Курсовая работа по дискретной математике
Первая задача

Ахметшин Б.Р. – М8О-103Б-22 – 2 вариант

Март, 2023

Дано

Матрица смежности орграфа

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

Найти

1. матрицу односторонней связности
2. матрицу сильной связности
3. компоненты сильной связности
4. матрицу контуров

Решение

1.

1. Найдем матрицу односторонней связности при помощи первого алгортима Уоршалла:

$$T = E \vee A \vee ... \vee A^{n-1}$$

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

(b) $A^2 = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

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

(c) $A^3 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

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

2. Найдем матрицу односторонней связности при помощи итеративного алгортима Уоршалла:

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

(b) $T^{(1)} = ||t_{ij}^{(1)}||, t_{ij}^{(1)} = t_{ij}^{(0)} \vee (t_{i1}^{(0)} \& t_{1j}^{(0)}) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$

(c) $T^{(2)} = ||t_{ij}^{(2)}||, t_{ij}^{(2)} = t_{ij}^{(1)} \vee (t_{i2}^{(1)} \& t_{2j}^{(1)}) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$

(d) $T^{(3)} = ||t_{ij}^{(3)}||, t_{ij}^{(3)} = t_{ij}^{(2)} \vee (t_{i3}^{(2)} \& t_{3j}^{(2)}) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$

(e) $T^{(4)} = ||t_{ij}^{(4)}||, t_{ij}^{(4)} = t_{ij}^{(3)} \vee (t_{i4}^{(3)} \& t_{4j}^{(3)}) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$

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

2.

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

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

3.

Вершины в первой строке $\overline{S}$ соотвествуют первой компоненте сильной связности, следовательно первая компонента сильной связности – $\{v_1, v_3\}$ $\Rightarrow$ $\overline{S_1} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$, вторая компонента – $\{v_2, v_4\}$

Ответ: $\{v_1, v_3\}, \{v_2, v_4\}$

4.

Матрица контуров вычисляется как: $\overline{S}\&A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

Ответ: $\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

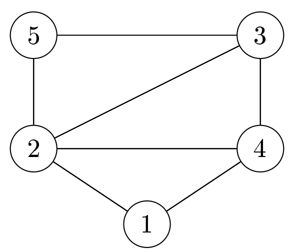
Курсовая работа по дискретной математике
Вторая задача

Ахметшин Б.Р. – М8О-103Б-22 – 2 вариант

Март, 2023

Дано

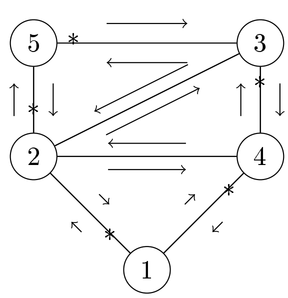
Граф:



Задание

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

Решение



Ответ

В итоге получился такой путь: 2 – 1 – 4 – 3 – 5 – 2 – 4 – 2 – 3 – 2 – 5 – 3 – 4 – 1 – 2

Курсовая работа по дискретной математике
Третья задача

Ахметшин Б.Р. – М8О-103Б-22 – 2 вариант

Март, 2023

Дано:

Матрица смежности орграфа:

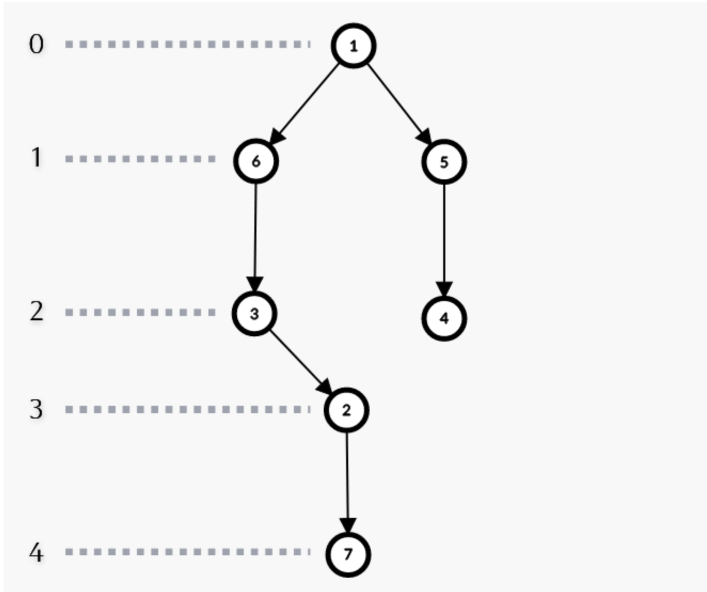
$$G = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Найти:

Используя алгоритм "фронта волны", найти все минимальные пути из первой вершины в последнюю.

Решение:

1. Помечаем вершину ν_1 индексом 0. Вершина ν_1 принадлежит фронту волны 0 уровня $W_0(\nu_1)$.
2. Вершины из множества $\Gamma_{W_0(\nu_1)} = \{\nu_5, \nu_6\}$ помечаем индексом 1, они принадлежат фронту волны 1 уровня $W_1(\nu_1)$.
3. Непомеченные ранее вершины из множества $\Gamma_{W_1(\nu_1)} = \Gamma\{\nu_5, \nu_6\} = \{\nu_3, \nu_4\}$ помечаем индексом 2, они принадлежат фронту волны 2 уровня $W_2(\nu_1)$.
4. Непомеченные ранее вершины из множества $\Gamma_{W_2(\nu_1)} = \Gamma\{\nu_3, \nu_4\} = \{\nu_2\}$ помечаем индексом 3, они принадлежат фронту волны 3 уровня $W_3(\nu_1)$.
5. Непомеченные ранее вершины из множества $\Gamma_{W_3(\nu_1)} = \Gamma\{\nu_2\} = \{\nu_7\}$ помечаем индексом 4, они принадлежат фронту волны 4 уровня $W_4(\nu_1)$.
6. Итак, вершина ν_7 достигнута, помечена индексом 4, следовательно, длина кратчайшего пути из ν_1 в ν_7 равна 4.



Теперь найдем все кратчайшие пути:

1. ν_7
2. $w_3(\nu_1) \cap \Gamma^{-1}_{\nu_7} = \{\nu_2\} \cap \{\nu_2\} = \{\nu_2\}$
3. $w_2(\nu_1) \cap \Gamma^{-1}_{\nu_2} = \{\nu_3, \nu_4\} \cap \{\nu_3, \nu_7\} = \{\nu_3\}$
4. $w_1(\nu_1) \cap \Gamma^{-1}_{\nu_3} = \{\nu_5, \nu_6\} \cap \{\nu_2, \nu_4, \nu_6\} = \{\nu_6\}$
5. $w_0(\nu_1) \cap \Gamma^{-1}_{\nu_6} = \{\nu_1\} \cap \{\nu_1, \nu_4, \nu_5\} = \{\nu_1\}$

Кратчайший путь один: $\nu_1 - \nu_6 - \nu_3 - \nu_2 - \nu_7$.

Курсовая работа по дискретной математике
Четвертая задача

Ахметшин Б.Р. – М8О-103Б-22 – 2 вариант

Апрель, 2023

Дано:

Матрица длин дуг нагруженого орграфа:

$$G = \begin{pmatrix} \infty & 3 & 5 & 6 & \infty & \infty & \infty & \infty \\ 2 & \infty & 1 & 2 & \infty & \infty & \infty & \infty \\ \infty & 1 & \infty & \infty & 3 & \infty & \infty & \infty \\ 3 & \infty & \infty & \infty & 4 & 7 & \infty & 9 \\ 5 & \infty & \infty & 4 & \infty & \infty & 4 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 1 & 2 \\ 7 & \infty & \infty & \infty & \infty & 1 & \infty & 2 \\ 8 & \infty & \infty & 13 & \infty & \infty & \infty & \infty \end{pmatrix}$$

Задание

Используя алгоритм Форда, найти минимальные пути из первой вершины во все достижимые вершины в нагруженном графе, заданном матрицей длин дуг A.

Решение

	$V1$	$V2$	$V3$	$V4$	$V5$	$V6$	$V7$	$V8$	$\lambda_i^{(0)}$	$\lambda_i^{(1)}$	$\lambda_i^{(2)}$	$\lambda_i^{(3)}$	$\lambda_i^{(4)}$	$\lambda_i^{(5)}$	$\lambda_i^{(6)}$	$\lambda_i^{(7)}$
$V1$	∞	3	5	6	∞	∞	∞	∞	0	0	0	0	0	0	0	0
$V2$	2	∞	1	2	∞	∞	∞	∞	∞	3	3	3	3	3	3	3
$V3$	∞	1	∞	∞	3	∞	∞	∞	∞	5	4	4	4	4	4	4
$V4$	3	∞	∞	∞	4	7	∞	9	∞	6	5	5	5	5	5	5
$V5$	5	∞	∞	4	∞	∞	4	∞	∞	∞	8	7	7	7	7	7
$V6$	∞	∞	∞	∞	∞	∞	1	2	∞	∞	13	12	12	12	12	12
$V7$	7	∞	∞	∞	∞	1	∞	2	∞	∞	∞	12	11	11	11	11
$V8$	8	∞	∞	13	∞	∞	∞	∞	∞	∞	15	14	14	13	13	13

1. Из v_1 в v_2 - $v_1 - v_2$, длина равна 3
   (a) $\lambda_1^{(0)} + c_{12} = 0 + 3 = \lambda_2^{(1)}$
2. Из v_1 в v_3 - $v_1 - v_2 - v_3$, длина равна 4
   (a) $\lambda_1^{(0)} + c_{12} = 0 + 3 = \lambda_2^{(1)}$
   (b) $\lambda_2^{(1)} + c_{23} = 3 + 1 = \lambda_3^{(2)}$
3. Из v_1 в v_4 - $v_1 - v_2 - v_4$, длина равна 5

