u>Цикломатическое число графа. Базис циклов.</u>

Опр. Цикломатическое число графа

<u>Алгоритм нахождения базиса цикловсвязного графа</u> **<u>G = (V, Q) :</u>**

1) Строим остовное дерево графа с $n-1$ ребром (вершин - n)

$$q_{i_1}, \dots, q_{i_{n-1}}$$

2) Добавляем по одному из оставшихся ребер $q_{i_n}, \dots, q_{i_m}$ к остовному дереву и получаем ровно один простой цикл, который и берем в базис циклов.

Обоснование. 1. Число циклов, построенных по алгоритму, равно

$$m-(n-1) = m-n+1 = \gamma(G)$$

– хроматическое число графа. Из теоремы следует, что столько циклов в базисе.

2.Все циклы независимые, т.к. имеют ребро, которого нет в других циклах.

$$D + q_{i_n} = \mu_1$$

<u>Законы Киргофа для напряжений.</u>

<u>Законы Киргофа для токов BI=0</u>

Запишем матрицу инциденций :

$$\begin{pmatrix} & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 \\ v_1 & -8 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ v_2 & 0 & -1 & 1 & 0 & 0 & 0 & 1 & 0 \\ v_3 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 1 \\ v_4 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ v_5 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & -1 \end{pmatrix}$$

$$rgB = n-1$$

$$\begin{cases} -I_1 + I_2 - I_1 = 0 \\ -I_2 + I_3 + I_7 = 0 \\ -I_3 + I_4 + I_8 = 0 \\ I_1 - I_4 + I_5 = 0 \\ -I_5 + I_6 - I_7 - I_8 = 0 \end{cases}$$

Транспортные сети

Опр. Транспортной сетью называется орграф $G = (V, X)$, $V = \{v_1, \ldots, v_n\}$, для которого выполняется:

1) ∃ единственная вершина v_1 (источник): $\Gamma^{-1}v_1 = \emptyset$;

2) ∃ единственная вершина v_n (сток): $\Gamma v_n = \emptyset$;

3) для каждой дуги $< v_i, v_j > \epsilon X$ задана пропускная способность $c(v_i, v_j) = c_{ij} \geq 0$.

Опр. Функцией потока (потоком в транспортной сети G) называется функция $\varphi: X \to R^+ \cup \{0\}$, удовлетворяющая следующим условиям:

1) для каждой дуги $< v_i, v_j > \epsilon X$ выполняется $0 \leq \varphi(v_i, v_j) \leq c_{ij} \ \ \forall i, j = 1, .., n$;

2) для любой промежуточной вершины u:

$$\sum_{v \epsilon \Gamma^{-1} \mathrm{u}} \varphi(v, \mathrm{u}) = \sum_{v \epsilon \Gamma \mathrm{u}} \varphi(\mathrm{u}, v)$$

Опр. Величина потока равна сумме потоков по всем дугам, входящим в сток.

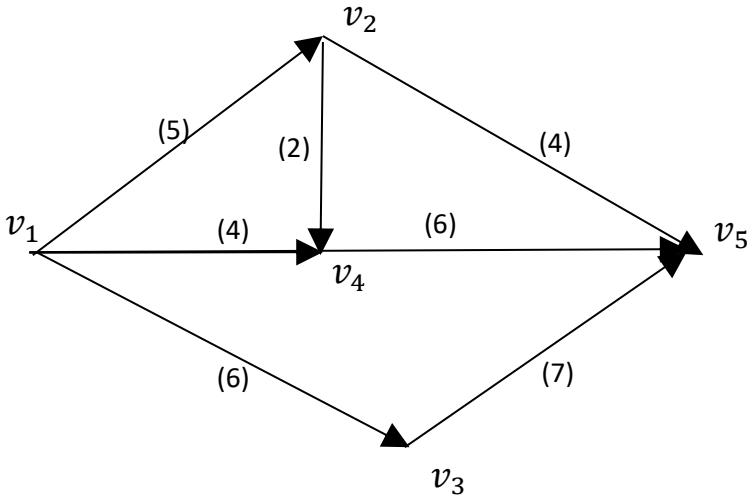
$$\Phi = \sum_{v \epsilon \Gamma^{-1} v_n} \varphi(v, v_n)$$

