

Московский авиационный институт
(Национальный исследовательский университет)
факультет “Информационные технологии и прикладная математика”
кафедра “Математическая кибернетика”

КУРСОВАЯ РАБОТА
по курсу «Дискретная математика»
2-й семестр

Тема: Теория графов, алгебраические структуры, теория алгоритмов.
Реализация алгоритма Дейкстры для поиска кратчайших путей в
нагруженных графах.

Студент: Чекменев Вячеслав
Алексеевич

Группа: М8О-107Б

Руководитель: Н.С. Алексеев

Оценка: _____

Дата: _____

Введение.

В настоящем отчете по курсовой работе приведены результаты, полученные в рамках изучения курса “Дискретная математика” во втором учебном семестре.

В части I приведены решения типовых задач по теории графов, а также представлена реализация алгоритма Дейкстры для поиска кратчайших путей в нагруженных графах. Текст программы и тестовые примеры вынесены в Приложение.

В части II приведены решения типовых задач по темам “Алгебраические структуры” и “Теория алгоритмов”.

Часть I. Теория графов.

Задача А

Условие:

Ориентированный граф задан матрицей смежности:

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

По матрице смежности определите:

- имеются ли контуры;
- матрицу односторонней связности;
- матрицу сильной связности;
- компоненты сильной связности графа;
- постройте изображение графа.

$$a) A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}; A^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

На главной диагонали матрицы есть элементы $\neq 0 \Rightarrow$ есть контур длины 2

$$b) A^3 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

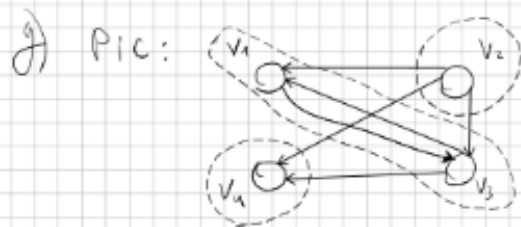
$$D = E V A V (A^2)^T V (A^3)^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} V \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} V \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0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

v_1, v_2, v_3, v_4

$$c) S = D \& D^T = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \& \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

v_1, v_2, v_3, v_4

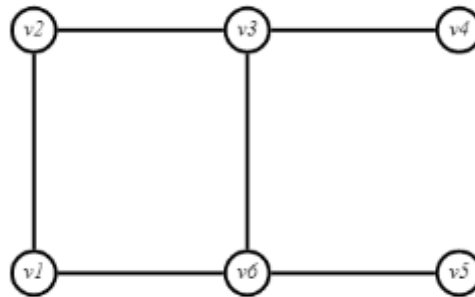
2) компоненты сильной связности: $K_1 = \{v_1, v_3\}; K_2 = \{v_2\}; K_3 = \{v_4\}$



Задача Б

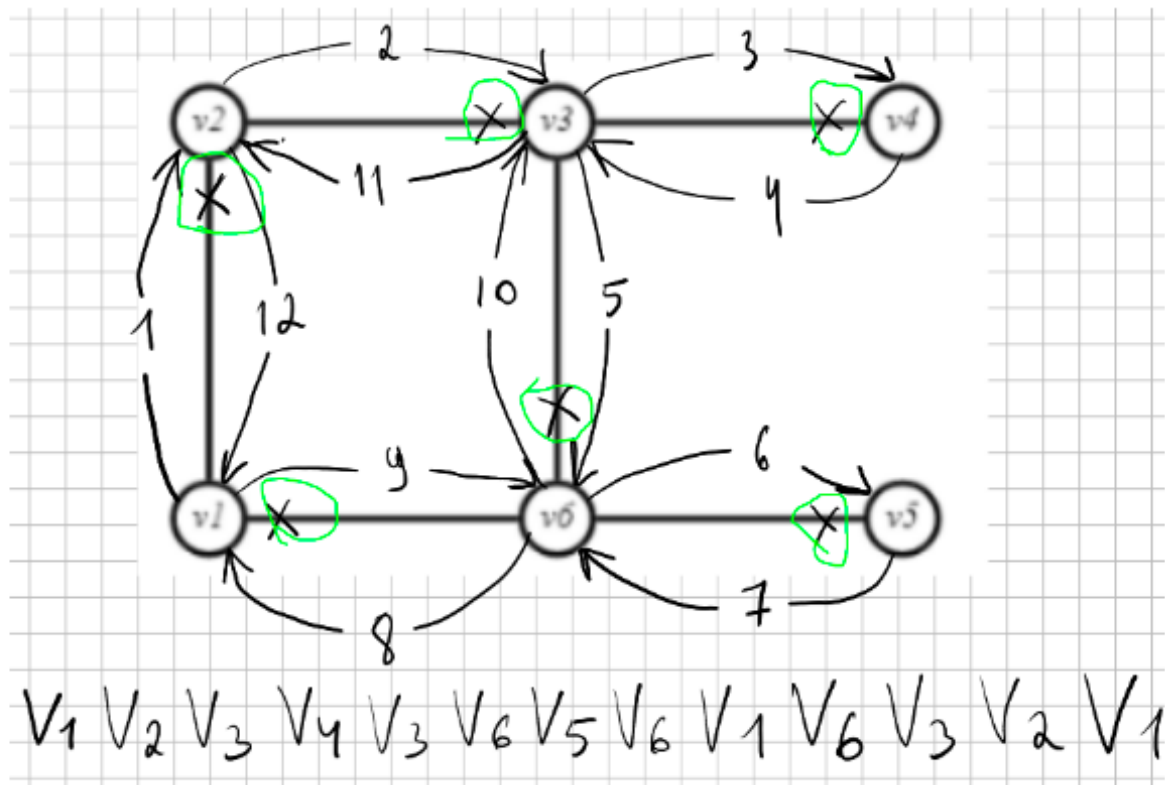
Условие:

Задан неориентированный граф:



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

Решение:



Задача 1

Условие:

Ориентированный граф задан матрицей смежности:

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

Используя алгоритм «фронта волны», найти все минимальные пути из первой вершины графа в его последнюю вершину.

