"Курсовая работа по дискретной математике"

Иван Кочкожаров, студент группы М8О-108Б-22

4 июня 2023 г.

1. Определить для орграфа, заданного матрицей смежности $A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

а) матрицу односторонней связности;

б) матрицу сильной связности;

в) компоненты сильной связности;

г) матрицу контуров.

Решение.

Изображение графа:

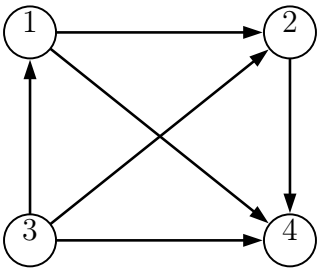


Рис. 1: Граф G

Матрица односторнней связности:

$$A = A(D) = \begin{array}{|c|c|c|c|c|} \hline & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 0 & 1 & 0 & 1 \\ \hline v_2 & 0 & 0 & 0 & 1 \\ \hline v_3 & 1 & 1 & 0 & 1 \\ \hline v_4 & 0 & 0 & 0 & 0 \\ \hline \end{array}.$$

$$A^2 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

$$A^3 = A \cdot A^2 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

$$T(D) = E \vee A \vee A^2 \vee A^3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \vee \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \vee \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \vee$$

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

Матрица двусторонней связности:

$$S(D) = T(D)\&[T(D)]^{\mathrm{T}} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \& \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$S(D) = E \Rightarrow$ в графе D нет контуров.

Компонентны сильной связности:

$S_2(D) = S(D) =$

	v_1	v_2	v_3	v_4
v_1	1	0	0	0
v_2	0	1	0	0
v_3	0	0	1	0
v_4	0	0	0	1

$D_1 = (V_1, X_1), V_1 = \{v_1\}$

$A(D_1) =$

	v_1
v_1	0

$D_1:$ (1)

$S_2(D) =$

	v_2	v_3	v_4
v_2	1	0	0
v_3	0	1	0
v_4	0	0	1

$D_2 = (V_2, X_2), V_2 = \{v_2\}$

$A(D_2) =$

	v_2
v_2	0

$D_2:$ (2)

$S_3(D) =$

	v_3	v_4
v_3	1	0
v_4	0	1

$D_3 = (V_3, X_3), V_3 = \{v_3\}$

$A(D_3) =$

	v_3
v_3	0

$D_3:$ (3)

$S_4(D) =$

	v_4
v_4	1

$D_4 = (V_4, X_4), V_4 = \{v_4\}$

$A(D_4) =$

	v_4
v_4	0

$D_4:$ (4)

Матрица контуров:

$$K(D) = A(D)\&S(D) = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

2. Используя алгоритм Терри, определить замкнутый маршрут, проходящий ровно по два раза (по одному в каждом направлении) через каждое ребро графа.

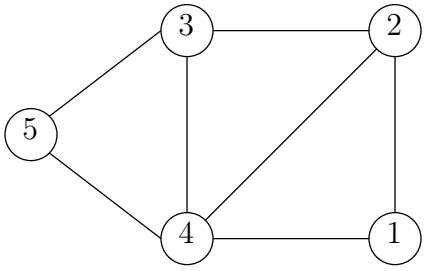


Рис. 2: Граф G

Решение.

Для решения этой задачи действуем в соответствии с алгоритмом Тэрри. Для реализации алгоритма помечаем первые заходящие в вершины ребра крестиками, которые наносим на ребрах ближе к той вершине в которую в первый раз заходим, а также указываем направления прохождения ребер и последовательность прохождения ребер. Алгоритм дает следующий возможный маршрут:

$$v_1v_2v_3v_5v_4v_3v_4v_2v_4v_1v_4v_5v_3v_2v_1$$

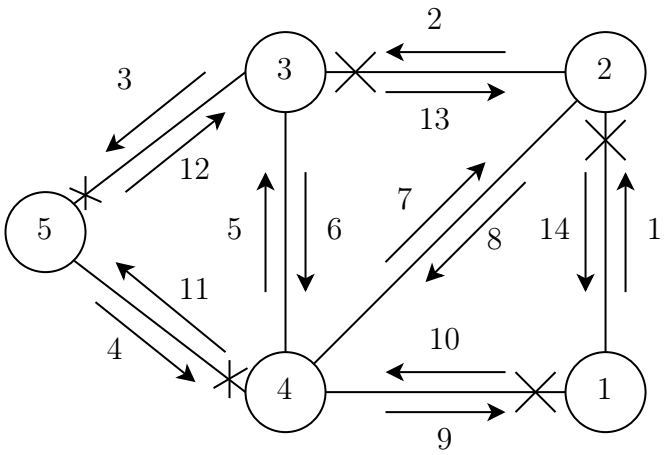


Рис. 3: Визуализация алгоритма Терри

3. Орграф $D = (V, X)$, где $V = \{v_1, \ldots, v_{10}\}$ задан матрицей смежности $A(D)$. Найти все минимальные пути v_1 в v_8.

$A = A(D) =$

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1	0	0	1	0	0	1	0	0
v_2	1	0	1	1	1	1	0	0
v_3	1	0	0	0	0	1	0	0
v_4	1	1	1	0	0	1	0	0
v_5	1	1	1	1	0	0	1	1
v_6	0	0	1	1	0	0	0	0
v_7	1	0	1	1	1	1	1	0
v_8	1	0	1	1	0	0	1	0

.

Решение.

Действуя согласно алгоритму фронта волны, последовательно определяем:

$FW_0(v_1) = \{v_1\}, FW_1(v_1) = D(v_1) = \{v_3, v_6\}$,
$FW_2(v_1) = D(FW_1(v_1)) \setminus (FW_0(v_1) \cup FW_1(v_1)) = D(\{v_3, v_6\}) \setminus \{v_1, v_3, v_6\} =$
$= \{v_1, v_3, v_4, v_6\} \setminus \{v_1, v_3, v_6\} = \{v_4\}$
$FW_3(v_1) = D(FW_2(v_1)) \setminus (FW_0(v_1) \cup FW_1(v_1) \cup FW_2(v_1)) = \{v_1, v_2, v_3, v_6\} \setminus$
$\setminus \{v_1, v_3, v_4, v_6\} = \{v_2\}$
$FW_4(v_1) = D(FW_3(v_1)) \setminus (FW_0(v_1) \cup FW_1(v_1) \cup FW_2(v_1) \cup FW_3(v_1)) =$
$= \{v_1, v_3, v_4, v_5, v_6\} \setminus \{v_1, v_2, v_3, v_4, v_6\} = \{v_5\}$
$FW_5(v_1) = D(FW_4(v_1)) \setminus (FW_0(v_1) \cup FW_1(v_1) \cup FW_2(v_1) \cup FW_3(v_1) \cup FW_4(v_1)) =$
$\{v_1, v_2, v_3, v_4, v_7, v_8\} \setminus \{v_1, v_2, v_3, v_4, v_5, v_6\} = \{v_7, v_8\}$

Таким образом, $v_8 \in FW_5(v_1)$, а следовательно, согласно алгоритму фронта волны существует минимальный путь в орграфе D из v_1 в v_8 длины 5. Найдём все эти пути.

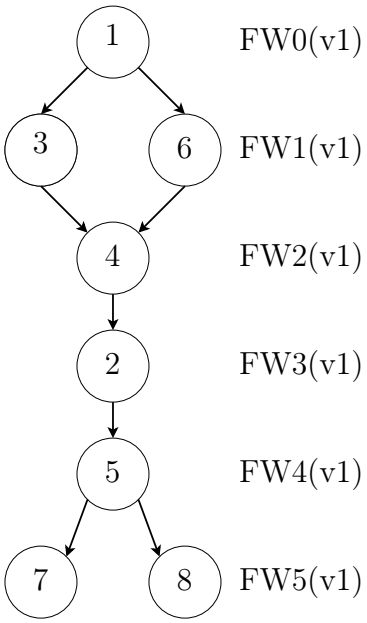


Рис. 4: Граф D'

На рисунке 4 изображен подграф D' орграфа D, на котором последовательно изображены множества $FW_k(v_1), k = 1, 2, 3, 4, 5$, а так же дуги вида (v, v'), где для некоторого $k \in \{0, 1, 2, 3, 4\}, v \in FW_k(v_1), v' \in FW_{k+1}(v_1)$, т.е. исходящие из вершин некоторого k-го фронта волны и заходящие в вершины следующего $(k + 1)$-го фронта волны.