Эта величина также равна сумме потоков по всем дугам, исходящим из источника

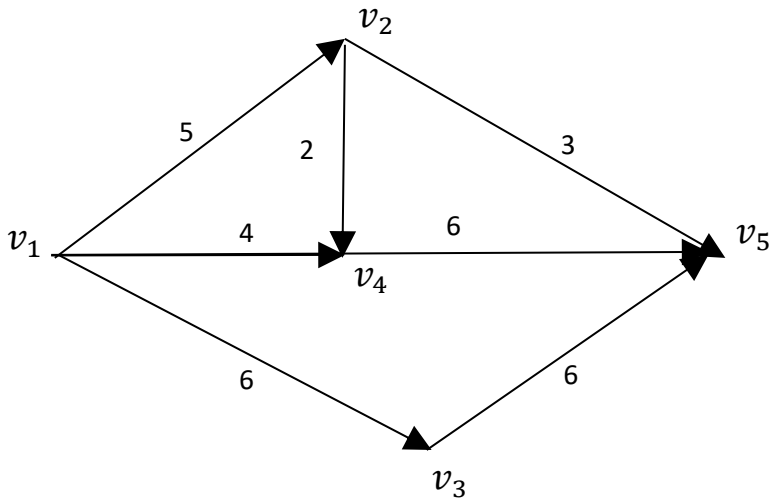
$$\Phi = \sum_{v \epsilon \Gamma v_1} \varphi(v_1, v)$$

Опр. Дуга называется насыщенная, если значение функции потока по ней равно пропускной способности.

Пример.



Транспортная сеть



Функция потока

Дуги $< v_1, v_2 >, < v_1, v_4 >, < v_1, v_6 >, < v_2, v_4 >, < v_4, v_5 >$ - насыщенные.

Опр. Поток называется полным, если любой путь из источника в сток содержит хотя бы одну насыщенную дугу.

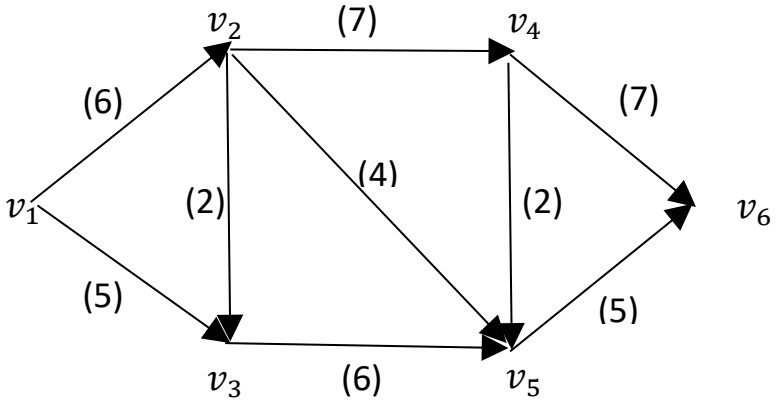
Обычно полный поток ищут как приближение к максимальному, в частном случае они могут совпасть, но в общем случае полный используется как начальный для построения максимального.

Опр. Поток называется максимальным, если значение величины потока наибольшее по сравнению со всеми потоками в данной транспортной сети.

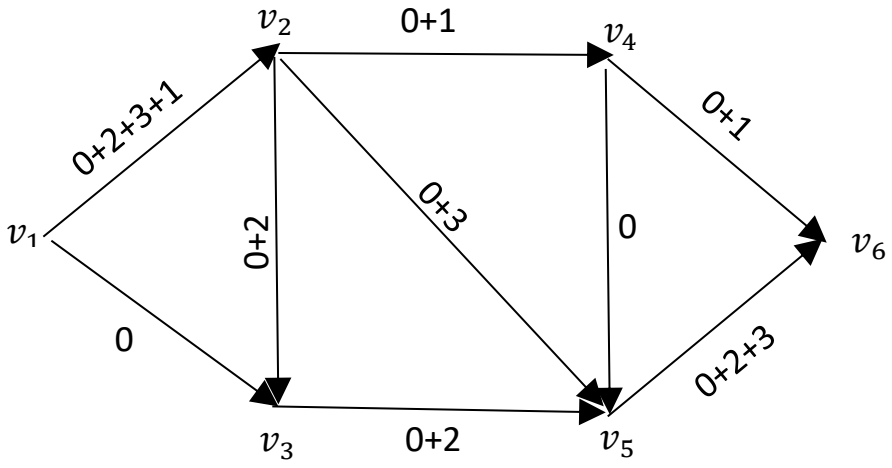
<u>Алгоритм построения полного потока:</u>