(a) $\lambda_1^{(0)}+c_{12} = 0+3 =\lambda_2^{(1)}$

(b) $\lambda_2^{(1)}+c_{24} = 3+2 =\lambda_4^{(2)}$

4. Из v_1 в v_5 - $v_1 - v_2 - v_3 - v_5$, длина равна 7

   (a) $\lambda_1^{(0)}+c_{12} = 0+3 =\lambda_2^{(1)}$

   (b) $\lambda_2^{(1)}+c_{23} = 3+1 =\lambda_3^{(2)}$

   (c) $\lambda_3^{(2)}+c_{35} = 4+3 =\lambda_5^{(3)}$

5. Из v_1 в v_5 - $v_1 - v_2 - v_4 - v_6$, длина равна 12

   (a) $\lambda_1^{(0)}+c_{12} = 0+3 =\lambda_2^{(1)}$

   (b) $\lambda_2^{(1)}+c_{24} = 3+2 =\lambda_4^{(2)}$

   (c) $\lambda_4^{(2)}+c_{46} = 5+7 =\lambda_6^{(3)}$

6. Из v_1 в v_7 - $v_1 - v_2 - v_3 - v_5 - v_7$, длина равна 11

   (a) $\lambda_1^{(0)}+c_{12} = 0+3 =\lambda_2^{(1)}$

   (b) $\lambda_2^{(1)}+c_{23} = 3+1 =\lambda_3^{(2)}$

   (c) $\lambda_3^{(2)}+c_{35} = 4+3 =\lambda_5^{(3)}$

   (d) $\lambda_5^{(3)}+c_{57} = 7+4 =\lambda_7^{(4)}$

7. Из v_1 в v_8 - $vv_1 - v_2 - v_3 - v_5 - v_7 - v_8$, длина равна 13

   (a) $\lambda_1^{(0)}+c_{12} = 0+3 =\lambda_2^{(1)}$

   (b) $\lambda_2^{(1)}+c_{23} = 3+1 =\lambda_3^{(2)}$

   (c) $\lambda_3^{(2)}+c_{35} = 4+3 =\lambda_5^{(3)}$

   (d) $\lambda_5^{(3)}+c_{57} = 7+4 =\lambda_7^{(4)}$

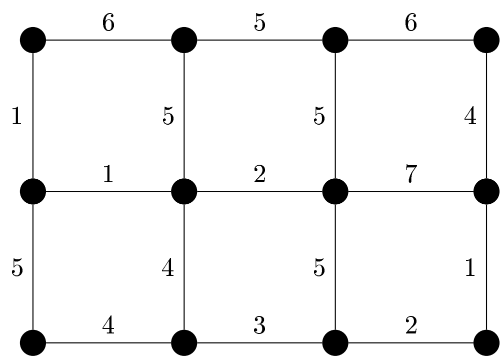
   (e) $\lambda_7^{(4)}+c_{78} = 11+2 =\lambda_8^{(5)}$

Курсовая работа по дискретной математике
Пятая задача

Ахметшин Б. Р. – М8О-103Б-22 – 2 вариант

Май, 2023

Дано

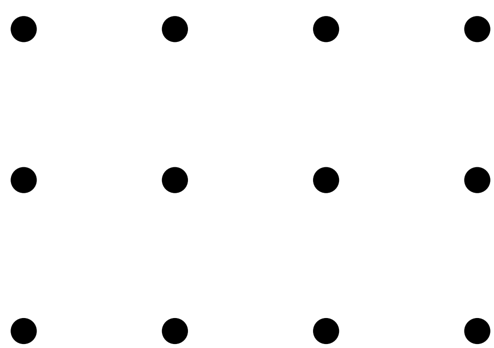


Задание

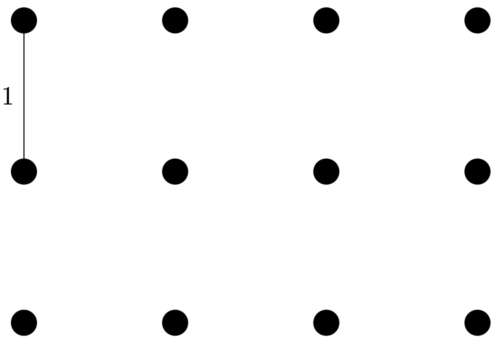
Найти остовное дерево с минимальной суммой длин входящих в него ребер