Используя изображение D' нетрудно выделить все минимальные пути из v_1 в v_8 в орграфе D. При этом, следуя алгоритму фронта волны, находим эти минимальные пути, используя орграф D' но двигаясь в D' в обратной последовательности (т.е. не из $v1$ в v_8 а наоборот, из v_8 в v_1). Используя рисунок 4, получаем, что в любом минимальном пути из v_1 в v_8 соблюдается следующая последовательность вершин. Вершиной, предшествующей вершине v_8 может быть v_5. Вершиной, предшествующей вершине 5 может быть v_2. Вершиной, предшествующей вершине v_2 – вершина v_4. Вершиной, предшествующей вершине v_4 – любая из вершин v_3, v_6. Вершиной, предшествующей вершинам v_3 и v_6 может быть только v_1. Этими условиями однозначно определяется множество минимальных путей из v_1 в v_8 которое компактно изображено на рисунке 5. На этом рисунке изображены все вершины, входящие в минимальные пути v_1 в v_8 Для каждой из промежуточных вершин v показано множество вершин, которые могут ей предшествовать, а также соответствующие дуги (исходящие из вершин, предшествующих v и заходящие в v). Из рисунка 5 видно, что всего существует два минимальных пути из v_1 в v_8: $v_1v_3v_4v_2v_5v_8$, $v_1v_6v_4v_2v_5v_8$.

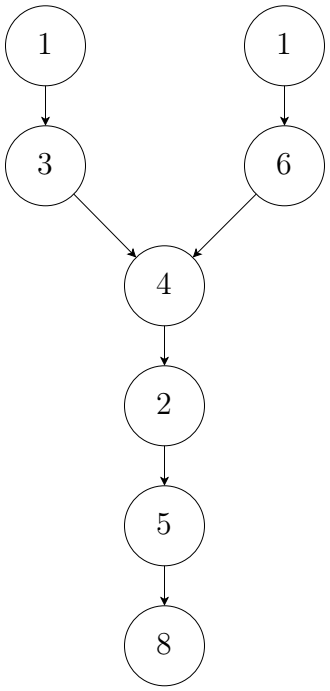


Рис. 5: Граф минимальных путей

4. Нагруженный орграф D задан матрицей длин дуг $C(D)$. Найти минимальные пути из v_1 во все достижимые вершины.

$$C(D) = \begin{pmatrix} \infty & 4 & \infty & \infty & 5 & \infty & \infty & \infty \\ 5 & \infty & 7 & 10 & 2 & \infty & \infty & \infty \\ \infty & \infty & \infty & 2 & \infty & 2 & \infty & \infty \\ 6 & \infty & \infty & \infty & \infty & \infty & 3 & 5 \\ 3 & 2 & \infty & \infty & \infty & 3 & 11 & \infty \\ 4 & \infty & 2 & \infty & \infty & \infty & 7 & \infty \\ 8 & \infty & \infty & 3 & \infty & \infty & \infty & 3 \\ \infty & \infty & \infty & \infty & 17 & \infty & \infty & \infty \end{pmatrix}$$

Решение.

Воспользуемся алгоритмом Форда. Сначала определим таблицу величин $\lambda_i^{(i)}, i = 1, 2, \ldots, n-1$, где $n = 8$

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8		$\lambda^{(0)}$	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$	$\lambda^{(4)}$	$\lambda^{(5)}$	$\lambda^{(6)}$	$\lambda^{(7)}$
v_1	∞	4	∞	∞	5	∞	∞	∞		**0**	0	0	0	0	0	0	0
v_2	5	∞	7	10	2	∞	∞	∞		∞	4	4	4	4	4	4	4
v_3	∞	∞	∞	2	∞	2	∞	∞		∞	∞	11	**10**	10	10	10	10
v_4	6	∞	∞	∞	∞	∞	3	5		∞	∞	14	13	**12**	12	12	12
v_5	3	2	∞	∞	∞	3	11	∞		∞	**5**	5	5	5	5	5	5
v_6	4	∞	2	∞	∞	∞	7	∞		∞	∞	**8**	8	8	8	8	8
v_7	8	∞	∞	3	∞	∞	∞	3		∞	∞	16	15	15	15	15	15
v_8	∞	∞	∞	∞	17	∞	∞	∞		∞	∞	∞	19	18	**17**	17	17

$C(D)$ Таблица величин

Обозначим $\lambda^{(k)} = (\lambda_1^{(k)}, \ldots, \lambda_8^{(k)})^{\mathrm{T}}$, где $k = 0, 1, \ldots, 7$. Это столбцы в таблице величин. Первая строка по таблицы величин состоит из нулевых элементов ($\lambda_1^{(k)} = 0, k = 0, 1, \ldots, 7$), а первый столбец заполняем следующим образом: $\lambda_i^{(0)} = \infty, i = 2, \ldots, 8$. Далее, используя формулу $\lambda_j^{(k+1)} = \min\limits_{1 \le i \le 8} \{\lambda_i^{(k)} + c_{ij}\}$ последовательно определяем элементы столбца $\lambda^{(1)}$, используя элементы столбца $\lambda^{(0)}$ (а так же элементы матрицы $C(D)$), затем находим элементы столбца $\lambda^{(2)}$, используя элементы столбца $\lambda^{(1)}$ и т.д.

Длина минимального пути из v_1 в v_8 равна 17. Вершине v_8 предшествует v_4, потому что $\lambda_8^{(5)} = 17 = \lambda_4^{(4)} + c_{48} = 12 + 5$. Вершине v_4 предшествует v_3 и т.д. В итоге получаем минимальный путь: $v_1v_5v_6v_3v_4v_8$ (в таблице выделен жирным шрифтом). Соответственно, $v_1v_5v_6v_3v_4$, $v_1v_5v_6v_3$, $v_1v_5v_6$, v_1v_5 - минимальные пути из v_1 в соответствующие вершины. Минимальный путь из v_1 в v_7 находится аналогично. Получаем такой минимальный путь: $v_1v_5v_6v_7$. Минимальный путь из v_1 в v_2, очевидно, v_1v_2.

5. Найти остовное дерево графа G с минимальной суммой длин входящих в него ребер.