0) Выбираем нулевой поток в качестве начального $\varphi_{ij} = 0 \quad \forall i, j.$
1) Проверяем, является ли построенный поток полным, т.е. существует ли путь из $v_1 \to v_n,$ не содержащий насыщенных дуг. Если такого пути нет ⇒ полный поток построен, если есть, то п.2.
2) Вдоль пути, не содержащего насыщенных дуг, увеличиваем поток на одну и ту же величину до тех пор, пока хотя бы одна дуга не станет насыщенной. Переходим к п.1.

Пример. Найти полный поток по данной транспортной сети



Транспортная сеть



Полный поток

Ищем пути, не содержащие насыщенных дуг

1. V_1 - V_2 - V_3 - V_5 - V_6

min {(6), (2), (6), (5)} = 2;

2. V_1 - V_2 - V_5 - V_6

min{(6-2), (4), (5-2)} = 3;

3. V_1 - V_2 - V_4 - V_6

min{(6-5), (7), (7)} = 1;

Величина полного потока $\Phi_{\text{пол}} = 1 + 5 = 6$

Для построения максимального потока введем понятие увеличивающей цепи.

Опр. Увеличивающей цепью называется последовательность вершин транспортной сети из $v_1 \to v_n$:

$$v_1 = \mathrm{u}_1, \mathrm{u}_2, \dots, \mathrm{u}_k = v_n\ , \qquad (*)$$

где $< \mathrm{u}_j, \mathrm{u}_{(j+1)} > \epsilon X$, либо $< \mathrm{u}_{(j+1)}, \mathrm{u}_j > \epsilon X$, причем для каждой пары вершин цепи $< \mathrm{u}_j, \mathrm{u}_{(j+1)} >$ определена положительная величина

$$\Delta_{j(j+1)} = \begin{cases} c_{j(j+1)} - \varphi_{j(j+1)}, & \text{если} < \mathrm{u}_j, \mathrm{u}_{(j+1)} > \epsilon X \\ \varphi_{(j+1)j}. & \text{если} < \mathrm{u}_{(j+1)}, \mathrm{u}_j > \epsilon X \end{cases}$$

Для увеличивающей цепи найдем величину

$$\Delta = \min_{j=1,\dots,k-1} \Delta_{j(j+1)}$$

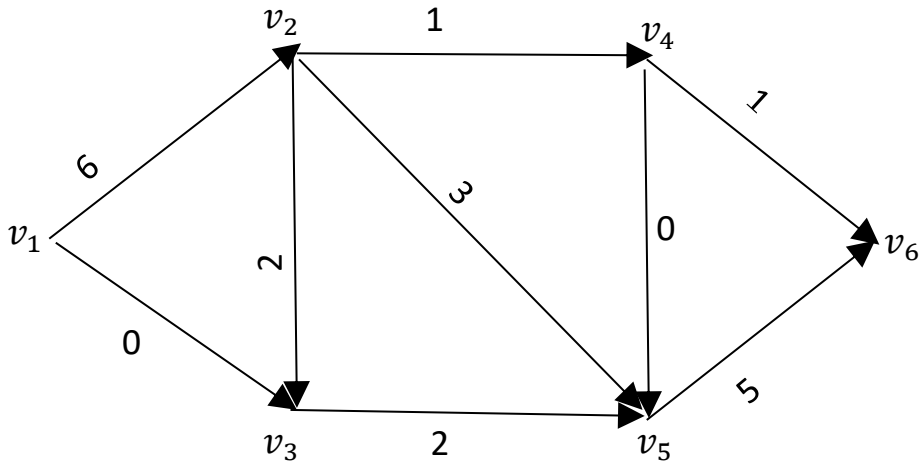
<u>Алгоритм поиска максимального потока в транспортной сети:</u>

0) Выбираем полный поток в качестве начального (можно любой).
1) Проверяем, является ли построенный поток максимальным, т.е. существует ли увеличивающая цепь из $v_1 \to v_n$. Если такой цепи нет ⇒ максимальный поток построен, если есть, то п.2.