Решение:

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	$ \left. \begin{aligned} k=0: & \text{FW}_0 = \{V_1\} \\ k=1: & \text{FW}_1 = \{V_4, V_7\} \\ k=2: & \text{FW}_2 = \{V_5, V_6\} \\ k=3: & \text{FW}_3 = \{V_3\} \\ k=4: & \text{FW}_4 = \{V_2\} \\ k=5: & \text{FW}_5 = \{V_8\} \end{aligned} \right\} $	
V_1	0	0	0	1	0	0	1	0		
V_2	1	0	1	1	0	1	0	1		
V_3	1	1	0	1	1	1	0	0		
V_4	0	0	0	0	1	1	1	0		
V_5	0	0	1	1	0	1	1	0		
V_6	1	0	1	0	1	0	0	0		
V_7	0	0	0	1	1	1	1	0		
V_8	0	1	1	0	0	0	1	0		

Минимальная длина пути = 5

Мин пути из V_1 в V_8 : 1) $V_1 V_4 V_5 V_3 V_2 V_8$; 2) $V_1 V_4 V_6 V_3 V_2 V_8$
 3) $V_1 V_7 V_6 V_3 V_2 V_8$; 4) $V_1 V_7 V_5 V_3 V_2 V_8$

Задача 2

Условие:

Нагруженный граф задан матрицей длин дуг:

$$\begin{pmatrix} \infty & 2 & 7 & 8 & \infty & \infty & \infty \\ 12 & \infty & 4 & \infty & 6 & \infty & \infty \\ \infty & 4 & \infty & 1 & 3 & 5 & 7 \\ \infty & \infty & 1 & \infty & \infty & 3 & \infty \\ \infty & \infty & 3 & \infty & \infty & \infty & 5 \\ \infty & \infty & 5 & \infty & \infty & \infty & 2 \\ 2 & \infty & \infty & 3 & 4 & 6 & 7 \end{pmatrix}$$

Используя алгоритм Форда, найти все минимальные пути из первой вершины графа во все достижимые вершины.

Решение:

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	$\lambda_i^{(0)}$	$\lambda_i^{(1)}$	$\lambda_i^{(2)}$	$\lambda_i^{(3)}$	$\lambda_i^{(4)}$	$\lambda_i^{(5)}$	$\lambda_i^{(6)}$
V_1	∞	2	7	8	∞	∞	∞	0	0	0	0	0	0	0
V_2	12	∞	4	∞	6	∞	∞	∞	2	2	2	2	2	2
V_3	∞	4	∞	1	3	5	7	∞	7	6	6	6	6	6
V_4	∞	∞	1	∞	∞	3	∞	∞	8	7	7	7	7	7
V_5	∞	∞	3	∞	∞	∞	5	∞	∞	9	8	8	8	8
V_6	∞	∞	5	∞	∞	∞	2	∞	∞	11	11	10	10	10
V_7	2	∞	∞	3	4	6	7	∞	∞	14	13	13	12	12

$V_2: V_1 V_2$
 $V_3: V_1 V_2 V_3$
 $V_4: V_1 V_2 V_3 V_4$
 $V_5: V_1 V_2 V_5$
 $V_6: V_1 V_2 V_3 V_4 V_6$
 $V_7: V_1 V_2 V_3 V_4 V_6 V_7$

Задача 3

Условие:

Ориентированного граф задан матрицей смежности:

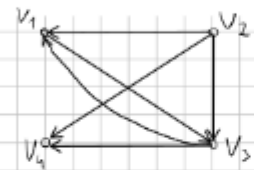
$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Методом Магу найти

- максимальные внутренне устойчивые множества;
- минимальные внешне устойчивые множества;
- ядра.

Решение:

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



а) $(\bar{v}_3 \vee \bar{v}_4) \wedge (\bar{v}_1 \vee \bar{v}_2) \wedge (\bar{v}_3 \vee \bar{v}_2) \wedge (\bar{v}_2 \vee \bar{v}_4) \wedge (\bar{v}_3 \vee \bar{v}_1) \wedge (\bar{v}_3 \vee \bar{v}_4)$
 $\equiv [\bar{v}_3 \vee (\bar{v}_1 \wedge \bar{v}_2 \wedge \bar{v}_4)] \wedge [\bar{v}_2 \vee (\bar{v}_1 \wedge \bar{v}_4)] \equiv (\bar{v}_2 \wedge \bar{v}_3) \vee (\bar{v}_1 \wedge \bar{v}_2 \wedge \bar{v}_4)$
 $\vee (\bar{v}_1 \wedge \bar{v}_3 \wedge \bar{v}_4) \vee (\bar{v}_1 \wedge \bar{v}_2 \wedge \bar{v}_4) \equiv (\bar{v}_2 \wedge \bar{v}_3) \vee (\bar{v}_1 \wedge \bar{v}_2 \wedge \bar{v}_4) \vee (\bar{v}_1 \wedge \bar{v}_3 \wedge \bar{v}_4)$

 \downarrow $\{v_1, v_4\}$ $\{v_3\}$ $\{v_2\}$

б) $(v_1 \vee v_3) \wedge (\cancel{v_1 \vee v_2 \vee v_3 \vee v_4}) \wedge (\cancel{v_1 \vee v_3 \vee v_4}) \wedge v_4 \equiv$
 $\equiv (v_1 \vee v_3) \wedge v_4 \equiv (\downarrow v_1 \wedge v_4) \vee (v_3 \wedge v_4)$

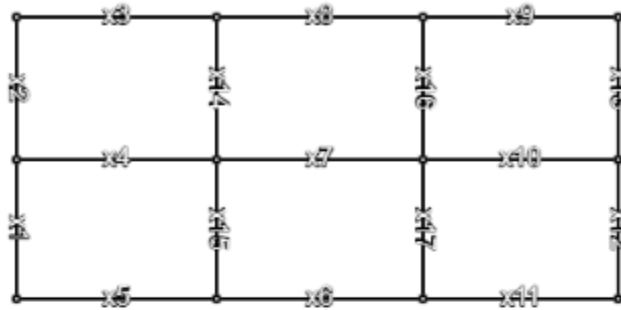
 \downarrow $\{v_1, v_4\}$ $\{v_3, v_4\}$