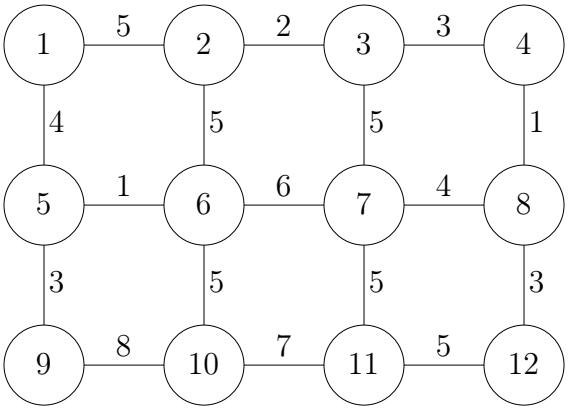


Рис. 6: Граф G

Решение. Согласно алгоритму Краскала выбираем ребро $\{v_5, v_6\}$ минимальной длины 1. Выделяем его жирной линией (см. рис. 7). Далее выбираем ребро минимальной длины, соединяющее либо v_5 либо v_6 с какой-нибудь новой (т.е. отличной от v_1, v_5) вершиной графа G (т.е. выбираем среди ребер $\{v_5, v_1\}$,$\{v_5, v_9\}$,$\{v_6, v_2\}$, $\{v_6, v_7\}$,$\{v_6, v_{10}\}$). Минимальную длину имеет ребро $\{v_5, v_9\}$. Выделяем его жирной линией (см. рис. 7). Далее выбираем ребро минимальной длины, соединяющее либо v_5, либо v_6, либо v_9 с какой-нибудь новой вершиной графа (выбираем между $\{v_9, v_{10}\}$,$\{v_6, v_{10}\}$,$\{v_6, v_7\}$,$\{v_6, v_2\}$,$\{v_5, v_1\}$). Минимальную длину имеет ребро $\{v_5, v_1\}$ Выделяем его жирной линией (см. рис. 7). Следующим ребром минимальной длины (если таких несколько, можно выбрать любое) среди всех возможных является $\{v_1, v_2\}$, затем $\{v_2, v_3\}$, далее $\{v_3, v_4\}$, далее $\{v_4, v_8\}$, далее $\{v_8, v_{12}\}$, далее $\{v_7, v_8\}$, далее $\{v_7, v_{11}\}$ и, наконец, $\{v_6, v_{10}\}$. Выделено $11 = 12 - 1 = n(G) - 1$ ребер, алгоритм окончен, выделяем минимальное остовное дерево графа (см. на рис. 7 подграф графа G, ребра которого выделены жирными линиями).

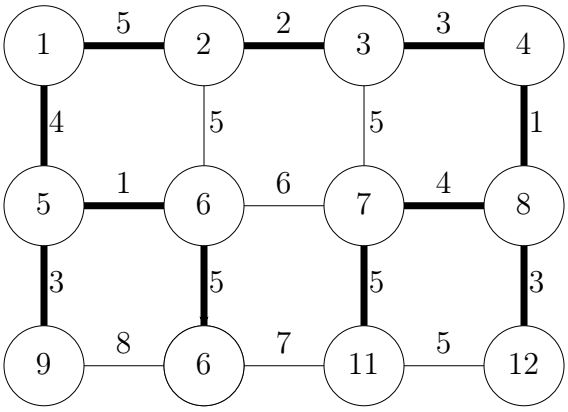


Рис. 7: Граф G с выделенным подграфом - минимальным остовным деревом

6. Пусть каждому ребру неориентированного графа изображенного на рис. 8 соответствует некоторый элемент электрической цепи. Составить линейно незави-

симые системы уравнений Кирхгофа для токов и напряжений. Пусть первому и пятому ребру соответствуют источники тока с ЭДС E_1 и E_2 (полярность выбирается произвольно), а остальные элементы являются сопротивлениями. Используя закон Ома, и, предполагая внутренние сопротивления источников тока равными нулю, получить общую систему уравнений для токов.

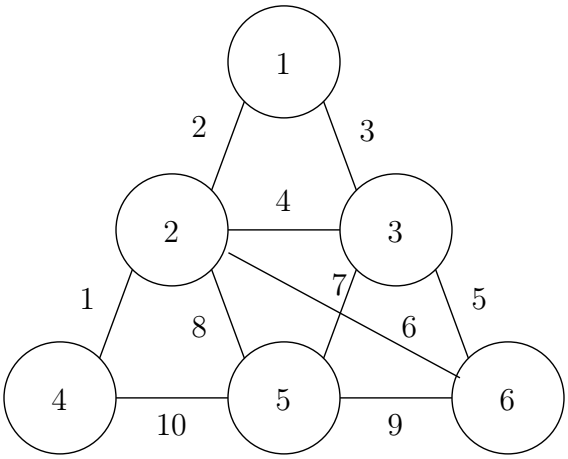


Рис. 8: Граф G

Решение. Выделим произвольным образом остовное дерево графа. Для графа изображенного на рис. 8, одним из возможных остовных деревьев является дерево, изображенное на рис. 9 (пунктирными линиями изображены удаленные из ребра).

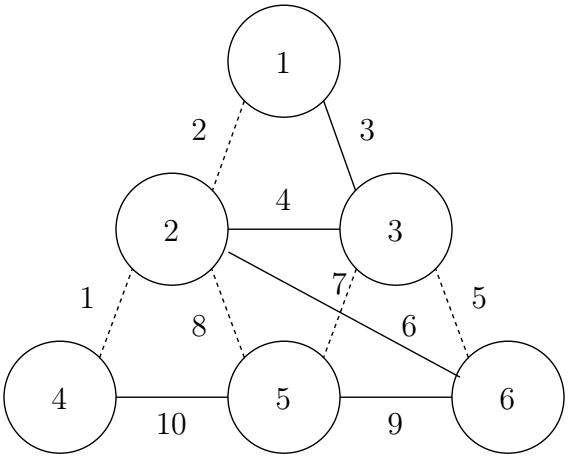


Рис. 9: Остовное дерево графа G

Добавляя любое из ребер, не вошедших в остовное дерево графа G (изображенных на рис. 9 пунктирными линиями), мы получим граф с некоторым простым циклом. Всего в остовное дерево не вошли $\nu(G) = m(G) - n(G) + 1$ ребер (для графа, изображенного на рис. 8, $\nu(G) = 10 - 6 + 1 = 5$), а поэтому можем получить таким образом $\nu(G) = 5$ простых циклов. Эти циклы различны в том смысле что каждый из них проходит через ребро (то самое, которое мы добавляли для выделения

данного цикла), через которое не проходит ни один другой цикл. Они образуют *цикловой базис графа* G.

Для графа, изображенного на рис. 8 в цикловой базис войдут циклы:
$\mu_1 = \mu_1(x_1) = x_1x_6x_9x_{10}, \mu_2 = \mu_2(x_2) = x_2x_3x_4, \mu_3 = \mu_3(x_5) = x_5x_6x_4$
$\mu_4 = \mu_4(x_7) = x_7x_9x_6x_4, \mu_5 = \mu_5(x_8) = x_8x_6x_9$

Введем произвольную ориентацию на ребрах графа G. В результате каждое ребро x_j превратится в дугу $\tilde{x_j}$ и, соответственно, множество ребер X в множество дуг $\widetilde{X}$, а сам граф $G = (V, X)$ в орграф $D = (V, \widetilde{X})$. Для графа G, изображенного на рис. 8, в результате введения ориентации на его ребрах получаем, например, орграф $D = (V, \widetilde{X})$, изображенный на рис. 10.

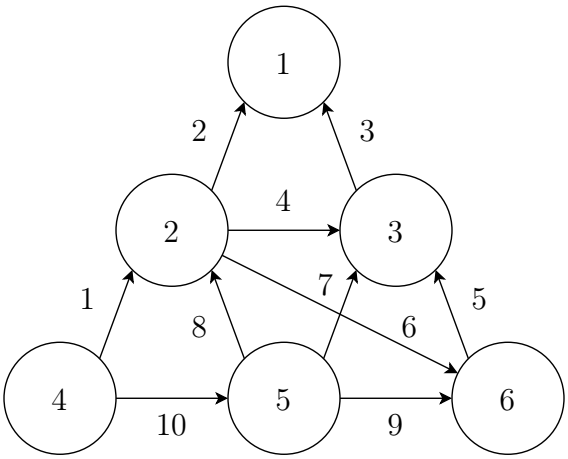


Рис. 10: Орграф D

Для графа G с выделенным ранее цикловым базисом $\mu1, \mu2, \mu3, \mu4, \mu5$ и выбранной ориентацией ребер, соответствующей орграфу D, изображенному на рис. 10, цикломатическая матрица имеет вид:

$C(G) =$

	*	*			*		*	*		
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
μ_1	1	0	0	0	0	1	0	0	-1	-1
μ_2	0	1	-1	-1	0	0	0	0	0	0
μ_3	0	0	0	1	-1	-1	0	0	0	0
μ_4	0	0	0	1	0	-1	-1	0	1	0
μ_5	0	0	0	0	0	1	0	1	-1	0

При построении циклового базиса графа G мы поочередно добавляли к остовному дереву графа G ребра x_1, x_2, x_5, x_7, x_8. Выделяем соответствующие этим ребрам столбцы в матрице $C(G)$. Из выделенных столбцов составим матрицу и найдем ее определитель.

$$\begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix} \neq 0$$

Ранг матрицы $C(G)$ равен числу строк, т.е. $\nu(G)$.