2) Вдоль увеличивающей цепи изменяем поток на величину Δ, причем если идем по дуге , то увеличиваем на Δ, если против направления дуги уменьшаем на Δ. Переходим к п.1.

Пример. Найти максимальный поток по данной транспортной сети

Возьмём в качестве начального полный поток, построенный в предыдущем примере.



Найдем увеличивающие цепи

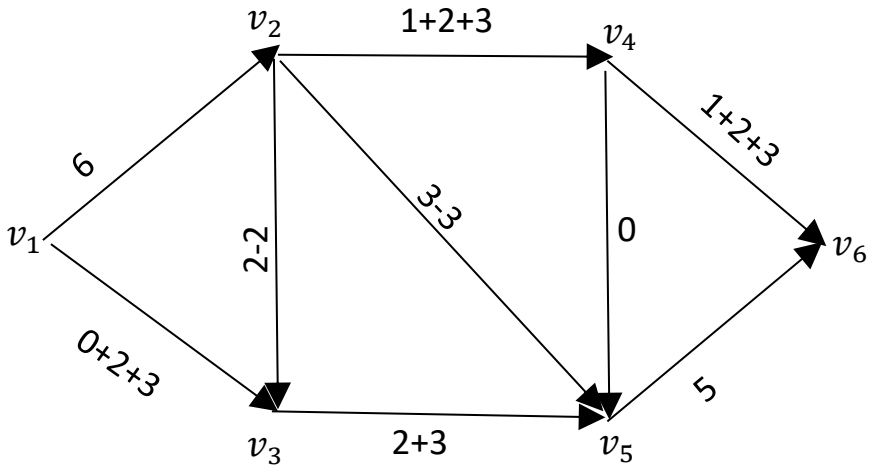
1. $V_1 - V_3 - V_2 - V_4 - V_6$

$\Delta_1 = \min \{ 5_+ , \ 2_- , \ 6_+ , \ 6_+\} = 2;$

2. $V_1 - V_3 - V_5 - V_2 - V_4 - V_6$

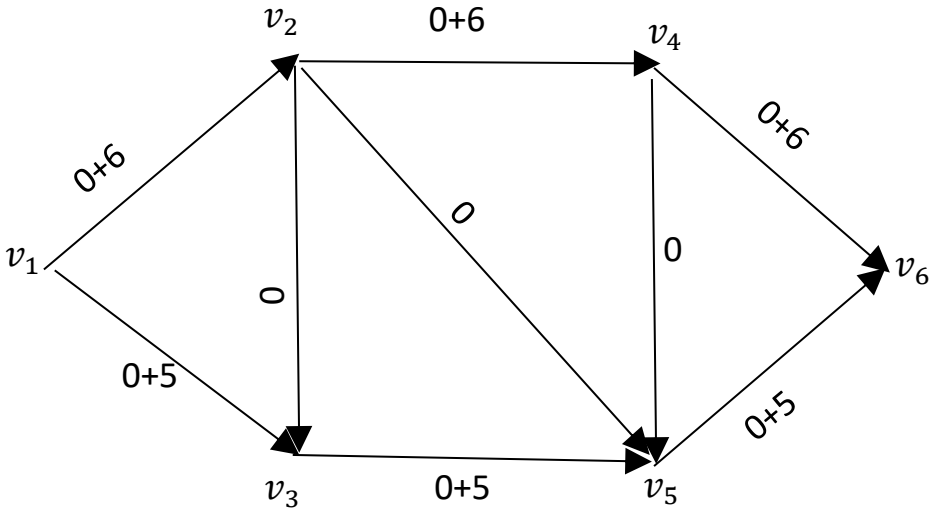
$\Delta_2 = \min \{(5-2)_+ , (6-2)_+ , 3_- , (7-3)_+ , (7-3)_+\} = 3;$

3. Больше цепей нет. $\Phi_{\text{макс}} = 6 + 5 = 6 + 5 = 11$



Максимальный поток

Построим снова полный поток так, чтобы его величина совпала с величиной максимального потока.