Решение

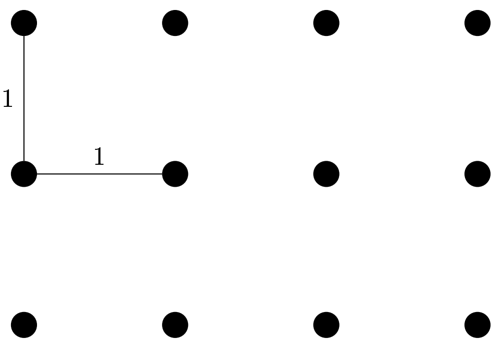
1



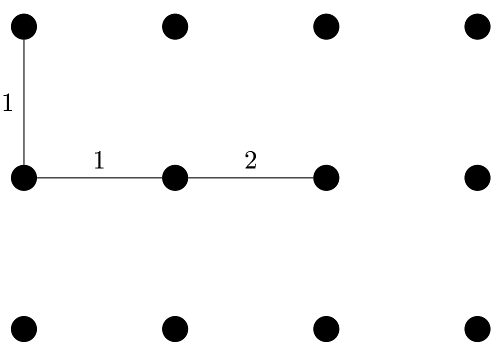
2



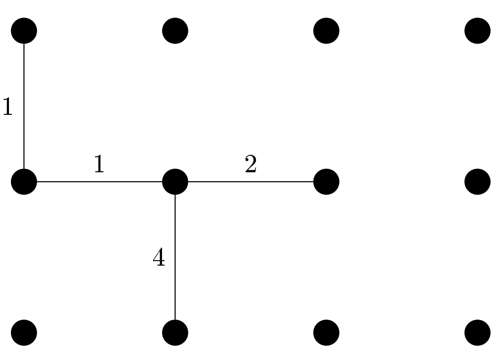
3



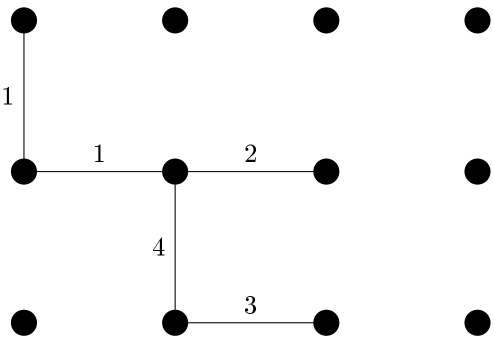
4



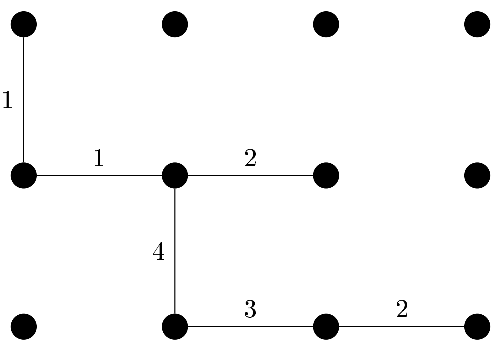
5



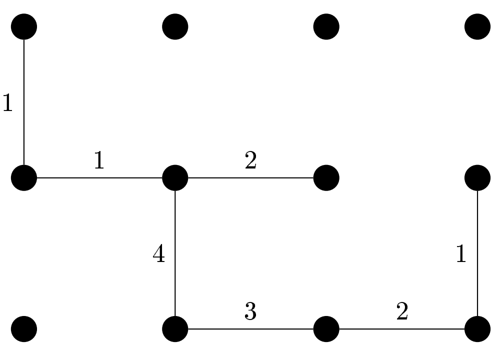
6



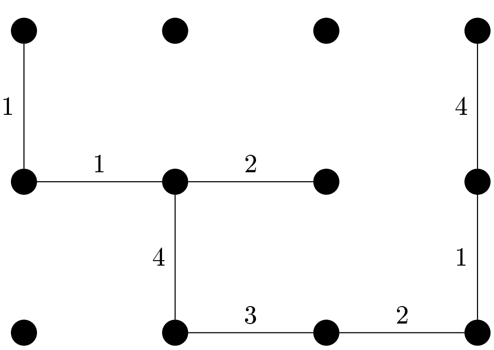
7



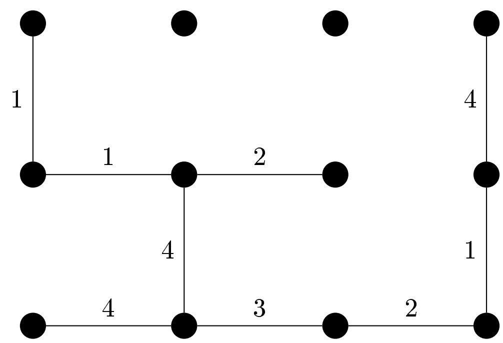
8



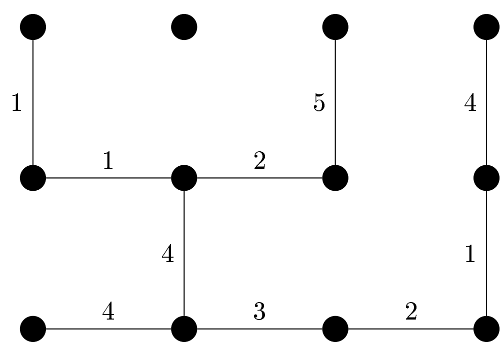
9



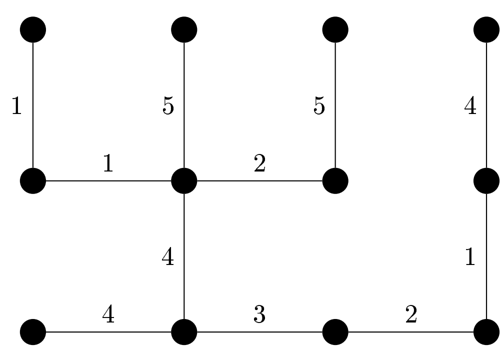
10



11

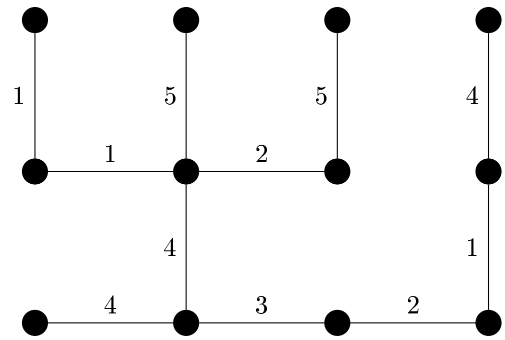


11



$L = 1 + 1 + 5 + 2 + 5 + 4 + 4 + 3 + 2 + 1 + 4 = 32$

Ответ



$L = 1 + 1 + 5 + 2 + 5 + 4 + 4 + 3 + 2 + 1 + 4 = 32$

Курсовая работа по дискретной математике
Шестая задача

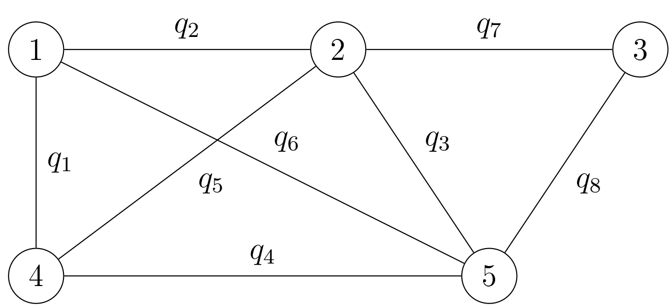
Ахметшин Б. Р. – М8О-103Б-22 – 2 вариант

Май, 2023

Задача

Пусть каждому ребру неориентированного графа соответствует некоторый элемент электрической цепи. Составить линейно независимые 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 систему уравнений для токов.

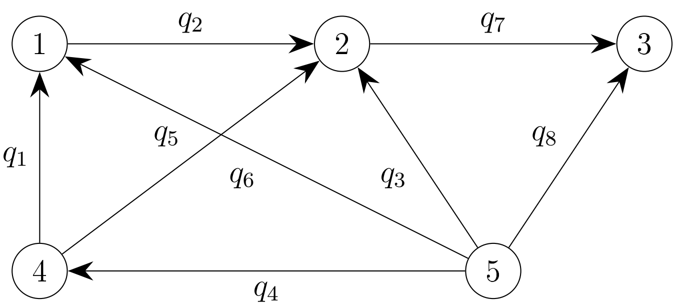
Дано



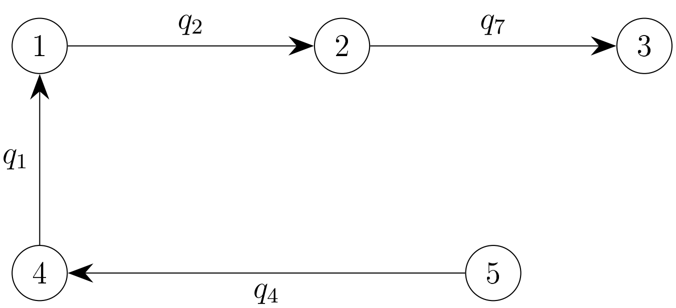
$$U = \begin{pmatrix} E_1 \\ U_2 \\ U_3 \\ U_4 \\ E_5 \\ U_6 \\ U_7 \\ U_8 \end{pmatrix}$$

Решение

1. Зададим ориентацию



2. Построим остовное дерево D



3. Добавляем по одному ребру из графа, получаем ровно один простой цикл. Записываем соответствующие вектор циклы.

$(D+q_5)$ μ_1: $v_4 - v_1 - v_2 - v_4 \to C(\mu_1) = (1,1,0,0,-1,0,0,0)$
$(D+q_6)$ μ_2: $v_5 - v_4 - v_1 - v_5 \to C(\mu_2) = (1,0,0,1,0,-1,0,0)$
$(D+q_3)$ μ_3: $v_5 - v_4 - v_1 - v_2 - v_5 \to C(\mu_3) = (1,1,-1,1,0,0,0,0)$
$(D+q_8)$ μ_4: $v_5 - v_4 - v_1 - v_2 - v_3 - v_5 \to C(\mu_4) = (1,1,0,1,0,0,1,-1)$

$$\Rightarrow C = \begin{pmatrix} 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & -1 \end{pmatrix}$$

4. По закону Кирхгофа для напряжений $C \times U = 0$

$$\begin{pmatrix} 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & -1 \end{pmatrix} \times \begin{pmatrix} E_1 \\ U_2 \\ U_3 \\ U_4 \\ E_5 \\ U_6 \\ U_7 \\ U_8 \end{pmatrix} = \begin{pmatrix} E_1 + U_2 - E_5 \\ E_1 + U_4 - U_6 \\ E_1 + U_2 - U_3 + U_4 \\ E_1 + U_2 + U_4 + U_7 - U_8 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\Rightarrow \begin{cases} E_1 + U_2 - E_5 = 0 \\ E_1 + U_4 - U_6 = 0 \\ E_1 + U_2 - U_3 + U_4 = 0 \\ E_1 + U_2 + U_4 + U_7 - U_8 = 0 \end{cases} \Leftrightarrow \begin{cases} U_2 = E_5 - E_1 \\ U_4 = U_6 - E_1 \\ U_3 = E_5 + U_4 \\ U_7 = U_8 - U_6 + E_1 - E_5 \end{cases}$$

5. Находим матрицу инцидентности B

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

6. По закону Кирхгофа для токов $B \times I = 0$

$$
\begin{pmatrix} 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ -1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \\ I_7 \\ I_8 \end{pmatrix} = \begin{pmatrix} I_1 - I_2 + I_5 \\ I_2 + I_3 + I_5 - I_7 \\ I_7 + I_8 \\ -I_1 + I_4 - I_5 \\ I_3 + I_4 + I_6 + I_8 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
$$

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

с учетом того, что $rgB = 4$.

7. Вместе с законом Ома получается система:

$$
\begin{cases} I_2 R_2 = E_5 - E_1 \\ I_4 R_4 = I_6 R_6 - E_1 \\ I_3 R_3 = E_5 + I_4 \\ I_7 R_7 = I_8 R_8 - I_6 R_6 + E_1 - E_5 \\ I_1 - I_2 + I_5 = 0 \\ I_2 + I_3 + I_5 - I_7 = 0 \\ I_7 + I_8 = 0 \\ -I_1 + I_4 - I_5 = 0 \end{cases}
$$

Курсовая работа по дискретной математике
Седьмая задача

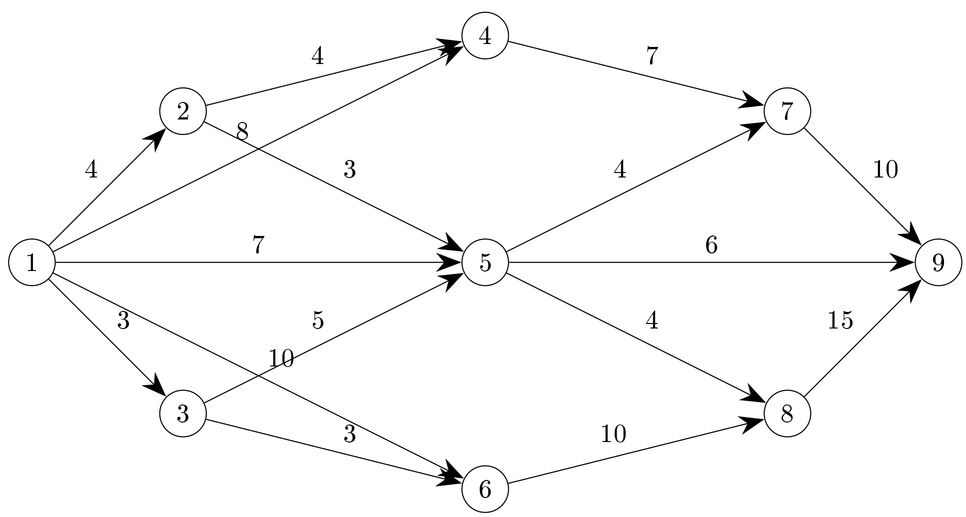
Ахметшин Б. Р. – М8О-103Б-22 – 2 вариант

Май, 2023

Задача

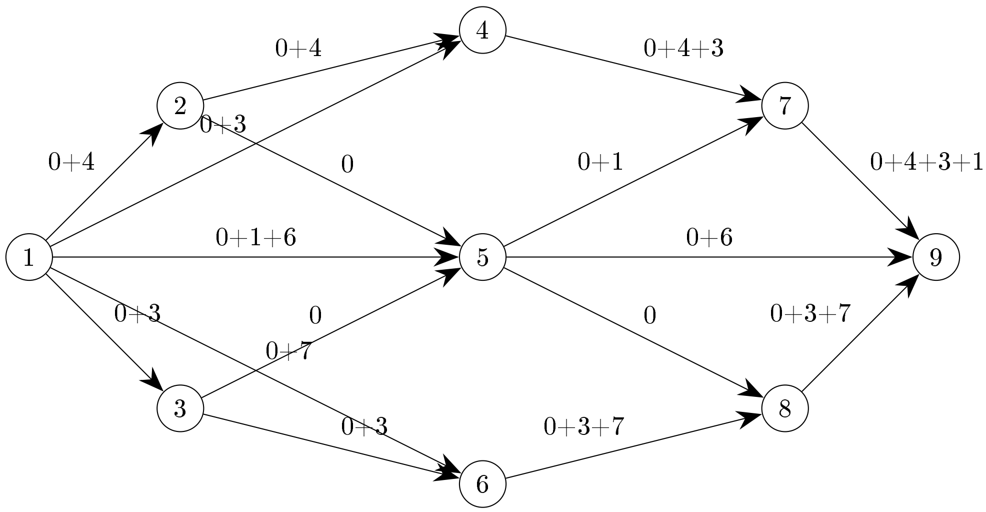
Построить максимальный поток по данной транспортной сети.