Пусть теперь граф G соответсвует электрической цепи, где первому и пятому ребру соответствуют E_1 и E_2 (полярность выбирается произвольно), а остальные элементы являются сопротивлениями.

Выберем произвольным образом направления токов в элементах цепи (условные направления; после решения соответствующей системы уравнений знаки при величинах токов покажут истинные направления токов). Пусть эти направления соответствуют выбранной ранее ориентации ребер графа G. Ввпишем систему уравнений Кирхгофа для напряжений:
$\mu_1 : -u_1 + u_6 - u_9 - u_{10} = 0$
$\mu_2 : u_2 - u_3 - u_4 = 0$
$\mu_3 : u_4 - u_5 - u_6 = 0$
$\mu_4 : u_4 - u_6 - u_7 + u_9 = 0$
$\mu_5 : u_6 + u_8 - u_9 = 0$
или, с учетом закона Ома, а также того, что $u_1 = E_1, u_5 = E_2$, а номера сопротивлений соответсвуют номерам ребер, имеем:

$$\begin{cases} -E_1 + i_6 r_6 - i_9 r_9 - i_{10} r_{10} = 0 \\ i_2 r_2 - i_3 r_3 - i_4 r_4 = 0 \\ i_4 r_4 - E_2 - i_6 r_6 = 0 \\ i_4 r_4 - i_6 r_6 - i_7 r_7 + i_9 r_9 = 0 \\ i_6 r_6 + i - 8 r_8 - i_9 r_9 = 0 \end{cases}$$

Система уравнений Кирхгофа для токов имеет вид, где

$C(G) =$

	$\widetilde{x}_1$	$\widetilde{x}_2$	$\widetilde{x}_3$	$\widetilde{x}_4$	$\widetilde{x}_5$	$\widetilde{x}_6$	$\widetilde{x}_7$	$\widetilde{x}_8$	$\widetilde{x}_9$	$\widetilde{x}_{10}$
v_1	0	1	1	0	0	0	0	0	0	0
v_2	1	-1	0	-1	0	-1	0	1	0	0
v_3	0	0	-1	1	1	0	1	0	0	0
v_4	-1	0	0	0	0	0	0	0	0	-1
v_5	0	0	0	0	0	0	-1	-1	-1	1
v_6	0	0	0	0	-1	1	0	0	1	0

При этом для достижения линейной независимости системы уравнений Кирхгофа для токов необходимо исключить из системы любое уравнение, например, второе. В результате система линейно независимых уравнений Кирхгофа для токов имеет вид:

$$\begin{cases} i_2 + i_3 = 0 \\ -i_3 + i_4 + i_5 + i_7 = 0 \\ -i_1 - i_{10} = 0 \\ -i_7 - i_8 - i_9 + i_{10} = 0 \\ -i_5 + i_6 + i_9 = 0 \end{cases}$$

Таким образом, общей системой уравнений для токов является объединение систем. Заметим, что полученная объединенная система уравнений состоит из девяти уравнений относительно девяти неизвестных $i_1, i_2, \ldots, i_{10}$: после нахождения которых нетрудно определить $u_1, u_2, \ldots, u_{10}$.

7. Построить максимальный поток по транспортной сети.

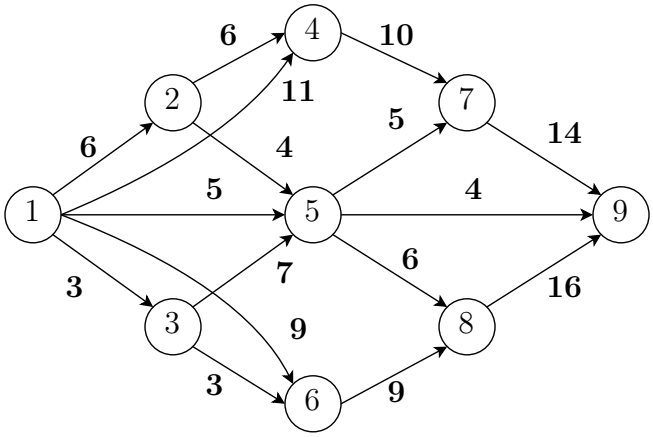


Рис. 11: Граф транспортной сети D

Решение. Начинаем с нулевого потока ϕ_0. Каждой новой цепи из v_1 в $v_n = v_5$ будем ставить в соответствие ее очередной номер, т.е. будем обозначать эти цепи через η_1, η_2 и т.д. Соответственно, после нахождения цепи η_1 поток ϕ_0 изменится на поток ϕ_1. После нахождения цепи η_2 поток ϕ_1 изменится на ϕ_2 и т.д. Числа, на которые увеличиваем потоки по дугам из η_i обозначаем через α_i. Насыщенные дуги при изображении транспортной сети D с очередным потоком ϕ_i помечаем символом $\times$. На рис. 12 приведены изображения орграфа D с потоком ϕ_0, а так же вспомогательного орграфа D', который на этом этапе совпадает с D.

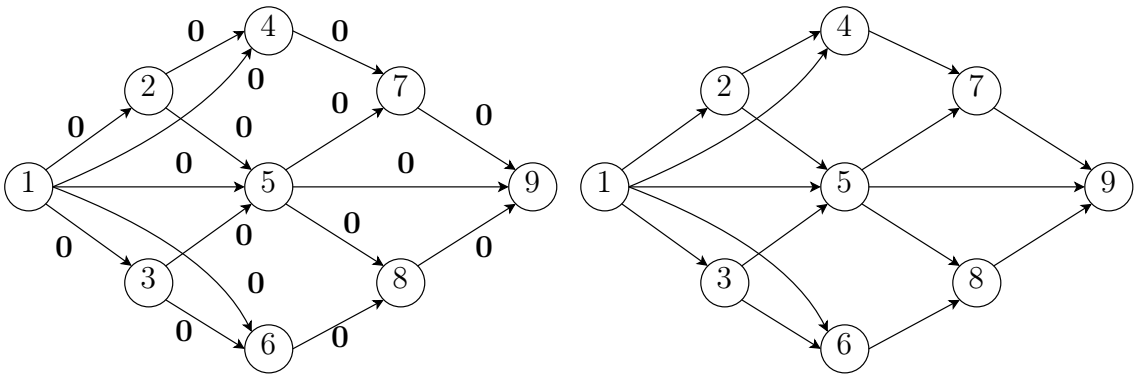


Рис. 12

Выделяем в D' простую цепь $\eta_1 = v_1v_2v_4v_7v_9$ из v_1 в v_9. Увеличиваем поток $\phi(x)$ по каждой дуге x из η_1 на одинаковую величину $\alpha_1 = 6$ до насыщения дуг (v_1v_2) и (v_2v_4). На рис. 13 приведены изображения орграфа D с потоком ϕ_1, а также соответсвующего этому потоку вспомогательного орграфа D'.