в) максимальные внутренне устойчивые множества: $\{v_1, v_4\}, \{v_3\}, \{v_2\} \Rightarrow$
 минимальные внешне устойчивые множества: $\{v_1, v_4\}, \{v_3, v_4\}$
 \Rightarrow ядро: $\{v_1, v_4\}$

Задача 4

Условие:

Задан нагруженный граф:



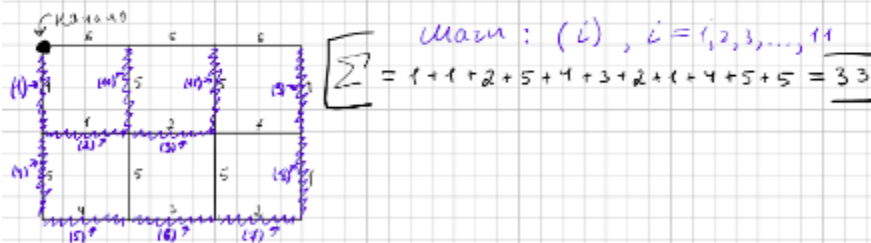
Длины ребер $x_1 - x_{13}$ равны соответственно 5, 1, 6, 1, 4, 3, 2, 5, 6, 7, 2, 1, 4, длины ребер $x_{14} - x_{17}$ равны 5. Найти остовное дерево с минимальной суммой длин входящих в него ребер двумя способами.

- Используя алгоритм Прима.
- Используя алгоритм Крускала.

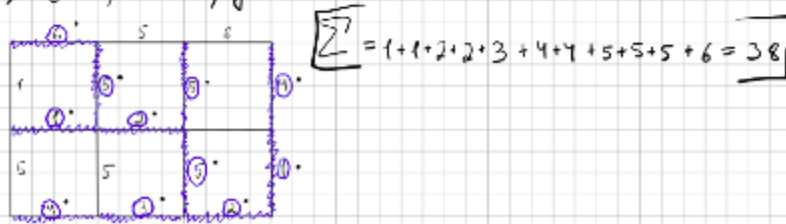
Решение:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}
5	1	6	1	4	3	2	5	6	7	2	1	4	5	5	5	5

а) Алгоритм Прима.



б) Алгоритм Крускала.

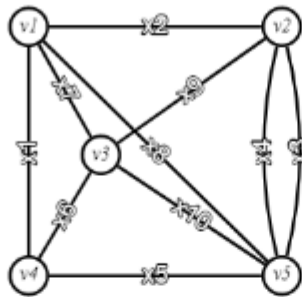


Ответ: размер минимальной суммы длин ребер, входящих в остовное дерево: **33**

Задача 5

Условие:

Электрическая цепь описывается с помощью неориентированного графа.



Первому и пятому ребру графа соответствуют источники тока с ЭДС ε_1 и ε_2 (полярность выбирается произвольно), а остальным ребрам соответствуют сопротивления. Составить линейно независимые системы уравнений Кирхгофа для токов и напряжений. Используя закон Ома, и, предполагая внутренние сопротивления источников тока равными нулю, получить систему уравнений для токов.

Решение:

- 1) Введём произвольным образом ориентацию на ребрах графа.
- 2) Найдём произвольное остовное дерево графа (выделено *синими*).
- 3) Найдём базис циклов графа:

$C_1: x_1 x_6 x_7$ (при увеличении x_1)
 $C_2: x_2 x_3 x_7$ (при увеличении x_2)
 $C_3: x_4 x_5 x_7$ (при увеличении x_4)

$C_4: x_3 x_3 x_{10}$ (при увеличении x_3)
 $C_5: x_5 x_{10} x_6$ (при увеличении x_5)
 $C_6: x_8 x_7 x_{10}$ (при увеличении x_8)
- 4) Составим цикломатическую матрицу:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
C_1	1	0	0	0	0	-1	1	0	0	0
C_2	0	1	0	0	0	0	-1	0	-1	0
C_3	0	0	0	1	0	0	0	0	1	-1
C_4	0	0	1	0	0	0	0	0	1	-1
C_5	0	0	0	0	1	1	0	0	0	1
C_6	0	0	0	0	0	0	1	1	0	1

$u = \varepsilon_1 \ u_2 \ u_3 \ u_4 \ \varepsilon_2 \ u_6 \ u_7 \ u_8 \ u_9 \ u_{10}$
- 5) Составим сист. урав-ий Кирхгофа для напряжений:

$$\begin{cases} \varepsilon_1 - u_6 + u_7 = 0 \\ u_2 - u_7 - u_3 = 0 \\ u_4 + u_3 - u_{10} = 0 \\ u_3 + u_3 - u_{10} = 0 \\ \varepsilon_2 + u_6 + u_{10} = 0 \\ u_7 + u_8 + u_{10} = 0 \end{cases} \quad (1)$$

[6] Составим матрицу инцидентности.

$$\begin{array}{c}
 V_1 \\
 V_2 \\
 V_3 \\
 V_4 \\
 V_5
 \end{array}
 \begin{array}{c}
 x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ x_8 \ x_9 \ x_{10} \\
 \left[\begin{array}{cccccccccc}
 -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\
 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 1 \\
 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 1 & -1 & 1 & 0 & 0 & 1 & 0 & -1
 \end{array} \right]
 \end{array}$$

$$I = (I_1 \ I_2 \ I_3 \ I_4 \ I_5 \ I_6 \ I_7 \ I_8 \ I_9 \ I_{10})^T$$

[7] Составим сист. ур-в

Кирхгофа для токов

$$\begin{cases}
 -I_1 + I_2 + I_4 - I_8 = 0 \\
 -I_2 + I_3 + I_4 - I_5 = 0 \\
 -I_6 - I_7 + I_9 + I_{10} = 0 \\
 I_1 - I_5 + I_6 = 0
 \end{cases} \quad (2)$$