1. V_1 - V_2 - V_4 - V_6 min { 6 , 7 , 7 } = 6;
2. V_1 - V_3 - V_5 - V_6 min { 5 , 6 , 5 } = 5;

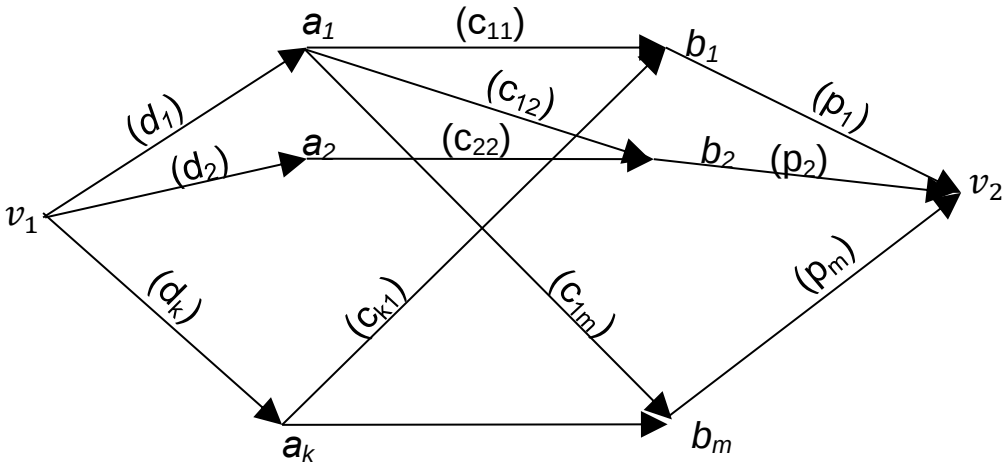
$Фполн = Фмакс = 11$

Задача о портовых перевозках

Имеются портовые города $a_1, a_2, \dots, a_k$ с запасом некоторого товара $d_1, d_2, \dots, d_k$, и города $b_1, b_2, \dots, b_m$ с потребностью в этом товаре $p_1, p_2, \dots, p_m$. Из города a_i в b можно провести не более c_{ij} товара ($i=1,\dots,k$; $j=1,\dots,m$).

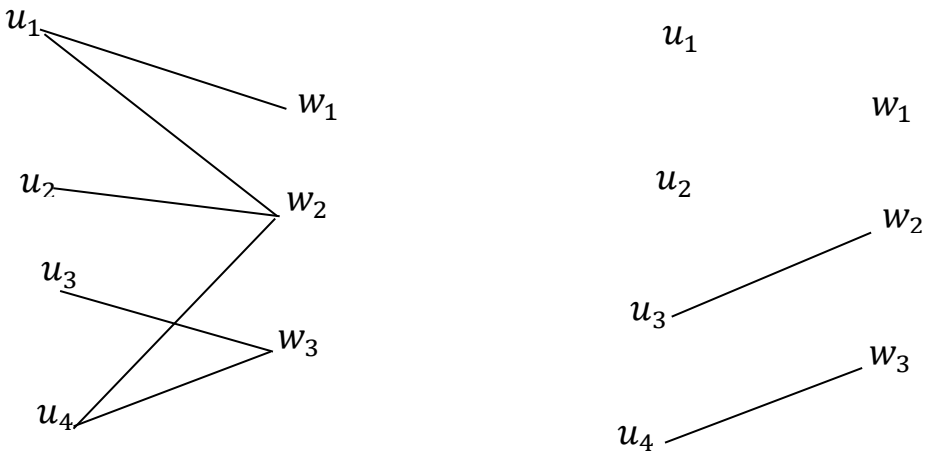
Как максимально обеспечить города $b_1, \dots, b_m$ товаром?