Дано



Решение

1. Построим полный поток



   (a) $v_1 - v_2 - v_4 - v_7 - v_9$
   $min\{4, 4, 7, 10\} = 4$

   (b) $v_1 - v_4 - v_7 - v_9$
   $min\{8, 7 - 4, 10 - 4\} = 3$

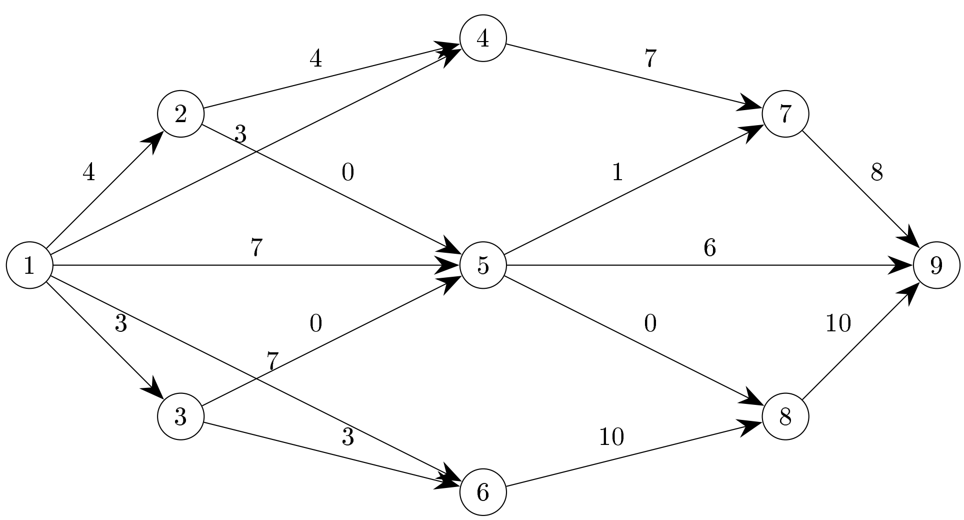
   (c) $v_1 - v_5 - v_7 - v_9$
   $min\{7, 4, 10 - 4 - 3\} = 1$

   (d) $v_1 - v_5 - v_9$
   $min\{7 - 1, 6\} = 6$

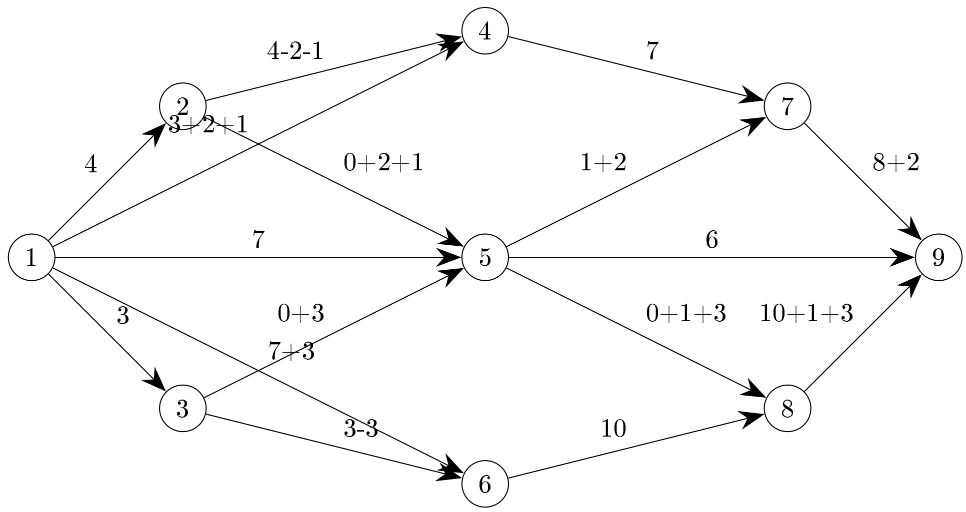
   (e) $v_1 - v_3 - v_6 - v_8 - v_9$
   $min\{3, 3, 10, 15\} = 3$

   (f) $v_1 - v_6 - v_8 - v_9$
   $min\{10, 10 - 3, 15 - 3\} = 7$

   Получился полный поток:



2. Построим максимальный поток



Найдем увеличивающие цепи:

(a) $v_1 - v_4 - v_2 - v_5 - v_7 - v_9$
$min\{8-3, \underline{4}, 3, 4-1, 10-8\} = 2$

(b) $v_1 - v_4 - v_2 - v_5 - v_8 - v_9$
$min\{8-3-2, \underline{4-2}, 3-2, 4, 15-10\} = 1$

(c) $v_1 - v_6 - v_3 - v_5 - v_8 - v_9$
$min\{10-7, \underline{3}, 5, 4-1, 15-10-1\} = 3$

Больше увеличивающих цепей нет.

Ответ

$\Phi_{max} = 10 + 6 + 14 = 30$

Московский авиационный институт
(национальный исследовательский университет)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

КУРСОВОЙ ПРОЕКТ
по курсу "Дискретная математика"
II семестр
на тему «Эйлеровы и гамильтоновы пути (циклы)»

Студент: Ахметшин Б.Р.
Группа: М8О-103Б-22, № 2
Руководитель: Яшина Н. П., доцент 805 кафедры

Москва, 2023

Contents

Москва, 2023

Теоретические сведения

Задан произвольный неориентированный псевдограф, требуется отыскать эйлеровы и гамильтоновы пути и циклы, если таковые имеются. (Далее речь будет идти только о неориентированных псевдографах)

Пусть *G* — произвольный псевдограф. Путь (цикл) в *G* называется эйлеровым, если по каждому ребру *G* он проходит ровно один раз, и гамильтоновым, если он проходит через каждую вершину *G* единожды.

Псевдограф, содержащий эйлеровый цикл называется эйлеровым, а содержащий эйлеровый путь — полуэйлеровым. Аналогично определяется гамильтоновый и полугамильтоновый псевдографы. Очевидно, что эйлеровым и/или гамильтоновым путем (циклом) могут обладать лишь связные псевдографы, что далее будет предполагаться в формулировках.

Теорема 1: Критерий существования эйлерового цикла.

Связный псевдограф *G* содержит эйлеровый цикл тогда и только тогда, когда степени всех вершин *G* четны.

Теорема 2: Критерий существования эйлерового пути.

Связный псевдограф *G* содержит эйлеровый путь тогда и только тогда, когда вершин с нечетной степенью либо два, либо ноль.

Критерии и эффективные алгоритмы гамильтонового пути и цикла сложны и не приведены в методических материалах, поэтому в своей работе я принял решение искать гамильтоновы пути и циклы только в псевдографах, удовлетворяющих условие следующей теоремы, потому как в таких графах их поиск выполняется за полиномиальное время.

Теорема 3: Условие Дирака.

Связный граф *G* обладает гамильтоновым циклом, если $n \geq 3$, $\delta \geq n/2$, где δ — наименьшая степень вершины *G*.

Очевидно, что Теорема 3 верна также и для псевдографов. Из Теоремы 3 следует, что связный псевдограф, удовлетворяющий её условию, также содержит гамильтоновый путь:

> Добавим к *G* вершину R и ребра, соединяющие R со всеми остальными вершинами *G*. Т.к. получившийся граф удовлетворяет условию Т3, в нем есть гамильтоновый цикл. Вычленив вершину R из цикла, получим гамильтоновый путь.

Описание алгоритма

Поиск эйлерового пути.

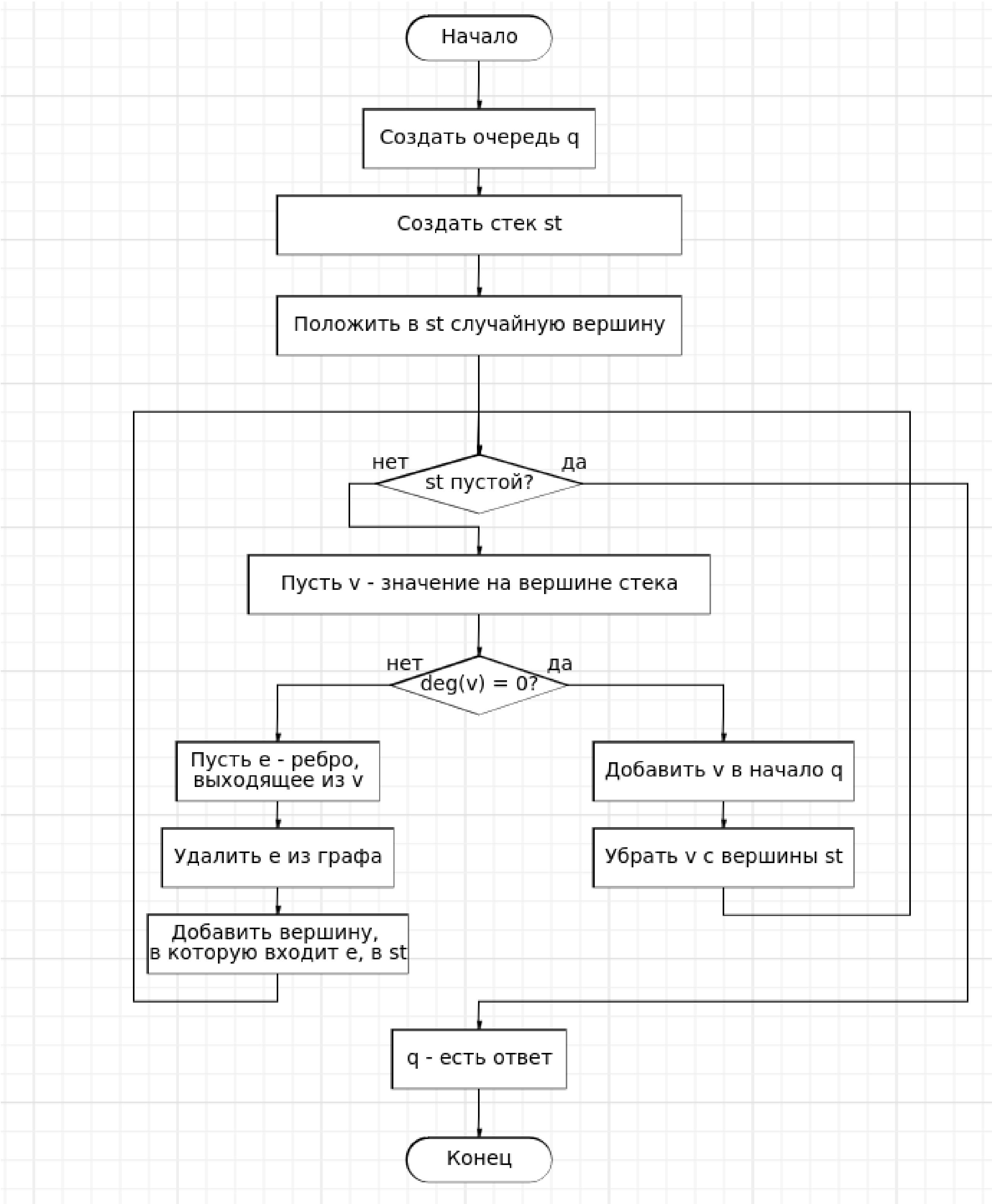
Если данный псевдограф *G* удовлетворяет условию T2, следующий алгоритм находит его:

Псевдокод:

```
St — стек;
в St кладём любую вершину (стартовая вершина);
пока St не пустой
      пусть V - значение на вершине St;
      если степень(V) = 0, то
            добавляем V к ответу;
            снимаем V с вершины St;
      иначе
            находим любое ребро, выходящее из V;
            удаляем его из графа;
            второй конец этого ребра кладём в St;
```

При условии, что каждое действие (на строчке псевдокода) выполняется за O(1), сложность алгоритма — O(m), где m — количество ребер псевдографа *G*.

Блок-схема:


Начало
Создать очередь q
Создать стек st
Положить в st случайную вершину
нет
да
st пустой?
Пусть v - значение на вершине стека
нет
да
deg(v) = 0?
Пусть e - ребро,
выходящее из v
Добавить v в начало q
Удалить e из графа
Убрать v с вершины st
Добавить вершину,
в которую входит e, в st
q - есть ответ
Конец

Поиск эйлерового цикла.

Пусть псевдограф G удовлетворяет условию Т1, тогда для нахождения эйлерового цикла достаточно вычленить из G произвольное ребро, тогда G будет удовлетворять условию Т2, следовательно, в нем найдется эйлеровый путь. Если начать его поиск с одной из вершин изъятого ребра и после дополнить другой, получится эйлеровый цикл. Сложность O(m).

Поиск гамильтонового цикла.

Из доказательства Т3 следует алгоритм нахождения гамильтонового цикла. Итак, «Поступим следующим образом: заведем очередь и положим в нее все вершины нашего графа (не важно в каком порядке). Пусть n — количество вершин псевдографа. Тогда $n(n - 1)$ раз будем делать следующую операцию:

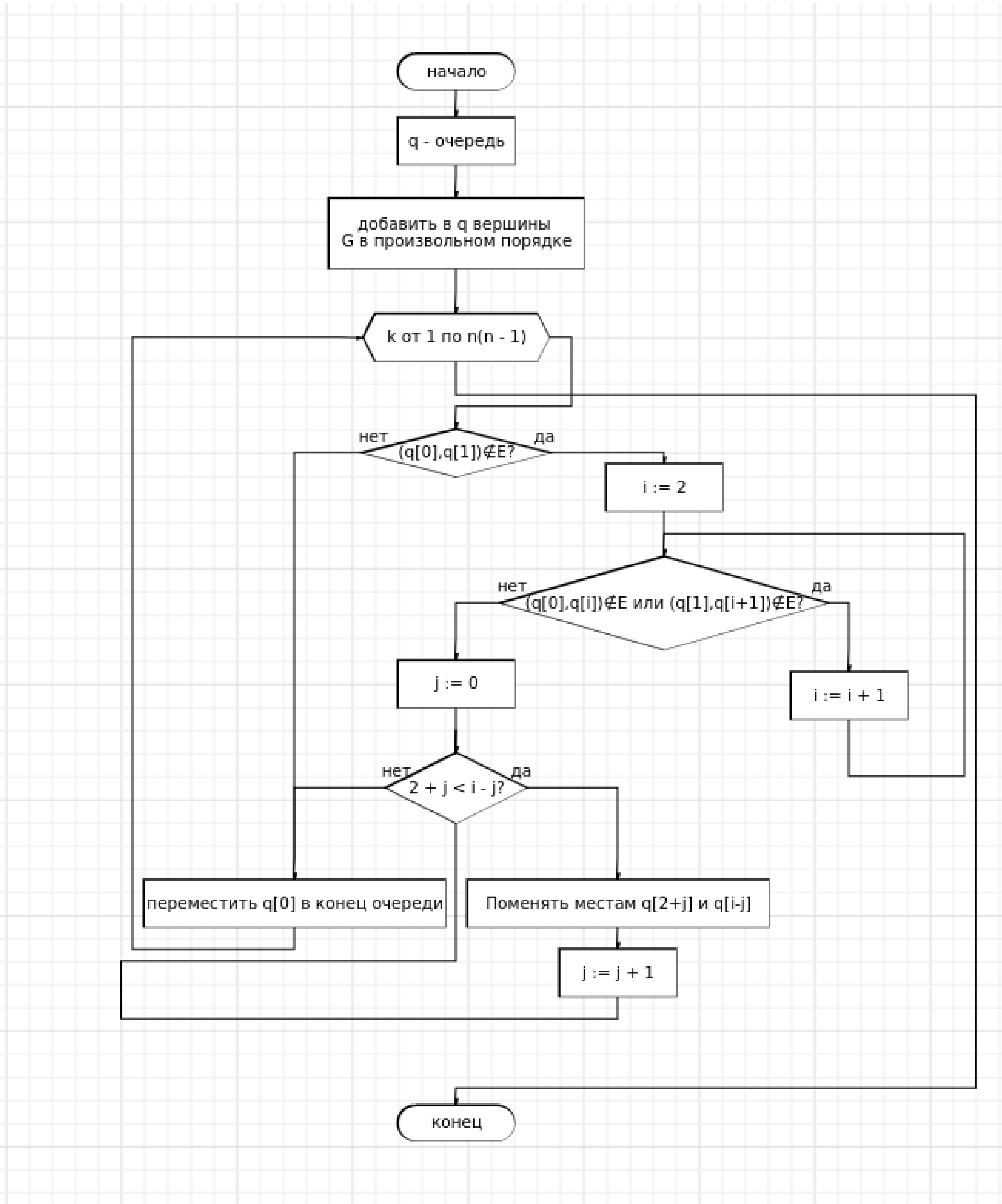
- Пусть v_1 — это голова очереди, v_2 — следующая за ней вершина и так далее. Если между первой и второй вершиной в очереди есть ребро в графе G, то перемещаем первую вершину в конец очереди и переходим к следующей итерации.
- Если между первой и второй вершиной в очереди ребра нет, то найдем вершину v_i где $i > 2$, такую что, ребра $v_1v_i, v_2v_{i+1} \in E$ (так как у нас для графа выполнена теорема Дирака, то такая вершина обязательно найдется). После чего поменяем в очереди местами вершины v_2 и v_i, v_3 и v_{i-1}, v_{2+j} и v_{i-j}, и так далее, пока $2 + j < i - j$ (то есть j пробегает все значения от 0 до значения заданного неравенством). Теперь у нас появилось ребро между первой и второй вершинами в очереди (теперь вторая вершина, это та, которая была до разворота на i-й позиции), а также, гарантированно существует ребро между i-й и $(i+1)$-й вершинами очереди. После этого, так же как и в первом случае, оправляем первую вершину в конец очереди.

Таким образом после n итераций, мы получаем последовательность (вершины лежащие в очереди), где любые 2 соседние вершины соединены ребром, все вершины графа находятся в этой последовательности, и более того, каждая ровно один раз, а также существует ребро между последней и первой вершинами очереди, а это и значит, что мы решили поставленную задачу.» - описание алгоритма нахождения гамильтонова цикла в условиях теорем Дирака и Оре по источнику **[3]**. Сложность: O(n(n-1)).