[8] Составим сист. ур-в для токов, заменив сист. (2) сист-й (1), в которой заменим $U_k = i\omega \cdot x_k$ (Закон Ома)

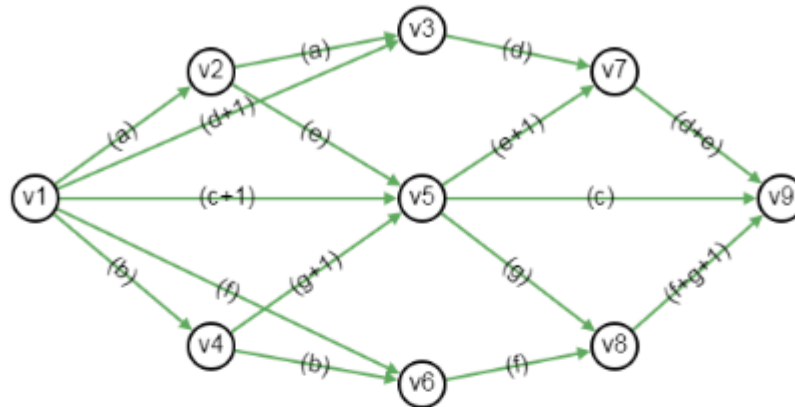
$$\begin{cases}
 E_1 - I_6 R_6 + I_7 R_7 = 0 \\
 I_2 R_2 - I_7 R_7 - I_9 R_9 = 0 \\
 I_4 R_4 + I_9 R_9 - I_{10} R_{10} = 0 \\
 I_3 R_3 + I_5 R_5 - I_{10} R_{10} = 0 \\
 E_2 + I_6 R_6 + I_{10} R_{10} = 0 \\
 I_7 R_7 + I_8 R_8 + I_{10} R_{10} = 0 \\
 -I_1 + I_2 + I_4 - I_8 = 0 \\
 -I_2 + I_3 + I_4 - I_5 = 0 \\
 -I_6 - I_7 + I_9 + I_{10} = 0 \\
 I_1 - I_5 + I_6 = 0
 \end{cases}$$

10 уравнений, 10 неизб-стх \Rightarrow решение.

Задача 6

Условие:

Задана транспортная сеть:

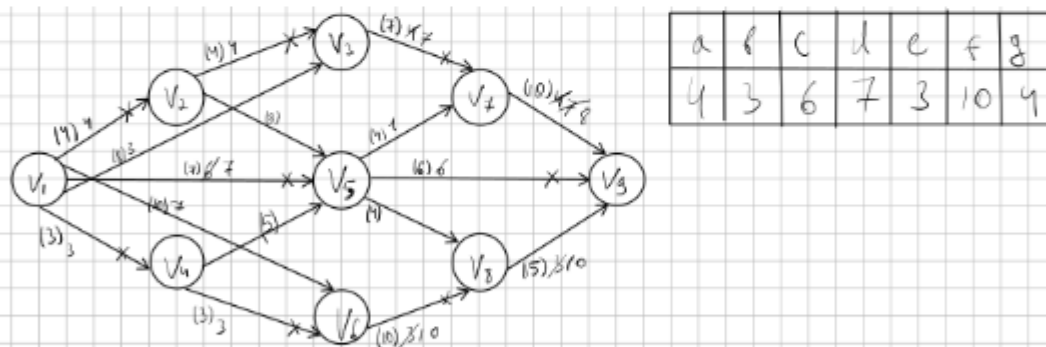


Значения величин a, b, c, d, e, f, g равны соответственно 4, 3, 6, 7, 3, 10, 4.

Построить максимальный поток по транспортной сети.

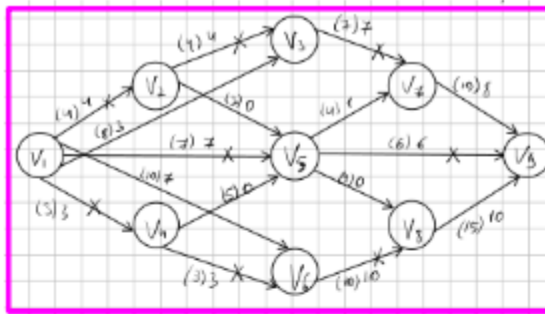
Указание: начинать с окаймляющих цепей.

Решение:



I) Нахождение полного потока:

- 1) $V_1 V_2 V_3 V_7 V_9$: $\Delta_1 = \min(4-0, 4-0, 7-0, 10-0) = 4$
- 2) $V_1 V_4 V_6 V_8 V_9$: $\Delta_2 = \min(3-0, 3-0, 10-0, 15-0) = 3$
- 3) $V_1 V_3 V_7 V_9$: $\Delta_3 = \min(8-0, 7-4, 10-4) = 3$
- 4) $V_1 V_5 V_9$: $\Delta_4 = \min(7-0, 6-0) = 6$
- 5) $V_1 V_5 V_7 V_9$: $\Delta_5 = \min(7-6, 4-0, 10-7) = 1$
- 6) $V_1 V_6 V_8 V_9$: $\Delta_6 = \min(10-0, 10-3, 15-3) = 7$



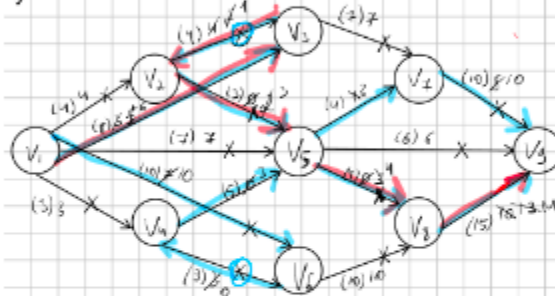
Величина полного потока.

$$\varphi_{\max} = 4 + 3 + 7 + 7 + 3 = 24 = 8 + 6 + 10$$

Ответ к (I)

$$\varphi_{\max} = 24$$

II) Поиск минимального разреза.