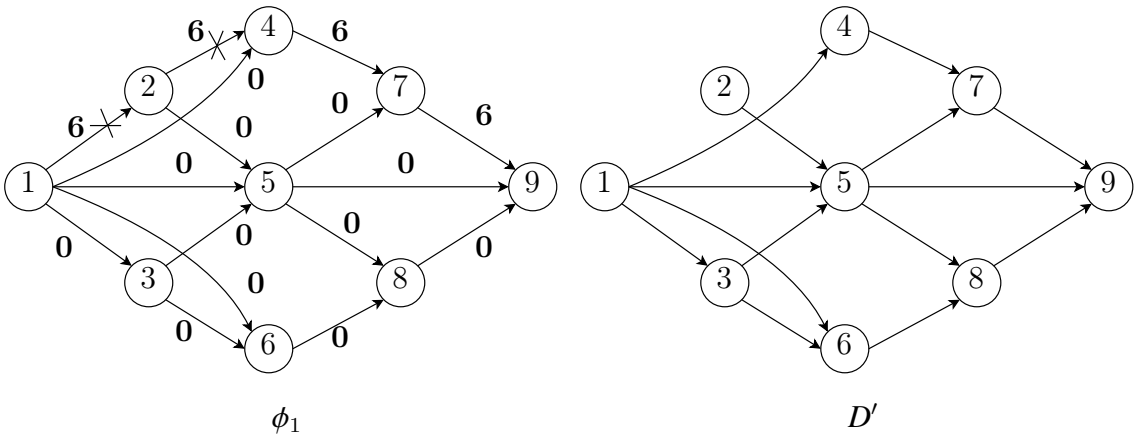


Рис. 13

Продолжаем выделять произвольные простые цепи и максимально увеличивать поток на них, а затем удалять насыщенные дуги, пока сток перестанет быть доступным из истока.

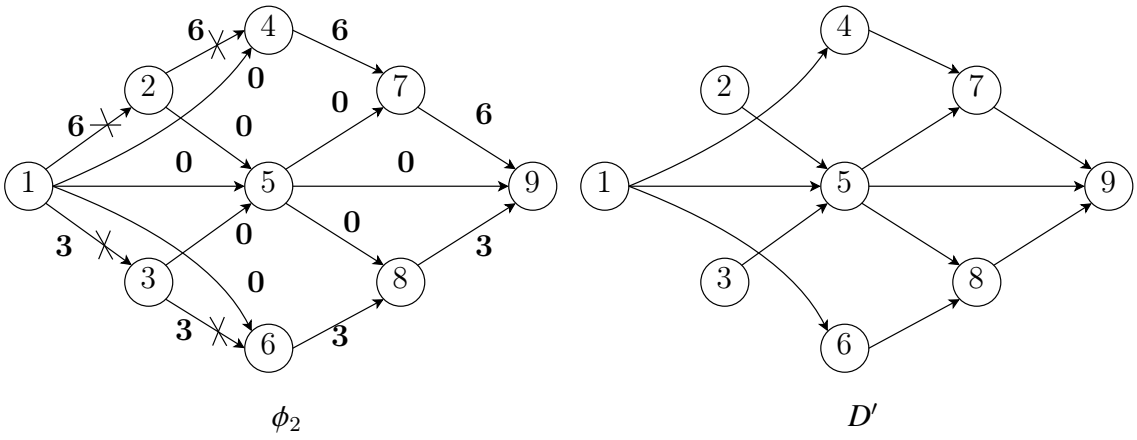


Рис. 14

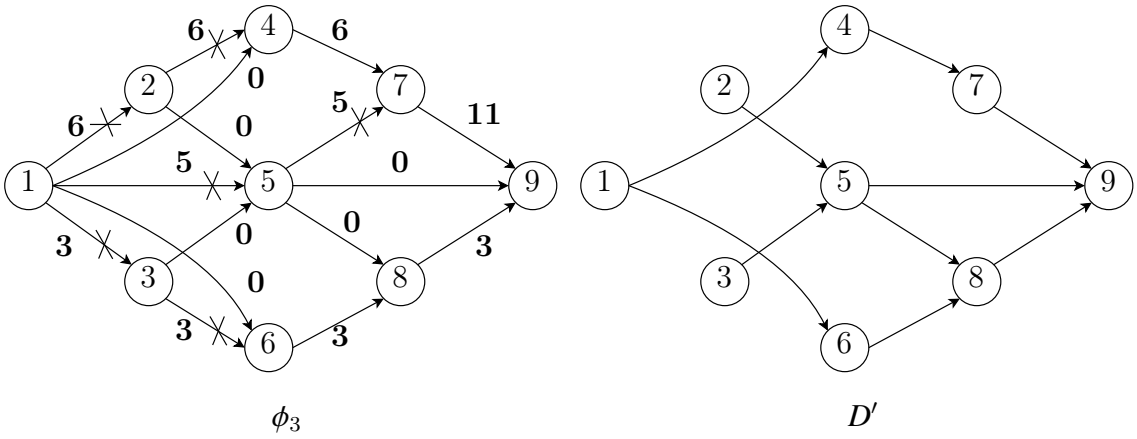


Рис. 15

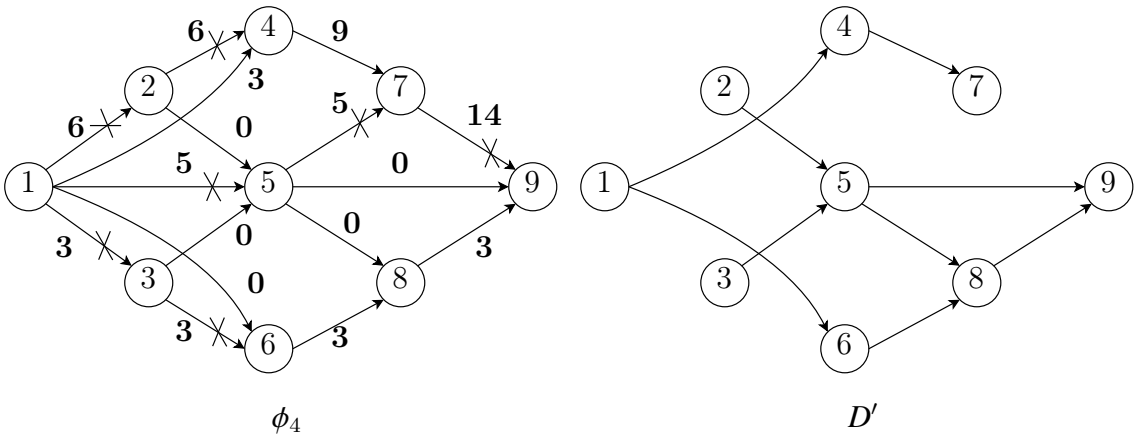


Рис. 16

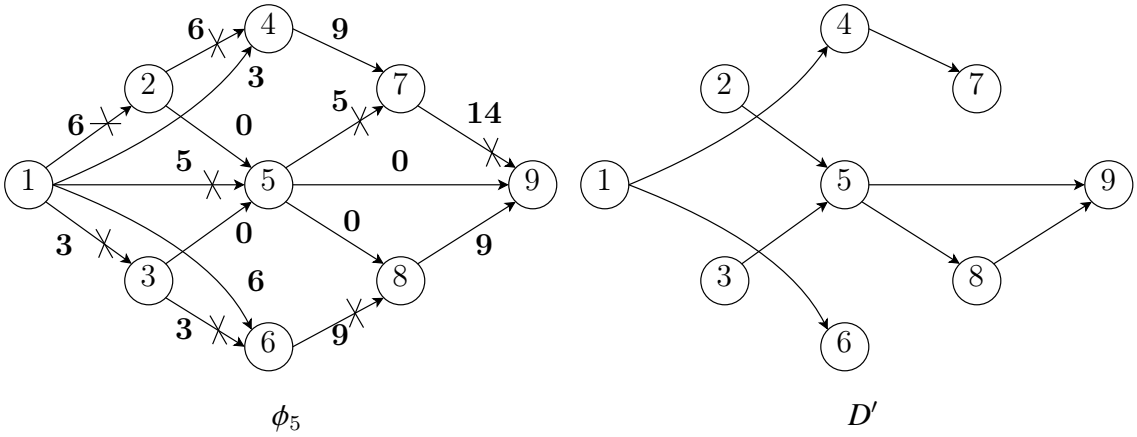


Рис. 17: Орграф приращений $I(D, \phi)$ и модифицированный орграф

Мы видим, что для орграфа D', соотвествующего потоку ϕ_2 не существует пути из источника в сток, а следовательно, ϕ_3 – полный поток. Для увеличения полного потока до максимального нужно построить орграф приращений.