Псевдокод:

```
q - очередь;
добавить в q вершины G в произвольном порядке;
Повторить n(n - 1) раз:
      если (q0, q1) ∉ E:
            i := 2;
            Пока (q0, qi) ∉ E или (q1, qi+1) ∉ E:
                  i = i + 1;
            Поменять местами все пары (q2+j,qi-j) вершин \
                                  в q такие, что 2 + j < i - j;
      Переместить первую вершину в q в конец очереди
```

Блок-схема:



Москва, 2023

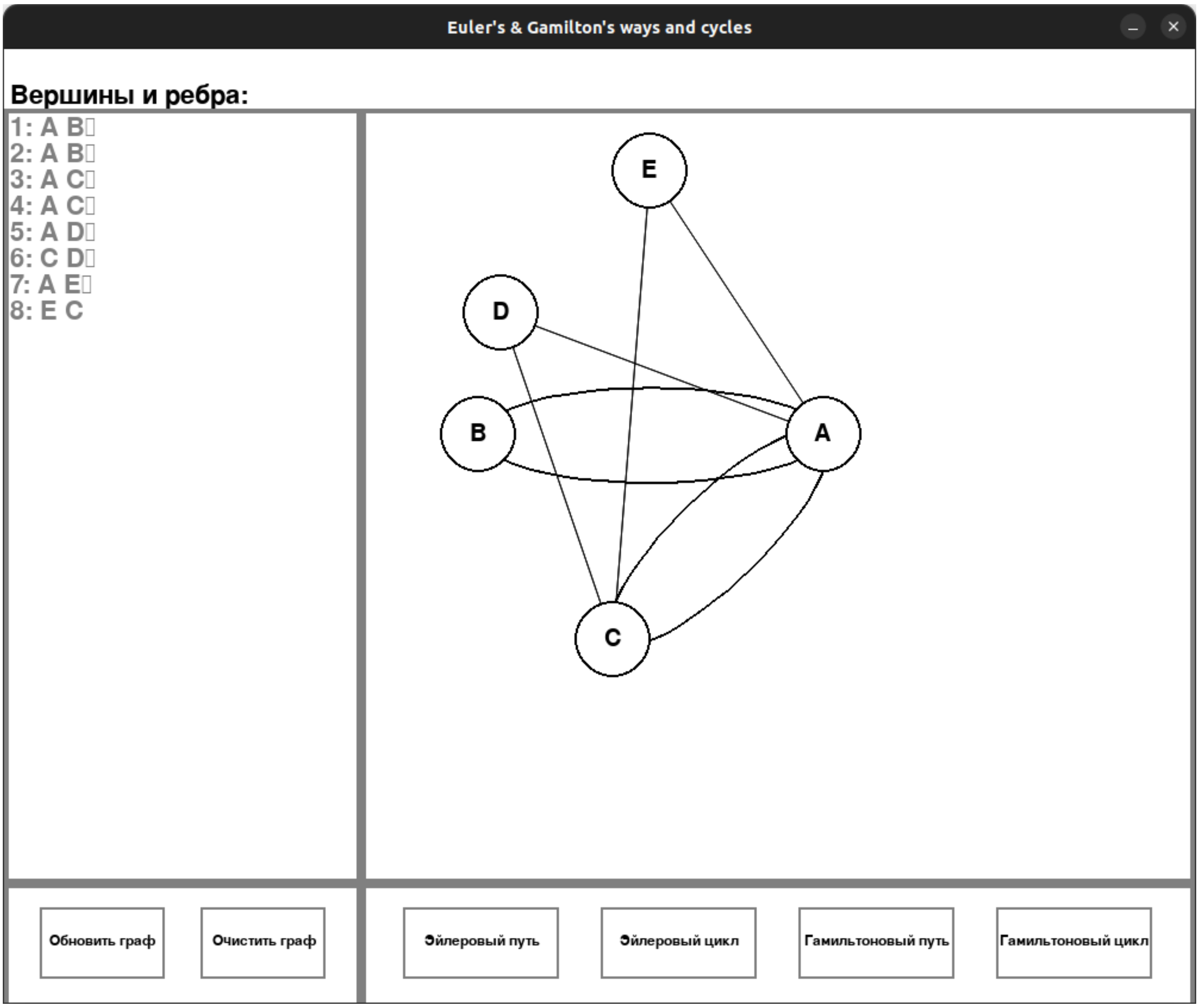
Поиск гамильтонового пути.

В конце пункта Теоретических сведений было указание для данного алгоритма. Еще раз: добавляем к псевдографу *G*, удовлетворяющему ТЗ, вершину *R*, соединенную ребрами со всеми остальными вершинами, получаем *G’*. Запустив предыдущий алгоритм, получим гамильтоновый цикл с вершиной *R*. Вычленив эту вершину из цикла, получим либо два гамильтоновых пути на подграфах *G*, тогда, совместив их начало и конец (за счет цикличности найдется ребро их соединяющее), получим гамильтоновый путь на *G*, либо один на подграфе *G*, совпадающем с ним самим, т. е. искомый путь. Сложность O(n(n - 1)).

Программная реализация

Описание программы

Программа и себя представляет графическое приложение, разделенное на окна.



Левое верхнее окно — поле для ввода вершин и ребер графа, допустимо указывать вершины в отдельной строке, тогда, если этой вершине не были даны ребра, она останется изолированной. Программа поддерживает произвольные псевдографы с любым числом вершин, ребер, в том числе кратных и петлевых.

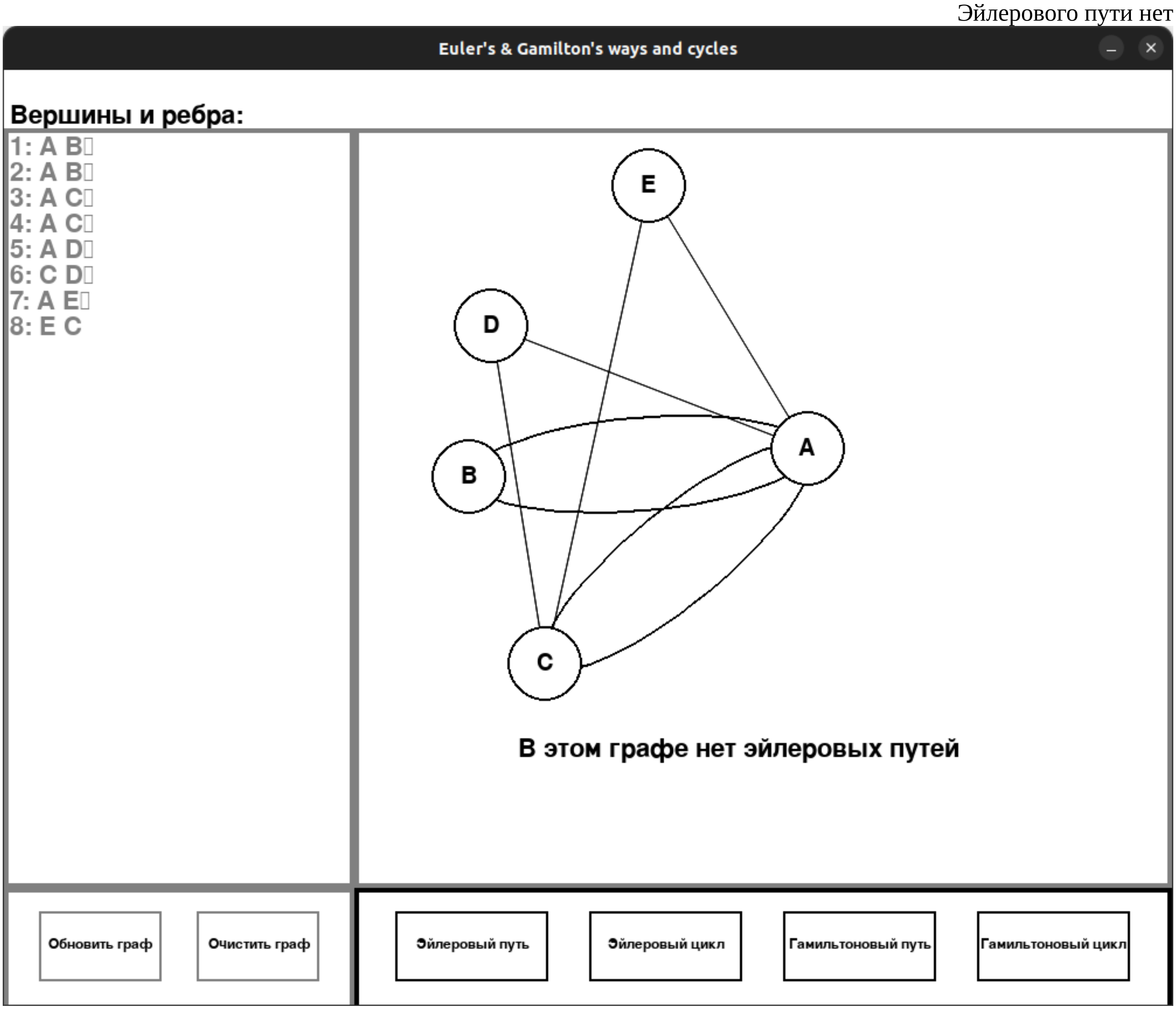
Слева снизу видно окно с кнопками «Update graph» и «Clear graph» - обновить граф и очистить граф соответственно. Первая кнопка обновляет граф согласно введенной информации в окне ввода, вторая очищает (удаляет) граф.

Наибольшее окно — окно отображения графа. В этой области будут отображаться вершины и ребра графа. Вершины графа подвижны и стремятся к стабильному состоянию, их начальное положение задается случайно, а далее силы притяжения и отталкивания придают им конечные положения, некоторые вершины так и остаются гармонически качаться из стороны в сторону.

Последнее — нижнее правое окно — окно визуализации обхода соотвествующих путей и цепей. Если для заданного графа существует эйлеровый путь или цикл, он будет выведен текстом в нижней части окна с графом, иначе будет выведено сообщение о том, что таковых пути или цепи не существует. Если заданный граф удовлетворяет Условию Дирака, то программа отображает последовательность вершин, соответствующую пути или циклу.