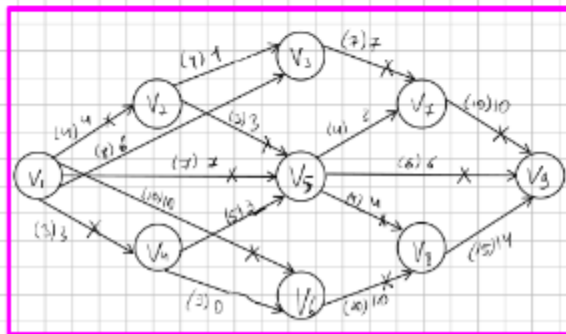


(X) - для потока не вычисляется, потому что "переходим" от источника к стоку.

$$1) V_1 V_3 V_2 V_5 V_7 V_9: \Delta_7 = \min(8-3, 4, 3-0, 7-1, 10-8) = 2$$

$$2) V_1 V_6 V_4 V_5 V_8 V_9: \Delta_8 = \min(10-7, 3, 5-0, 4-0, 15-10) = 3$$

$$3) V_1 V_3 V_2 V_5 V_6 V_9: \Delta_9 = \min(4-5, 2, 3-1, 4-3, 15-13) = 1$$



Величина макс. потока.

$$\varphi_{\max} = 4 + 6 + 7 + 10 + 3 = 30 = 10 + 6 + 14$$

$$\varphi_{\max} = 30$$

Ответ к (II)

Часть II. Теория алгоритмов и алгебраические структуры.

Задача 7

Условие:

Получить заданную функцию $f(x, y) = 3x(y + 2)$ с помощью оператора примитивной рекурсии, используя оператор суперпозиции, а также функции:

$$S(x) = x + 1, O(x) = 0, I_n^m(x_1, \dots, x_n) = x_m \ (1 \leq m \leq n), \Sigma(x_1, x_2) = x_1 + x_2.$$

Решение:

$$\begin{aligned} f(x, y) &= 3x(y+2) \\ \begin{cases} f(x, 0) = 3 \cdot x \cdot 2 = 6x = \varphi(x) \\ f(x, y+1) = 3x(y+2) + 1 = \overbrace{3x(y+2)}^{f(x, y)} + 3x = \overbrace{f(x, y)}^z + 3x \end{cases} \\ \cdot \varphi(x) &= 6x \\ \cdot \Psi(x, y, z) &= z + 3x = \Sigma(z, \Sigma(\Sigma(x, x), x)) \end{aligned}$$

Задача 8

Условие:

Для заданной подстановки $[(85214)(6231)(8145)(4726)]^{-85}$ из S_8 определить:

- разложение на независимые циклы;
- порядок;
- представить в виде произведения транспозиций;
- четность подстановки.

Решение:

Разложим на непересекающиеся циклы

$$\begin{aligned}
 \text{а) } & [(85214)(6231)(8145)(4726)]^{-85} = \\
 & = [(1862)(347)(5)]^{-85} = (1862)^{41 \cdot 21 - 1} \cdot (347)^{31 \cdot 28 - 1} \cdot (5)^{11 \cdot 25} = \\
 & = (1862)^{-1} \cdot (347)^{-1} = (2681) \cdot (743) = (1268)(374) \quad (a) \\
 \text{б) } & \text{порядок} = \text{НОК}(4, 3) = 12 \quad (5) \\
 \text{в) } & \text{представим в виде произведения транспозиций:} \\
 & (1268)(347) = (18)(16)(12)(37)(34) \quad (6) \\
 \text{г) } & \text{четность: } n = 5 \text{ — нечетная} \quad (2)
 \end{aligned}$$

Задача 9

Условие:

Определить для подгруппы $H = \langle (1243), (23) \rangle$ группы S_4 :

- а) элементы из H ;
- б) левые смежные классы группы S_4 по H ;
- в) правые смежные классы группы S_4 по H ;
- г) является ли H нормальной подгруппой?

Решение:

Элементы группы S_4

$$S_4 = \{e, (12), (13), (14), (23), (24), (34), (123), (124), (134), (234), (132), (142), (143), (243), (12)(34), (13)(24), (14)(23), (1234), (1243), (1324), (1342), (1423), (1432)\}$$

а) $H = \langle (1234), (23) \rangle$

Входят в H элементы:

- (I) единицы \rightarrow и-т из S_4 — e
- (II) обратные к $(1234), (23)$:
 $(1234)^{-1} = (4321) = (1432)$; $(23)^{-1} = (23)$
- (III) все композиции всех элементов:

$h_2 \backslash h_1$	e	(1234)	(23)
e	e	(1234)	(23)
(1234)	(1234)	$(13)(24)$	(124)
(23)	(23)	(134)	(23)

Новые \rightarrow и-т: $(13)(24), (124), (134)$

Зеленым цветом буду писать новые элементы H

$(1234)(1234) = (13)(24)$
 $(1234)(23) = (124)$
 $(23)(1234) = (134)$
 $(23)(23) = (23)$

$h_i \backslash h_j$	1	2	3	4	5	6
h_i	e	(1234)	(23)	$(13)(24)$	(124)	(134)
1	e	(1234)	(23)	$(13)(24)$	(124)	(134)
2	(1234)	(1234)	$(13)(24)$	(124)	(1432)	(1423)
3	(23)	(23)	(134)	(23)	(1234)	(1324)
4	$(13)(24)$	$(13)(24)$	(1432)	(1342)	e	$(143)(24)$
5	(124)	(124)	e	e	e	e
6	(134)	(134)	e	e	e	e

2 группа;

$$(1234)(13)(24) = (1432)$$

$$(1234)(124) = (1342)$$

$$(1234)(134) = (1423)$$

3 группа;

$$(23)(13)(24) = (1243)$$

$$(23)(124) = (1324)$$

$$(23)(134) = (1234)$$

$$4 \text{ группа: } (13)(24)(1234) = (1432); (13)(24)(23) = (1342);$$

$$(13)(24)(13)(24) = (1)(2)(3)(4) = e; (13)(24)(124) = (143); (13)(24)(134) = (1)(24)(3)$$

Можно получить 13 элементов в группе $H \Rightarrow$ по т. Лагранжа в H содержится 24 элемента, иначе было бы $H = S_4$