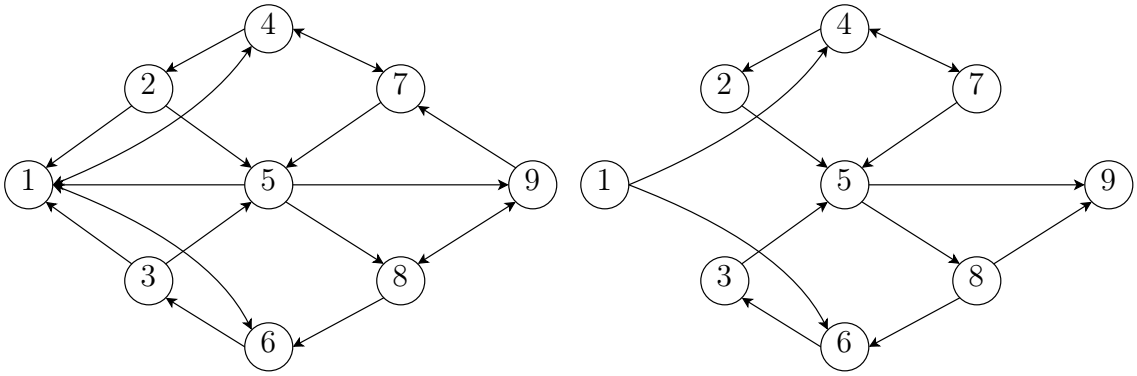


Рис. 18

Для построения максимального потока воспользуемся алгоритмом Форда-Фалкерсона. Начинаем с ранее построенного потока ϕ_5. Выделяем в $I(D, \phi)$ простую цепь $\eta_6 =$

$v_1v_6v_3v_5v_9$. Увеличиваем потоки по дугам из η_4 на одинаковую величину, равную 3, до насыщения (v_1, v_6), при этом поток по дугам $(v_3, v_5), (v_5, v_9)$ не превышает пропускной способности, а по дуге (v_3, v_6) уменьшается на 3. В результате поток ϕ_5 меняется на поток ϕ_6. Далее строим модифицированный орграф приращений.

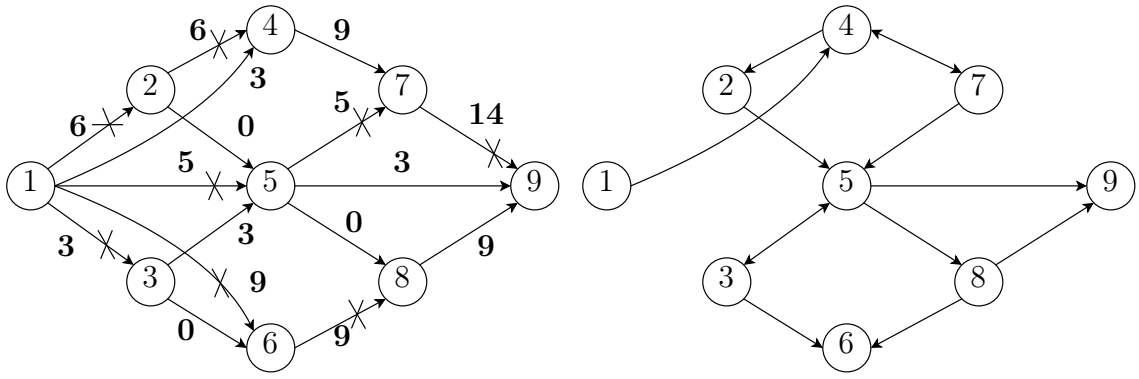


Рис. 19

Выполняем те же самые действия для цепи $v_1v_4v_2v_5v_8v_9$ (увеличиваем поток по ней на 4, получаем ϕ_7)

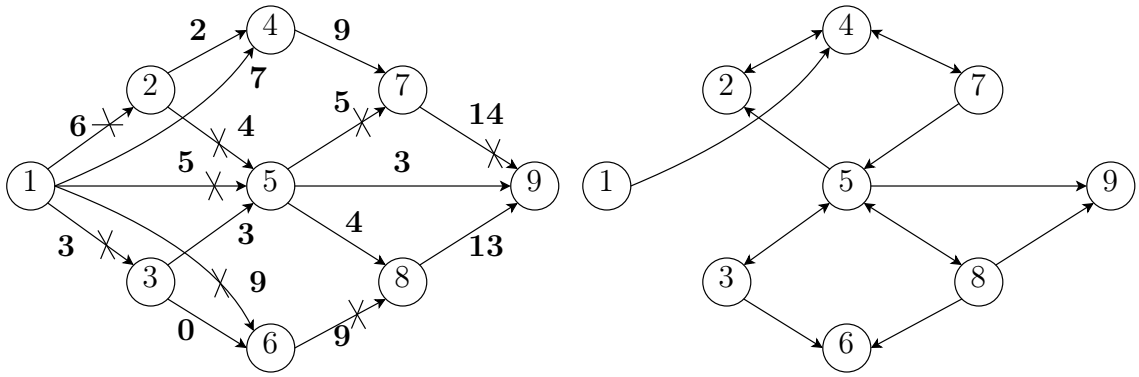


Рис. 20

Выполняем те же самые действия для цепи $v_1v_4v_7v_5v_9$ (увеличиваем поток на 1 по ней на 1, получаем ϕ_8)

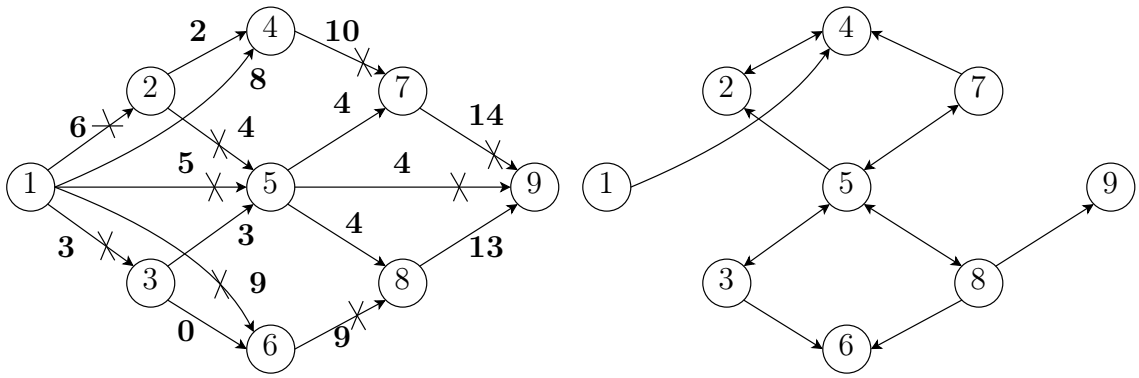


Рис. 21

Поскольку в $I(D, \phi_8)$ вешина v_9 не достижима из v_1, то согласно алгоритму - ϕ_8 - искомый максимальный поток, при этом $\overline{\phi}_8 = 31$.

8. Построение функции Гранди графа. Изучить возможность построения функции Гранди для графа, содержащего контуры

1 Алгоритм

1.1 Определение функции Гранди

Рассмотрим орграфа $D = (V, X)$. Функция $g(v)$, ставящая в соответствие каждой вершине $v \in V$ целое число $g(v) \geq 0$, называется *функцией Гранди* для орграфа D, если в каждой вершине $v \in V$ число $g(v)$ является миинмальным из всех целых неотрицательных чисел, не принадлежаших множеству $\{g(w)|w \in D(v)\}$, и $g(v) = 0$ при $D(v) = \varnothing$. Если для оргафа D существует функция Гранди, то говорят, что орграф *допускает* (в противном случае - *не допускает*) функцию Гранди. Не всякий орграф D допускает функцию Гранди, а если допускает, то она не обязательно единственная.