Для решения логистической задачи построим транспортную сеть с вершинами $\{a_1, a, \dots, a_k, b_1, b, \dots, b_m, v_1, v_2\}$, где $v_1 -$ источник, $v_2 -$ сток. Найдем максимальный поток по транспортной сети. Его значения по дугам соответствуют количеству перевозимого товара.



Двудольный граф. Паросочетания.

Опр. Двудольным графом называется неориентированный граф $G = (U \cup W, Q)$ с множеством вершин $U \cup W$ и множеством ребер Q, причем ребро $(u, w) \in Q \Leftrightarrow u \in U, w \in W$.



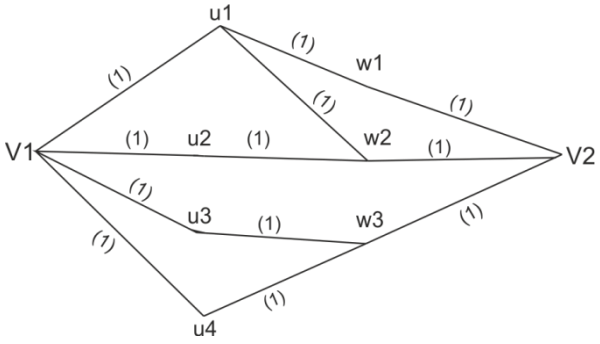
Опр. Паросочетание - подмножество рёбер двудольного графа такое, что никакие два ребра не инцидентны одной вершине графа.

Опр. Максимальное паросочетание - паросочетание с наибольшим числом ребер среди всех паросочетаний в данном графе.

Для нахождения максимального паросочетания построим максимальный поток в следующей транспортной сети.

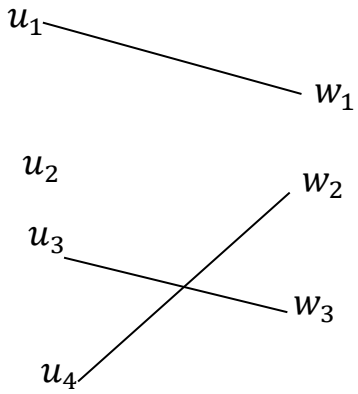
Транспортная сеть: $G_{\text{тр}} = (V, X)$, где $V = \{v_1, v_2\} \cup U \cup W$ $X = \{< u_i, w_j >,$ если $(u_i, w_j) \in Q\} \cup \{< v_1, u_i >\} \cup \{< w_j, v_2 >\}$ для всех $u_i \in U, w_j \in W$.

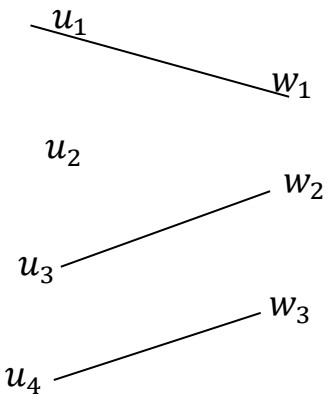
Пропускные способности всех дуг положим равными единице.



Построим максимальный поток. Рёбра графа $G = (U \cup W, Q)$, соответствующие дугам с функцией потока равной единицам, входят в максимальное паросочетание.

Число ребер в максимальном паросочетании равно трем (величина максимального потока), но находится оно неоднозначно, например:

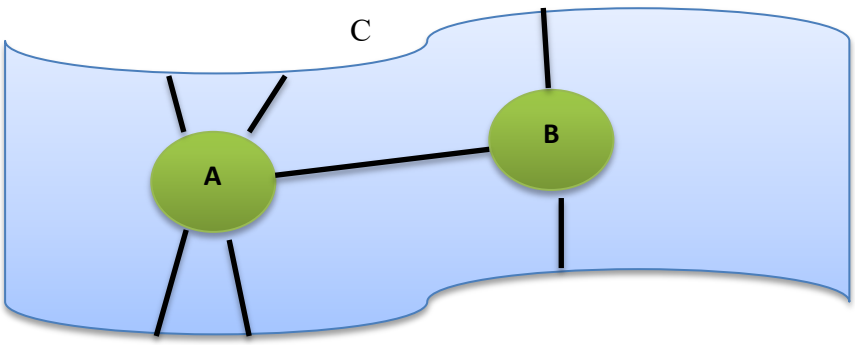




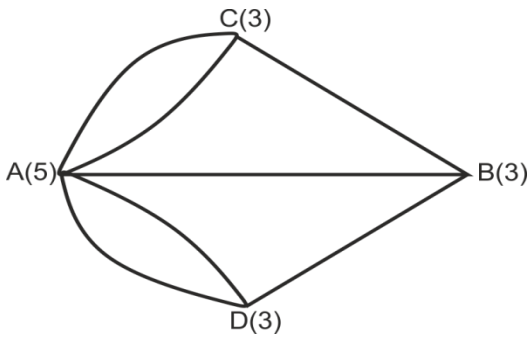
Эйлеровы и Гамильтоновы пути в графе.

Эйлеровы пути

В городе Кёнигсберге протекала река. На ней было два острова, соединенные мостами.



Прогуливаясь по мостам, Эйлер подумал: можно ли обойти все мосты и ровно по одному разу? Нарисовал граф и, обобщив, доказал теорему.

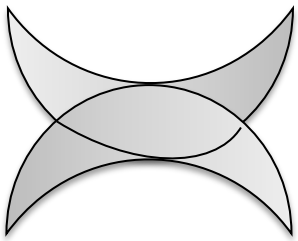


Опр. Эйлеров путь (цикл) – это путь (цикл), содержащий все ребра графа, причем ровно по одному разу.

Опр. Степень вершины - число инцидентных ей ребер (дуг).

Теорема. Эйлеров путь существует тогда и только тогда, когда число вершин с нечетной степенью 0 или 2. Эйлеров цикл существует тогда и только тогда, когда вершин с нечетной степенью нет.
У графа, представленного на рисунке, четыре вершины и все с нечетной степени. Следовательно, обойти все мосты по одному разу нельзя.

Сабли Магомеда. Можно ли обвести контур по одному разу, не отрывая руки? (Да)



Гамильтоновы пути

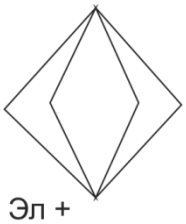
Опр. Гамильтонов путь (цикл) – это путь (цикл), содержащий все вершины графа, причем ровно один раз. Если цикл, то кроме последней вершины.

Задача. Коммивояжер должен проехать все города побывав в каждом ровно один раз и пройдя минимальный путь.
В этой задачи среди всех Гамильтоновых путей выбирают путь с минимальной длиной.

Примеры.

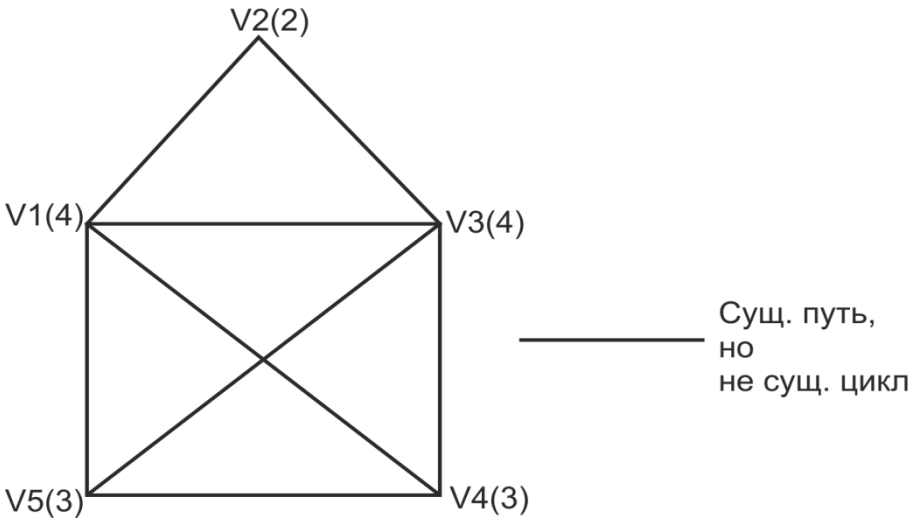
Эл -
Г +

Существует Гамильтонов путь и цикл, но не существует Эйлеровых.