8) Т.е. если классы пересечения, то они совпадают \Rightarrow

$$\Rightarrow \text{ЛСК: } eH = \{e, (12), (13), (14), (23), (24), (34), (123), (124), (134), (234), (132), (142), (143), (243), (12)(34), (13)(24), (14)(23), (1234), (1243), (1324), (1342), (1423), (1432)\} = (12)H = (13)H = \dots = (34)H = (123)H = \dots = (243)H = (12)(34)H = \dots = (14)(23)H = (134)H = \dots = (1432)H \text{ — здесь } SH, \text{ где } S \in S_4$$

$$6) \text{ ПСК: } H = \{e, (12), (13), (14), (23), (24), (34), (123), (124), (134), (234), (132), (142), (143), (243), (12)(34), (13)(24), (14)(23), (1234), (1243), (1324), (1342), (1423), (1432)\} = H(12) = H(13) = \dots = H(34) = H(123) = \dots = H(243) = H(12)(34) = \dots = H(14)(23) = H(134) = \dots = H(1432) \text{ — здесь } HS, \text{ где } S \in S_4$$

2) ЛСК и ПСК совпадают \Rightarrow подгруппа H является нормальным делителем группы S_4 .

Задача 10

Условие:

Рассматривается (4,7)-код Хемминга. Для слова $a = 1001$ определить соответствующее ему кодовое слово. Пусть при приеме слов $c' = 1011101$, $c'' = 1011110$ была допущена ошибка не более, чем в одной позиции. Определить наличие и положение ошибки. Какие слова были переданы? Какие слова были закодированы?

Решение:

Решение: $a = (1001)$; $c' = (1011101)$; $c'' = (1011110)$

$$b = b_1 b_2 1 b_4 001$$

$$b \cdot M = (b_1 b_2 1 b_4 001) \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \Leftrightarrow \begin{cases} b_4 \oplus 1 = 0; b_4 = 1 \\ b_2 = 0 \\ b_1 = 0 \end{cases}$$

$$b = 0011001$$

$$c' M = (1011101) \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = (100)_2 = 4_{10} \Rightarrow$$

\rightarrow ошибка в 4:

$$b = (1010101); a = (1101)$$

$$c'' M = (1011110) \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = (101)_2 = 5_{10} \Rightarrow$$

\rightarrow ошибка в 5:

$$b = (1011010); a = (11010)$$

Часть III. Программная реализация алгоритмов.

Описание выбранного алгоритма.

Алгоритм Дейкстры (англ. Dijkstra's algorithm) — алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании и технологиях.

Формальное определение задачи алгоритма.

Дан взвешенный ориентированный граф $G(V, E)$ без дуг отрицательного веса [1]. Найти кратчайшие пути от некоторой вершины a графа G до всех остальных вершин этого графа.

Взвешенный граф — граф, каждому ребру которого поставлено в соответствие некое значение (вес ребра).

Оrientированный граф (кратко орграф) — (мульти) граф, рёбрам которого присвоено направление. Направленные рёбра именуются также дугами, а в некоторых источниках и просто рёбрами. Граф, ни одному ребру которого не присвоено направление, называется неориентированным графом или неорграфом.

Неформальное объяснение алгоритма.

- Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до a .
- Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки.
- Работа алгоритма завершается, когда все вершины посещены.

[1] Для графа с отрицательными весами применяется более общий алгоритм — Алгоритм Дейкстры с потенциалами. Но здесь перед мной стоит задача изучить только стандартный алгоритм дейкстры.

Инициализация.

Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности.

Это отражает то, что расстояния от a до других вершин пока неизвестны.

Все вершины графа помечаются как непосещённые.

Шаг алгоритма.

Если все вершины посещены, алгоритм завершается.

В противном случае, из ещё не посещённых вершин выбирается вершина u , имеющая минимальную метку.

Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u , назовём соседями этой вершины. Для каждого соседа вершины u , кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки u и длины ребра, соединяющего u с этим соседом.

Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину u как посещённую и повторим шаг алгоритма.

Описание

В простейшей реализации для хранения чисел $d[i]$ можно использовать массив чисел, а для хранения принадлежности элемента множеству U — массив булевых переменных.

В начале алгоритма расстояние для начальной вершины полагается равным нулю, а все остальные расстояния заполняются большим положительным числом (большим максимального возможного пути в графе). Массив флагов заполняется нулями. Затем запускается основной цикл.

На каждом шаге цикла мы ищем вершину v с минимальным расстоянием и флагом равным нулю. Затем мы устанавливаем в ней флаг в 1 и проверяем все соседние с ней вершины u . Если в них (в u) расстояние больше, чем сумма расстояния до текущей вершины и длины ребра, то уменьшаем его. Цикл завершается, когда флаги всех вершин становятся равны 1, либо когда у всех вершин с флагом 0 $d[i]=\infty$. Последний случай возможен тогда и только тогда, когда граф G несвязный.

Доказательство корректности алгоритма

Пусть $l(v)$ — длина кратчайшего пути из вершины a в вершину v . Докажем по индукции, что в момент посещения любой вершины z $d(z)=l(z)$.

База. Первой посещается вершина a . В этот момент $d(a)=l(a)=0$.

Шаг. Пусть мы выбрали для посещения вершину $z \neq a$. Докажем, что в этот момент $d(z)=l(z)$. Для начала отметим, что для любой вершины v всегда выполняется $d(v) \geq l(v)$ (алгоритм не может найти путь короче, чем кратчайший из всех существующих). Пусть P — кратчайший путь из a в z . Вершина z находится на P и не посещена. Поэтому множество непосещённых вершин на P непусто. Пусть y — первая непосещённая вершина на P , x — предшествующая ей (следовательно, посещённая). Поскольку путь P кратчайший, его часть, ведущая из a через x в y , тоже кратчайшая, следовательно $l(y)=l(x)+w(xy)$. По предположению индукции, в момент посещения вершины x выполнялось $d(x)=l(x)$, следовательно, вершина y тогда получила метку не больше чем $d(x)+w(xy)=l(x)+w(xy)=l(y)$. Следовательно, $d(y)=l(y)$. Если $z=y$, то индукционный переход доказан. Иначе, поскольку сейчас выбрана вершина z , а не y , метка z минимальна среди непосещённых, то есть $d(z) \leq d(y)=l(y) \leq l(z)$. Комбинируя это с $d(z) \geq l(z)$, имеем $d(z)=l(z)$, что и требовалось доказать.