1.2 Описание алгоритма

Алгоритм находит все возможные функции Гранди для орграфа $D = (V, X)$, или определяет, что она для него недопустима. Граф вводится в виде матрицы смежности $n \times n$.

1. Разбить граф на уровни $V_0, V_1 \dots V_n$, используя алгоритм разбиения орграфа на уровни. Если разбиение удалось, перейти к пункту 2. Иначе, перейти к пункту 3 (т.к. заданный орграф имеет контуры).

2. Определяем функцию Гранди для первых двух уровней: $\forall v \in V_0, g(v) = 0; \forall v \in V_1, g(v) = 1$. Далее находим значения функции на каждом уровне $V_i, i \geq 2$, используя значения функции на предыдущих уровнях. Функция Гранди будет однозначно определена для этого графа.

3. Находим ядра графа N ($\forall v \in N, N \cap D(v) = \varnothing; \forall v \in V \setminus N, n \cap D(v) \neq \varnothing$). Для этого можно воспользоваться методом Магу. Значения функции Гранди в вершинах, входящих в ядро равны 0. Рекурсивно определяем значения функции Гранди в остальных вершинах. Повторив эти действия для всех ядер, получаем все возможные функции Гранди, допустимые для этого графа. Если ядер нет — функция Гранди недопустима.

1.3 Обоснование алгоритма

1. Если у орграфа $D = (V, X)$ нет контуров, то разбиение на уровни $V_0, V_1 \dots V_r$:

$$V_0 = \{v \in v || D(v) = \varnothing\};$$
$$V_1 = \{v \in V \setminus V_0 | D(v) \subseteq V_0\};$$
$$V_2 = \{v \in V \setminus (V_0 \cup V_1) | D(v) \subseteq V_0 \cup V_1\};$$
$$\dots$$
$$V_r = \{v \in V \setminus \bigcup_{k=0}^{r-1} V_k | D(v) \subseteq \bigcup_{k=0}^{r-1} V_k\},$$
$$r = \min\{n \in \mathbb{N} | V \setminus \bigcup_{k=0}^{r} = \varnothing\}$$

являются непустыми множествами, образующими разбиение множества вершин V (теорема 4.8 и утверждение 4.53 из "Курса дискретной математики").

Для нахождения этих уровней можно воспользоваться специальным алгоритмом, который так же может показать, что орграф имеет контуры, и разбиение невозможно. Для начала надо выписать матрицу смежности графа $A(D)$. Под матрицей образуем строку Λ_0, в i-м месте которой укажем число единиц в i-ой строке матрицы $A(D)$. Уровень образуют вершины, которым в строке Λ_0 соответсвует число 0. Если $V = V_0$, то V_0 - единственный уровень орграфа. Иначе, под Λ_0 образуем Λ_i, ставя под нулям из Λ_{i-1} символы $\times$ (и под символами $\times$ тоже ставим $\times$) и при подсчете единиц, не учитываем те, которые находятся над $\times$, до тех пор, пока вся Λ_i не будет состоять из 0 и $\times$. Тогда Λ_i будет соответсвовать V_i состоящий из вершин, которым в Λ_i соответсвует число 0. Если в течение алгоритма появится строка без 0, то это говорит о том, что орграф содержит контуры (алгоритм 4.9 и утверждение 4.54 из "Курса дискретной математики").

2. Напомним, что для орграфа $D(V, X)$ функция Гранди определяется рекурсивно: $g(v) = \min(\{n \in \mathbb{Z} | n \geq 0\} \setminus \{g(w) | w \in D(v)\})$ и $g(v) = 0, D(v) = \varnothing$.

Рассмотрим два случая. В первом случае орграф, для которого необходимо определить функцию Гранди, не имеет контуры, а во втором он с контурами.

Докажем следующее: *Если орграф допускает функцию Гранди, то найдется вершина $v \in V$ такая, что $g(v) = 0$.*

Пусть в орграфе D $\forall v \in V, g(v) > 0$. Рассмотрим произвольную вершину $w \in V$. Тогда, с одной стороны, в силу нашего утверждения имеем $g(w) > 0$, а с другой стороны, используя это же утверждение получаем, что либо $D(w) = \varnothing$, либо $\forall v \in D(w), g(v) > 0$, а следовательно, по определению функции Гранди, $g(w) = 0$, т.е. противоречие.

3. В первом случае орграф *допускает и притом единственную функцию Гранди.* Множество вершин разбито на уровни $V_0, \dots, V_r$. По определению функции Гранди, если она допустима для D, то $\forall v \in V_0, g(v) = 0; \forall v \in V_1, g(v) = 1$. Заметим, что значения функции Гранди на каждом уровне V_i, где $i \geq 2$, однозначно находятся по ее значениям на предыдущих уровнях $V_0, \dots, V_{i-1}$ (поскольку $\forall v \in V_i, D(v) \subseteq$

$\bigcup_{k=0}^{i-1} V_k$), а следовательно, исходя из определенных значений функции для вершин нулевого и первого уровней, ее можно однозначно определить на всех последующих уровнях.

4. Для рассмотрения второго случая докажем следующую теорему: *Если орграф* $D = (V, X)$ *допускает функцию Гранди* $g(v)$, *то множество вершин* $N = \{v \in V | g(v) = 0\}$ *является ядром этого орграфа.*

Покажем, что множество N удовлетворяет условиям (см. определение ядра):

1. $\forall v \in N, N \cap D(v) = \varnothing$;
2. $\forall v \in V \setminus N, N \cap D(v) \neq \varnothing$.

Докажем сначала, что выполняется первое условие. Пусть $v \in N$. Тогда $g(v) = 0$. Если предположить, что $N \cap D(v) \neq \varnothing$, то существует вершина $w \in N \cap D(v)$. Но тогда из $w \in N$ имеем $g(w) = 0$, а из $w \in D(w)$ получаем, что не может выполнятся равенство $g(v) = 0$. Данное противречие подтверждает справедливость первого условия.

Докажем теперь, что выполняется второе условие. Пусть $v \in V \setminus N$. Тогда $g(v) \neq 0$. Если предположить, что $N \cap D(v) = \varnothing$, то по определению функции Гранди должно выполнятся равенство $g(v) = 0$, т.е. пришли к противоречию, а значит $N \cap D(v) \neq \varnothing$.

Из этой теоремы следует то, что количество ядер графа равно количеству различных функций Гранди, которые он допускает. Поэтому, если найти ядра N_i графа с помощью метода Магу, можно рекурсивно определить i функций Гранди для остальных, учитывая, что она равна 0 в вершинах принадлежащих N_i. А если ядер нет, то нарушается ранее доказанное утверждение сушествования функции Гранди.

5. На ЭВМ проще всего реализовать метод Магу следующим образом. КНФ, получаемые из матрицы $A(D)$ смежности по специальным формулам.

$$D = (V, X), X \neq \varnothing, U \subseteq V$$

$$\varepsilon = \begin{cases} 1, \text{ если } v_i \in U \\ 0, \text{ если } v_i \in U \end{cases}$$

$$\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} (\bar{a}_{ij} \vee \bar{\varepsilon}_i \vee \bar{\varepsilon}_j) = 1 \Rightarrow \text{ U — внтуренне устойчивое множество}$$

$$\bigwedge_{i=1}^{n} (\varepsilon_i \vee \bigvee_{a_{ij}=1} \varepsilon_j) = 1 \Rightarrow \text{ U — внешне устойчивое множество}$$

$$\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} (\bar{a}_{ij} \vee \bar{\varepsilon}_i \vee \bar{\varepsilon}_j) \wedge \bigwedge_{i=1}^{n} (\varepsilon_i \vee \bigvee_{a_{ij}=1} \varepsilon_j) = 1 \Rightarrow \text{ U — ядро}$$

Эту КНФ нужно привести к СДНФ с помощью построения таблиц истинности. Получаем СДНФ, которая будет соответсвовать всем ядрам графа.