Эл +
Г -

Существует Эйлеровы путь и цикл, но Гамильтоновых путей нет.

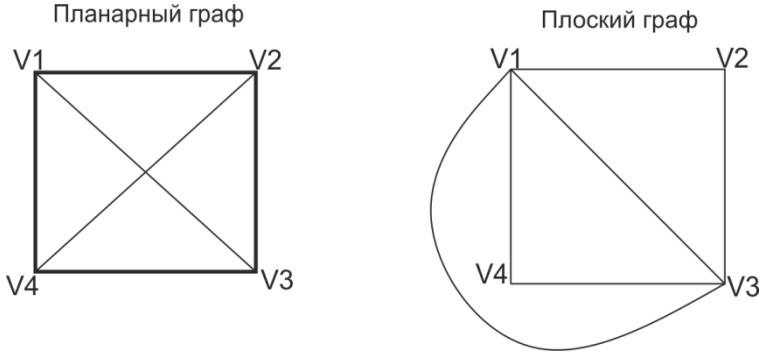


Существует Гамильтоновы путь и цикл, существует Эйлеров путь, но не существует Эйлеров цикл.

Планарные и плоские графы

Опр. Планарный граф - граф, для которого существует изоморфные ему (матрица смежности одна и та же), такой, что его можно нарисовать на плоскости без пересечения рёбер (кроме пересечения в вершинах).

Опр. Плоский граф - нарисованный без пересечений на плоскости планарный граф.



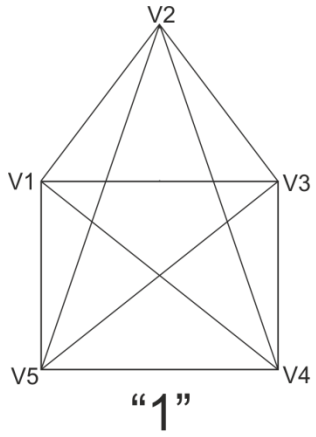
Опр. Грань планарного графа - часть плоскости, ограниченная рёбрами, и внутри нет рёбер и вершин.

$n - m + \gamma = 2$ - ***формула Эйлера для плоского графа***

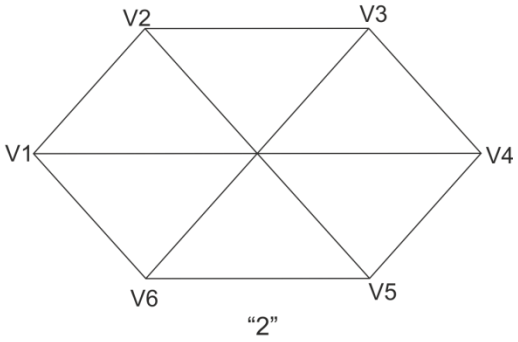
В примере: 4-6+4=2 ($\gamma = 4$), внешняя грань тоже считается.

Число циклов в базисе: $\gamma - 1$.

Теор. Граф является планарным тогда и только тогда, когда в нем не существует подграфов “1” и “2”.



“1”



“2”

Хроматическое число графа

Опр. Хроматическое число графа ($\gamma(a)$) - минимальное число цветов, в которые можно раскрасить вершины так, что смежные вершины были бы окрашены в разные цвета.

Алгоритм раскраски основывается на нахождении максимальных внутренне устойчивых подмножеств графа (Кофман). Алгоритм переборный –большая вычислительная сложность. Существует множество приближенных алгоритмов меньшей сложности, но цветов в раскраске может получиться больше, чем хроматическое.

Раскраска плоскостей плоского графа (раскраска карт).

Теор. Плоский граф можно раскрасить не более, чем четырьмя цветами.

Доказана теорема с использованием вычислительных средств (написана программа перебора всех вариантов).

Реберная раскраска: Ребра, инцидентные одной вершине раскрашиваются в разные цвета.

Задача сводится к раскраске вершин реберного графа, т.е. графа, вершинами которого являются ребра исходного. Причем вершины соединены ребром, если соответствующие вершинам ребра исходного графа были инцидентны одной вершине.