Поскольку алгоритм заканчивает работу, когда все вершины посещены, в этот момент $d=l$ для всех вершин.

Описание реализации алгоритма.

Так как программа будет работать с системой ГРАФОИД, необходимо в программе алгоритма реализовать получение данных и возврат данных с системой ГРАФОИД. Программа будет получать текстовый файл с матрицей смежности, преобразовывать его и возвращать обратно в систему для последующей визуализации и вывода графа кратчайших путей.

Представим граф в виде матрицы смежности $matrix[razmer+1][razmer+1]$. В массиве $distance[razmer+1]$ хранятся минимальные расстояния от вершины источника, до остальных. $visited[razmer+1]$ — тут хранятся посещенные вершины, добавленные в особый путь. В первом цикле алгоритма выполняется поиск путей и их длина от источника до остальных. Во втором цикле релаксируем длину, так как может найтись меньший путь.

Оценка вычислительной сложности алгоритма.

Сложность алгоритма Дейкстры зависит от способа нахождения вершины v , а также способа хранения множества непосещённых вершин и способа обновления меток. Обозначим через n количество вершин, а через m — количество рёбер в графе G .

В простейшем случае, когда для поиска вершины с минимальным $d[v]$ просматривается всё множество вершин, а для хранения величин d используется массив, время работы алгоритма есть $O(n^2)$. Основной цикл выполняется порядка n раз, в каждом из них на нахождение минимума тратится порядка n операций. На циклы по соседям каждой посещаемой вершины тратится количество операций, пропорциональное количеству рёбер m (поскольку каждое ребро встречается в этих циклах ровно дважды и требует константное число операций). Таким образом, общее время работы алгоритма $O(n^2 + m)$, но, так как $m \leq n(n-1)$, оно составляет $O(n^2)$.

Для разреженных графов (то есть таких, для которых m много меньше n^2) непосещённые вершины можно хранить в двоичной куче, а в качестве ключа использовать значения $d[i]$, тогда время удаления вершины из U станет $\log(n)$ при том, что время модификации $d[i]$ возрастет до $\log n$. Так как цикл выполняется порядка n раз, а количество релаксаций (смен меток) не больше m , время работы такой реализации — $O(n \cdot \log n + m \cdot \log n)$.

Если для хранения непосещённых вершин использовать фибоначчиеву кучу, для которой удаление происходит в среднем за $O(\log n)$, а уменьшение значения в среднем за $O(1)$, то время работы алгоритма составит $O(n \cdot \log n + m)$.

Описание программы и инструкции по работе с ней.

Программа работает с системой ГРАФОИД. Для начала необходимо открыть ГРАФОИД, построить или открыть неориентированный ненагруженный граф.

Чтобы открыть граф:

- Нажать на кнопку “Граф”
- Выбрать “Открыть”
- Нажать на “обычный граф”
- Выбрать текстовый файл из имеющихся на устройстве

Чтобы создать граф:

- Нажать на кнопку “Граф”

- Выбрать “Создать”
- Нажать на “Новый обычный граф”
- В ячейке “Настройки” выбрать “Вершины”
- Расставить вершины первой доли в поле
- В ячейке “Настройки” выбрать “Вершины 2”
- Расставить вершины второй доли в поле
- В ячейке “Настройки” выбрать “Рёбра”
- Расставить их

Для запуска программы:

- В графе “Приложения” “Выбрать исполняемый файл” в формате .exe
- В графе “Приложения” “Запустить исполняемый файл”

После этого граф перестроится, появится граф, состоящий из кратчайших путей между вершинами.

Тестовые примеры с решением и скриншотами.

См. приложения 2-5.

Пример прикладной задачи.

Вариант 1. Дана сеть автомобильных дорог, соединяющих города Московской области. Некоторые дороги односторонние. Найти кратчайшие пути от города Москвы до каждого города области (если двигаться можно только по дорогам).

Вариант 2. Имеется некоторое количество авиарейсов между городами мира, для каждого известна стоимость. Стоимость перелёта из А в В может быть не равна стоимости перелёта из В в А. Найти маршрут минимальной стоимости (возможно, с пересадками) от Копенгагена до Барнаула.

Заключение.

В ходе выполнения курсовой работы были решены 12 задач по различным разделам курса “Дискретная математика”.

Кроме того был изучен вопрос о различных методах поиска путей и маршрутов в графах. Была написана и отлажена программа, реализующая алгоритм Дейкстры. Программа написана на языке программирования C++. Программа обеспечивает связь по установленному формату с системой ГРАФОИД, разработанной на кафедре 805, что дает возможность обеспечить графический интерфейс при ее использовании. Эта программа является основным результатом курсового проектирования.

Список использованных источников

1. В.А. Таланов, В.Е. Алексеев. Графы и алгоритмы. <https://www.intuit.ru> .
2. Нефедов В.Н., Осипова В.А. Курс дискретной математики. МАИ, 1992
3. Нефедов В.Н. Дискретные задачи оптимизации. <https://goo.gl/faUEEU>
4. Алгоритм Дейкстры. Википедия https://ru.wikipedia.org/wiki/Алгоритм_Дейкстры