2 Программа алгоритма

Для работы скрипта необходимы библиотеки NetworkX и MatPlotLib. На вход требуется корректная матрица смежности графа. Если функция допустима, то в отдельном окне откроется граф с подписанными значениями функции Гранди в вершинах. При закрытии окна откроется следующая возможная функция (если такая существует). Иначе, скрипт сообщит о том, что функция недопустима.

```
#!/usr/bin/env python3
import matplotlib.pyplot as plt
import networkx as nx
from itertools import product

class InvalidMatrixException(Exception):
    pass

def input_digraph() -> list[list[int]]:
    first_line = list(map(int, input().split()))
    set_line = set(first_line)

    if set_line != set([1]) and set_line != set([0])\
                            and set_line != set([0,1]):
        raise InvalidMatrixException("Adjacency matrix "
                                     "can only contain 0 or 1")
    n = len(first_line)

    m = [first_line]

    for _ in range(n-1):
        line =list(map(int, input().split()))
        set_line = set(line)
        if set_line != set([1]) and set_line != set([0]) \
                                and set_line != set([0,1]):
            raise InvalidMatrixException("Adjacency matrix"
                                         "can only contain 0 or 1")
        if len(line) != n:
            raise InvalidMatrixException("Invalid matrix line size")
        m.append(line)
    return m

def digraph_levels(m: list[list[int]]):
    levels=[]
    n = len(m[0])
    lmbd = [sum(i) for i in m]
    levels.append([i for i in range(n) if lmbd[i]==0])
    if lmbd == [0]*n: return [lmbd]
    while (set(lmbd) != set([0]) ):
        new_lmbd = [0]*n
```

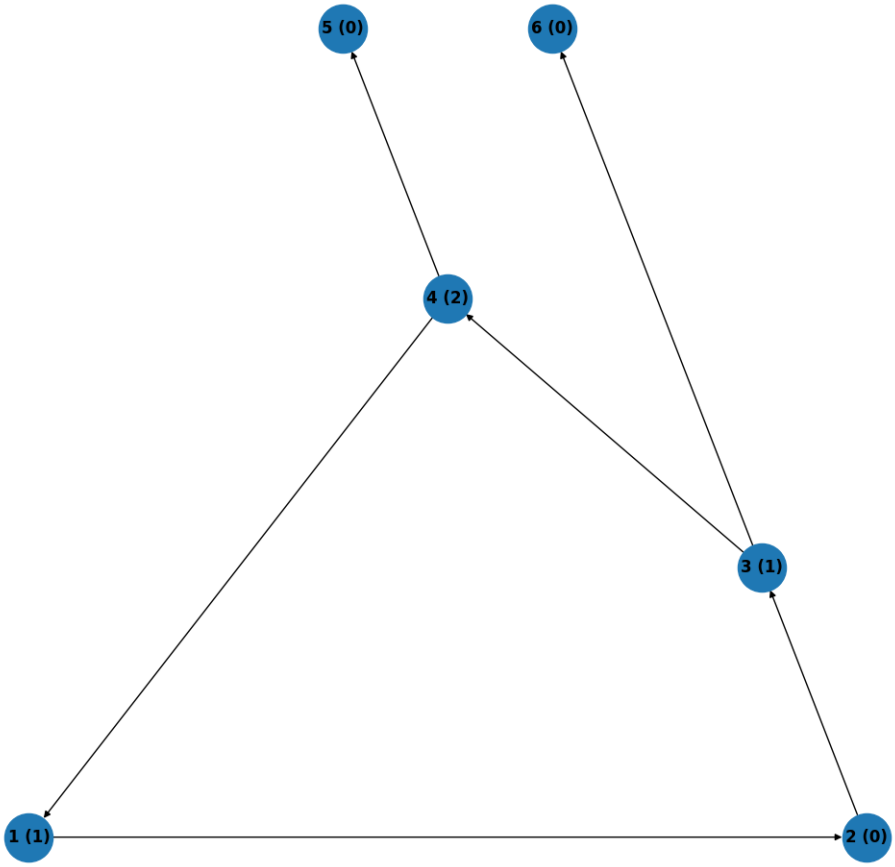
```
        level = []
        has_circuit = True
        for i in range(n):
            s = 0
            for j in range(n):
                if lmbd[j]!=0:
                    s+=m[i][j]
            new_lmbd[i] = s
            if lmbd[i] != 0 and s == 0:
                level.append(i)
                has_circuit=False
        if has_circuit: return False
        lmbd = new_lmbd
        levels.append(level)
    return levels

def grundy_from_levels(m, levels):
    n = len(m[0])
    grundy_values = [0]*n
    if len(grundy_values) > 1:
        for v in levels[1]:
            grundy_values[v] = 1
        for level in levels[2:]:
            for v in level:
                temp = [dv for i,dv in enumerate(grundy_values)
                           if m[v][i] == 1]
                temp = set(temp)
                mex = 0
                while mex in temp:
                    mex += 1
                grundy_values[v] = mex
    return grundy_values

def graph_cores(m):
    n = len(m)
    cnf1 = []
    for i in range(n):
        for j in range(n):
            if m[i][j] == 1:
                cnf1.append((i,j))

    cnf2 = []
    for i in range(n):
        dis = [i]
        for j in range(n):
            if m[i][j] == 1:
                dis.append(j)
```

```
        cnf2.append(dis)

    cores = []

    for line in product([0,1], repeat=n):
        res = True
        for dis in cnf1:
            res = res and (not line[dis[0]]
                           or not line[dis[1]])
        for dis in cnf2:
            res = res and any(line[i] for i in dis)
        if res == True:
            cores.append(line)

    for i in range(len(cores)):
        cores[i] = [j for j,v in enumerate(cores[i]) if v==1]
    return cores

def rec_grundy(matrix, vertex, core):
    if vertex in core:
        return 0
    values = [rec_grundy(matrix, i, core) for
              i in range(len(matrix)) if matrix[vertex][i]]
    result = 0
    while result in values:
        result += 1
    return result

def graph_display(m, grundy):
    G = nx.DiGraph()
    n = len(m[0])
    for i in range(n):
        G.add_node(str(i+1)+' ('+str(grundy[i]) +")")
        for j in range(n):
            if m[i][j] == 1:
                G.add_edge(str(i+1)+' ('+str(grundy[i]) + ')',
                           str(j+1)+' ('+str(grundy[j]) + ')')
    nx.draw_planar(G, with_labels=True, node_size=1300
                   ,font_weight="bold")
    plt.show()

def main():
    m = input_digraph()
    l = digraph_levels(m)
    n=len(m)
    if l:
        grundy = grundy_from_levels(m,l)
```

```
            graph_display(m, grundy)
        else:
            print("The digraph contains circuits, "
                  "proceed to the search for the cores")
            cores = graph_cores(m)
            if cores == []:
                print("Grundy's function is"
                      "invalid for this digraph")
            else:
                for core in cores:
                    grundy = [rec_grundy(m,i,core)
                              for i in range(n)]
                    graph_display(m, grundy)

if __name__ == "__main__":
    main()
```

3 Примеры

```
1  0 1 0 0 0 0
2  0 0 1 0 0 0
3  0 0 0 1 0 1
4  1 0 0 0 1 0
5  0 0 0 0 0 0
6  0 0 0 0 0 0
```



4 Оценка сложности

Для графа без контуров с количством вершин равным n алгоритм работает за $O(n^3)$, а для графа с контурами — за $O(2^n)$ (нахождение ядер графа — NP-полная задача, поэтому переборный алгоритм является разумным решением).