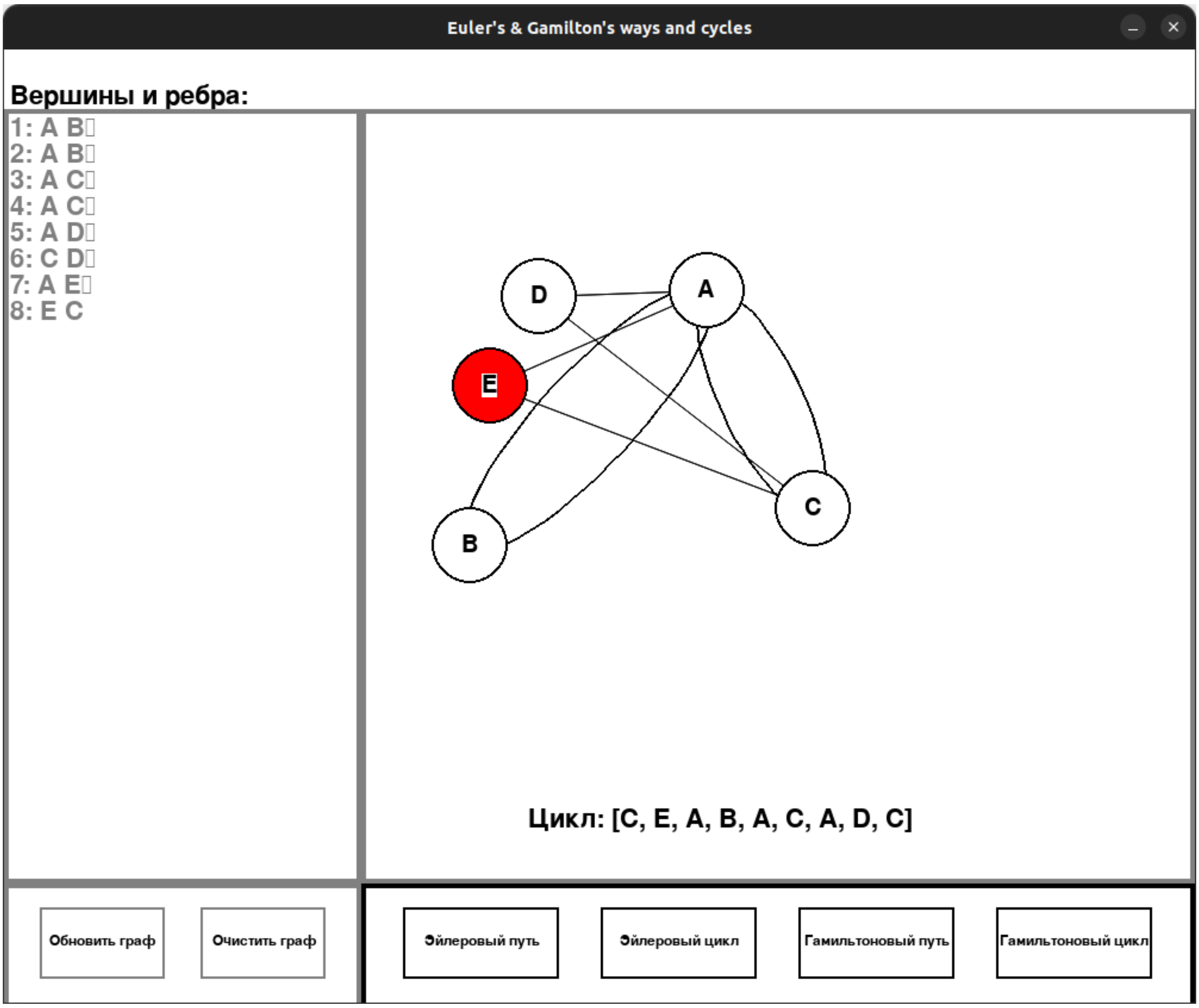
Дополнительно обход иллюстрирует покраска вершин в красный цвет в порядке пути (цикла).

Эйлерового пути нет



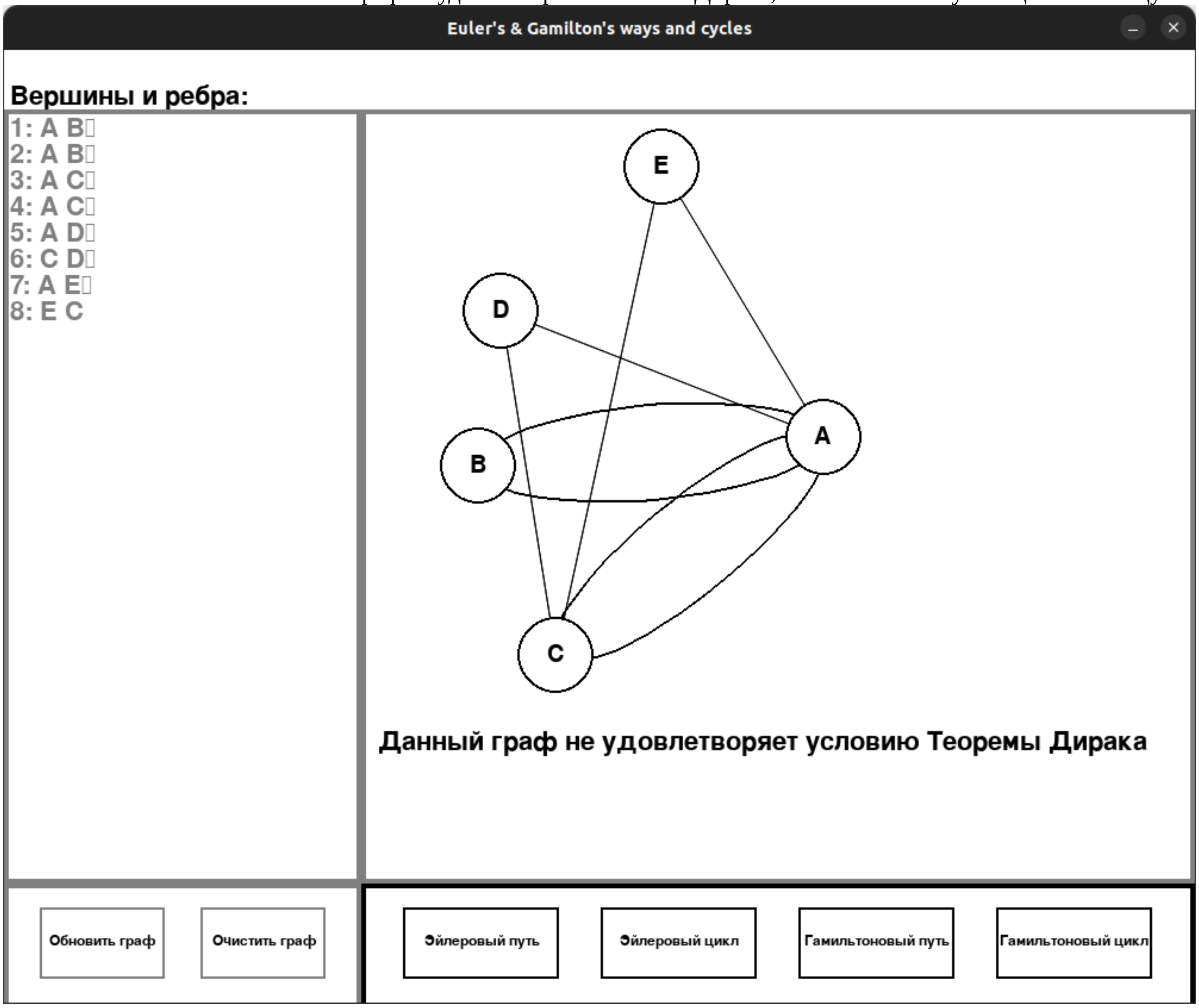
Москва, 2023

Обход эйлерового цикла



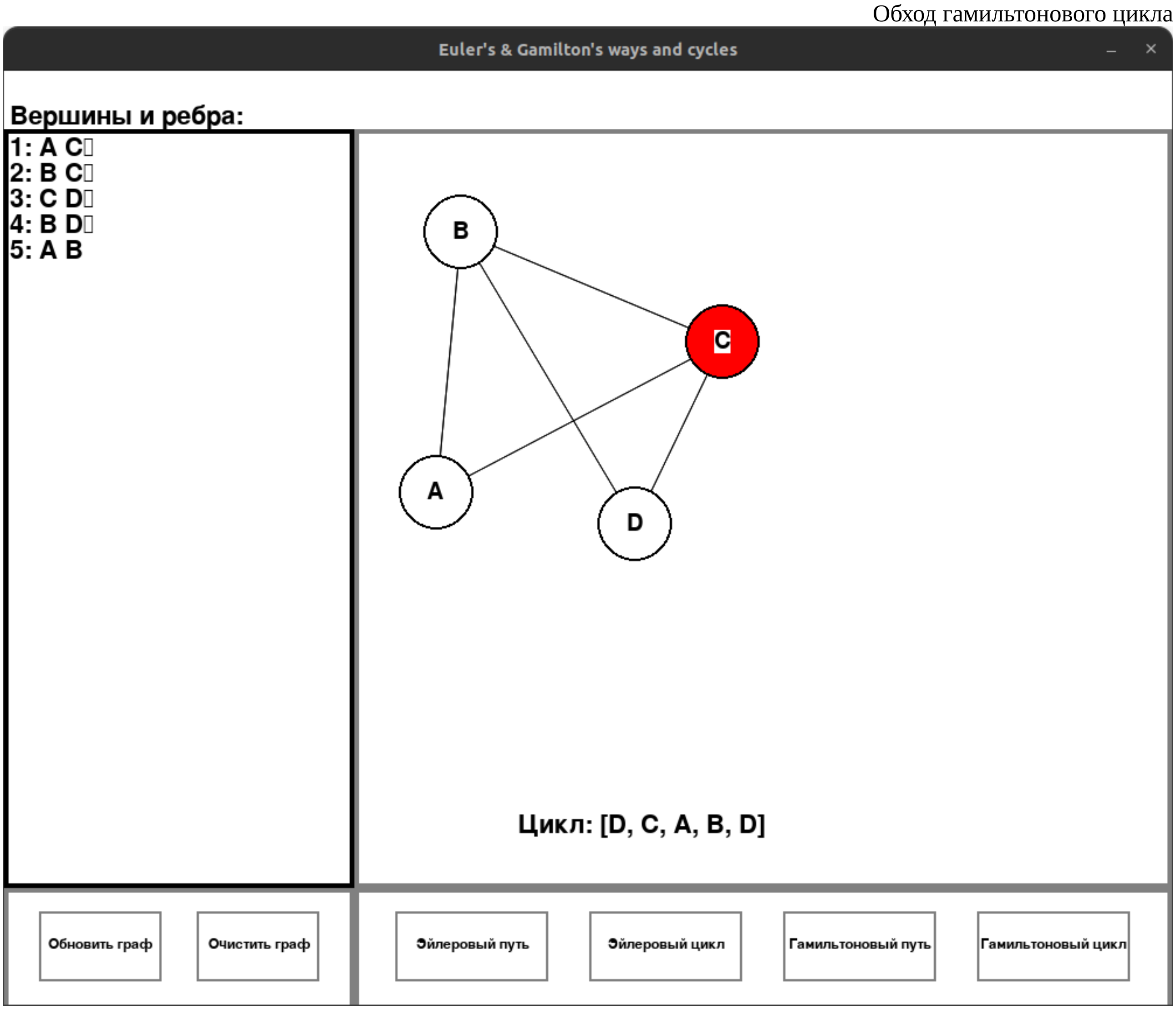
Москва, 2023

Граф не удовлетворяет Условию Дирака, гамильтоновы пути и циклы не ищутся



Москва, 2023

Обход гамильтонового цикла



Рассмотрим подробнее последний пример. В графе *G 4* вершины, наименьшая степень вершины *G — 2*, следовательно, по Теореме Дирака найдется гамильтоновый путь. Используем описанный выше алгоритм его нахождения:

Пусть $q := \{C,A,D,B\}$
Т.к. $(C,A) \in E \Rightarrow q := \{A,D,B,C\}$
$(A,D) \notin E \Rightarrow$
$i := 2$
$(A,B), (D, C) \in E \Rightarrow i = 2$
$j := 0$
$2 + 0 < 3 - 0 \Rightarrow q := \{A,B,D,C\}, j := 1$
$2 + 1 > 3 - 1 \Rightarrow q = \{A,B,D,C\}$
$q := \{B,D,C,A\}$
…

{B,D,C,A,B} — гамильтоновый цикл, а это значит, что алгоритм далее будет просто циклически сдвигать последовательность. В конце алгоритм дополнит путь q до цикла, выбрав последней вершиной *D*

Москва, 2023

Описание структур данных

Программа написана на языке Python. Выбор языка программирования обусловлен широкой пользовательской библиотекой и высокой скоростью разработки.

Логически граф представляется как упорядоченная пара двух множеств $G = <V, E>$, где V — множество вершин графа, E — множество ребер графа. Множества реализуются встроенным классом set. Физически граф представляет класс Graph, полями которого являются:

self.V — множество вершин;
self.E — множество ребер;
self.ortype — тип графа (ориентированный/неориентированный)
self.wtype — тип графа (взвешенный/невзвешенный);
self.delta — наименьшая степень вершин графа;
self.is_coherent — является-ли граф связным;
self.n — количество вершин;
self.m — количество ребер;
self.scale — масштаб визуализируемого графа;
self.size — визуальный размер вершин;
self.edges — границы для генерации положений вершин,

а методами:

def __init__(self, input, type, wtype, ortype, scale, size, edges) — конструктор класса
def clear(self) — очистить граф;
def copy(self) — копировать граф;
def find_edge(self, v, u, name, weighted, w) — найти ребро в графе, удовлетворяющее параметрам;
def find_vertex(self, v) — найти вершину, удовлетворяющую параметрам;
def generate_positions(self) — функция генерации позиций для вершин;
def copy_vertex(self) — скопировать вершины графа в отдельное множество;
def remove_edges(self, E) — вернуть новый граф без указанных ребер;
def remove_nodes(self, V) — вернуть новый граф без указанных вершин;
def remove_edges_update(self, E) — удалить указанные ребра из графа;
def remove_nodes_update(self, V) — удалить указанные вершины из графа;
def coherent(self) — проверить, является-ли граф связным;
def update_positions(self, dt) — обновить позиции вершин, учитывая их нынешние положения, скорости и ускорения за указанный промежуток времени;
def eq_classes(self) — вернуть разбиение множества ребер на классы эквивалентности по отношению e.vpe.v — e.v инцидентно e.u, где e — ребро, исходящее из e.v и входящее в e.u; (т.е. вернуть множество множеств ответствующих кратных ребер);
def eulers_way(self) — вернуть эйлеровый путь;
def eulers_cycle(self) — вернуть эйлеровый цикл;
def least_degree(self) — найти наименьшую степень вершин графа;
def is_gamiltons_graph(self) — проверка связности и Условия Дирака;
def gamils_cycle(self) — вернуть гамильтоновый цикл;
def gamils_way(self) — вернуть гамильтоновый путь;

и перегруженные операторы.

Вершины графа представляются классом Vertex, полями которого являются:

self.name — имя вершины;

self.R — визуальный радиус вершины;
self.r, self.v, self.a — координата, скорость и ускорение вершины, выражается классом Point;
self.color — визуальный цвет вершины;
self.index — индекс вершины (по порядку создания);
self.degree — степень вершины
Vertex.count — статическое поле, показывает, сколько всего вершины было создано,

и методы:

def __init__(self, name, R, degree, r, v, a, color) — конструктор класса;
def copy(self) — копия вершины;
def move(self, dt, edges, size) — передвинуть вершину в заданных границах (учитывая, что её размер - size), согласно её координатам, скорости и ускорению на время dt;
def pos(self) — координаты вершины;
и перегруженные операторы.

Ребра графа представляются классом Edge, полями которого являются:

self.name — имя ребра;
self.v, self.u — инцидентные вершины;
self.weighted — взвешенно-ли ребро;
self.w — вес ребра;
self.color — цвет ребра;
self.index — индекс ребра;
Edge.count — статическое поле, показывает, сколько ребер было создано,

и методы:

self.__init__(self, v, u, name, weighted, w, color) — конструктор класса;
self.copy() - копировать ребро, притом поля self.v, self.u указывают на те же вершины, а не на копии;
self.reverse() - вернуть копию развернутого ребра.

Класс Point.
поля:

self.x, self.y — координаты точки,

и методы:

def __init__(self, x, y) — конструктор класса;
def copy(self) — вернуть копию точки;
sef scalar(self, other) — скалярное произведение радиус-векторов на точках в стандартном базисе;
def len(self) — длина радиус-вектора на точке;
def pos(self) — вернуть пару (int(self.x), int(self.y)) — визуальные координаты;
и перегруженные операторы.

Реализация алгоритмов

Следующие реализации дословно повторяют описанные выше алгоритмы, уточненные на выбранные структуры данных.

Поиск эйлерового пути

```
def eulers_way(self, vs=None):
    if self.is_coherent == None and self.m > 0:
        self.is_coherent = self.coherent()
    if self.is_coherent and len([v for v in self.V if v.degree % 2 == 1]) != 2:
        return []
    else:
        eulers_way = []
        _G = self.copy()
        st = list()
        if vs is None:
            for v in _G.V:
                if v.degree % 2 == 1:
                    st.append(v)
                    break
        else:
            st.append(_G.find_vertex(vs))
        while len(st) > 0:
        v = st[-1]
        if v.degree == 0:
            eulers_way.append(v)
            st.pop(-1)
        else:
            for _e in _G.E:
                if v == _e.v:
                    e = _e
                    break
            st.append(e.u)
            _G.remove_edges_update(set([e]))
    return eulers_way
```

Москва, 2023

Поиск эйлерового цикла:

```
def eulers_cycle(self):
	if self.is_coherent == None and self.m > 0:
			self.is_coherent = self.coherent()
	 if self.is_coherent and len([v for v in self.V if v.degree % 2 == 1]) > 0:
			return []
	else:
			bridge = next(iter(self.E))
			v = bridge.v
			_G = self.remove_edges(set([bridge]))
			eulers_cycle = _G.eulers_way(bridge.u)
			eulers_cycle.append(v)
	return eulers_cycle
```

Поиск гамильтонового цикла:

```
def gamils_cycle(self):
	if not self.is_gamiltons_graph():
		return []
	q = list(self.V)
	for k in range(0, self.n * (self.n - 1)):
		if self.find_edge(q[0], q[1]) is None:
			i = 2
			while self.find_edge(q[0], q[i]) is None or \
				self.find_edge(q[1], q[i + 1]) is None:
				 i += 1
			j = 0
			while 1 + j < i - j:
				q[1 + j], q[i - j] = q[i - j], q[1 + j]
				j += 1
		q += [q[0]]
		q = q[1:]
	return [q[-1]] + q
```

Поиск гамильтонового пути:

```
def gamils_way(self):
	if not self.is_gamiltons_graph():
		return []
	_G = self.copy()
	v = Vertex('R')
	for u in _G.V:
		_G.E.update([Edge(v, u), Edge(u, v)])
	v.degree = _G.n
	_G.V.add(v)
	q = _G.gamils_cycle()[1:]
	i = 0
	while q[i] != v:
		i += 1
```

```
q = q[i+1:] + q[:i]
return q
```

Практическое применение

Помимо образовательных целей, эта программа может найти свое применение при построении туристических и проверочных маршрутов. Поиск эйлеровых путей и циклов может быть полезен, если целью экскурсии или туристического маршрута являются набережные или уникальные улицы какого-нибудь города, а поиск гамильтоновых путей и циклов, например, поможет тем же туристам построить маршрут, проходящий через каждый из интересующий их городов единожды.

Так же эти алгоритмы позволяют пользователю произвести перебор мест при поиске чего-либо. Допустим, менеджер офиса при переходе между кабинетами и этажами утерял свой пропуск. Вместо того, чтобы дожидаться помощи персонала, он может найти его самостоятельно, сделав эффективный перебор мест, в которых он был в последнее время.

Источники и полезные ссылки:

- Мой githab с исходным кодом приложения,
- Алгоритм нахождения эйлерова пути и цикла (maximal),
- Алгоритм нахождения гамильтонова пути и цепи (итмо),
- Статья Википедии про гамильтонов граф (wiki),
- Алгоритм посторения эйлерова пути и цикла (итмо),
- Вдохновитель идеи.

Москва, 2023