Приложение 1

Текст программы

```
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <climits>
#include <string.h>
#include <ctype.h>

#define STR 10

using namespace std;

char *itoa(int value, char* result, int base) // функция перевода int в char[]
{
    // check that the base is valid
    if (base < 2 || base > 36) { *result = '\0'; return result; }

    char *ptr = result, *ptr1 = result, tmp_char;
    int tmp_value;

    do {
        tmp_value = value;
        value /= base;
        *ptr++ = "zyxwvutsrqponmlkjihgfedcba9876543210123456789abcdefghijklmnopqrstuvwxyz"
[35 + (tmp_value - value * base)];
    } while ( value );

    // Apply negative sign
    if (tmp_value < 0) *ptr++ = '-';
    *ptr-- = '\0';
    while(ptr1 < ptr) {
        tmp_char = *ptr;
        *ptr-- = *ptr1;
        *ptr1++ = tmp_char;
    }
    return result;
}
```

```

int main(int argc, char *argv[])
{
    //Loading matrix
    ifstream in(argv[1]);
    int razmer=0,i,j;
    in >> razmer;
    razmer--;
    int matrix[razmer+1][razmer+1];
    for(int i=0;i<=razmer;i++) {
        for(int j=0;j<=razmer;j++) {
            in >> matrix[i][j];
        }
    }
    in.close();

    ////////// Algorithm
    for (int l = 1; l <= razmer + 1; l++) {
        int distance[razmer + 1]; // массив минимальных расстояний
        int count, index, i, u, st = l - 1, m = st + 1; // st источник
        bool visited[razmer + 1]; // посещенные вершины, добавленные в особый путь
        int matr[razmer + 1];
        for (i = 0; i < razmer + 1; i++) {
            distance[i] = INT_MAX; // сохраняем бесконечность в расстояния
            visited[i] = false; // делаем вершины непосещенными
        }
        distance[st] = 0; // путь из v1 -> v1 = 0

        for (count = 0; count < razmer; count++) {
            int min = INT_MAX;
            for (i = 0; i < razmer + 1; i++) { // ищем минимальные пути в каждую вершину
                if (!visited[i] && distance[i] <= min) {
                    min = distance[i];
                    index = i;
                }
            }
            u = index; // индекс вершины, добавленной в особый путь
            visited[u] = true;

            for (i = 0; i < razmer + 1; i++) { // ищем меньшую длину пути, если есть (через другие

```

вершины)

```

        if (!visited[i] && matrix[u][i] && (distance[u] != INT_MAX) && (distance[u] + matrix[u][i] <
distance[i])) {
            distance[i] = distance[u] + matrix[u][i];
        }
    }
}
for (int k = 0; k < razmer + 1; k++) {
    if (distance[k] != INT_MAX) {
        matr[k] = distance[k]; // записываем в массив кратчайшие пути
    } else {
        matr[k] = 0; // если пути нет
    }
}
for (int k = 0; k < razmer + 1; k++) {
    matrix[l - 1][k] = matr[k]; // конкатенируем матрицы, получаем матрицу смежности
}
}

```

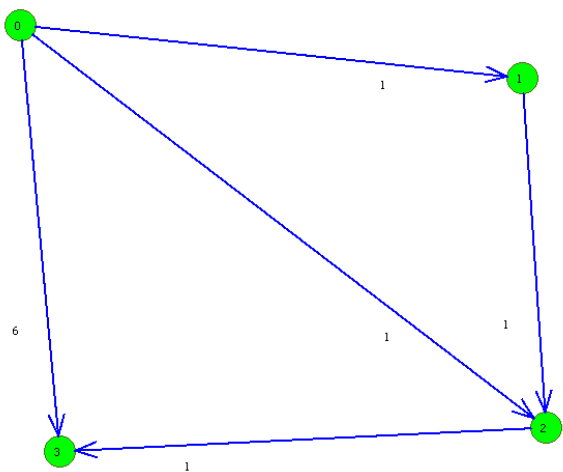
```

//Saving new matrix
fstream out;
out.open(argv[1]);
out.clear();
char buffer [33];
itoa(razmer+1,buffer,10);
out << buffer << "\n";
for(i=0; i<=razmer;i++) {
    for(j=0;j<=razmer;j++) {
        out << matrix[i][j];
        if(j!=razmer) {
            out << " ";
        }
    }
    if(i!=razmer) {
        out << "\n";
    }
}
out.close();
return 0;
}

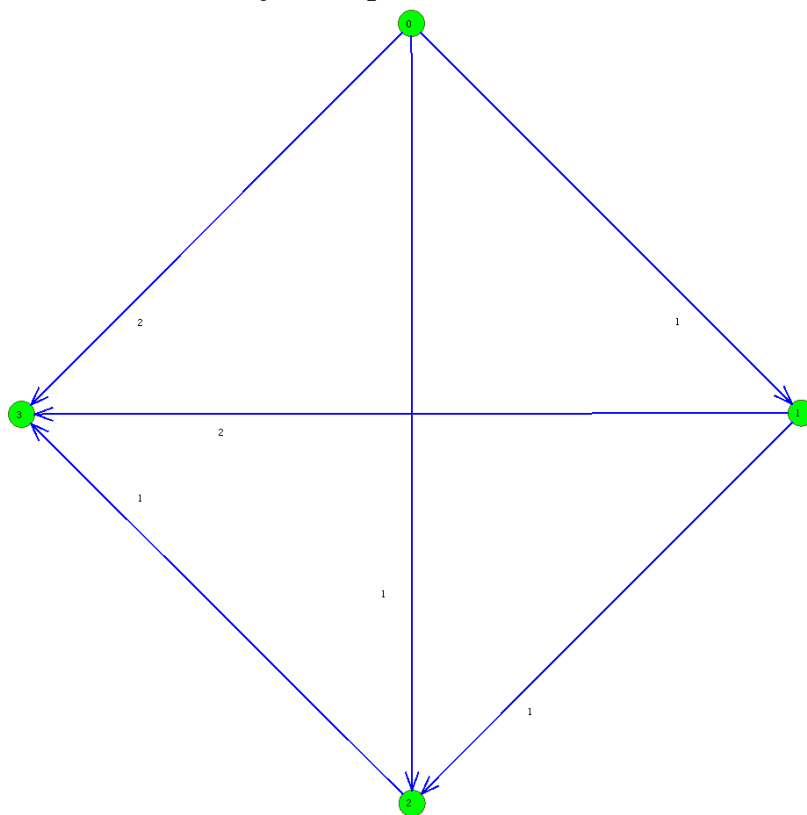
```

Приложение 2

Граф:



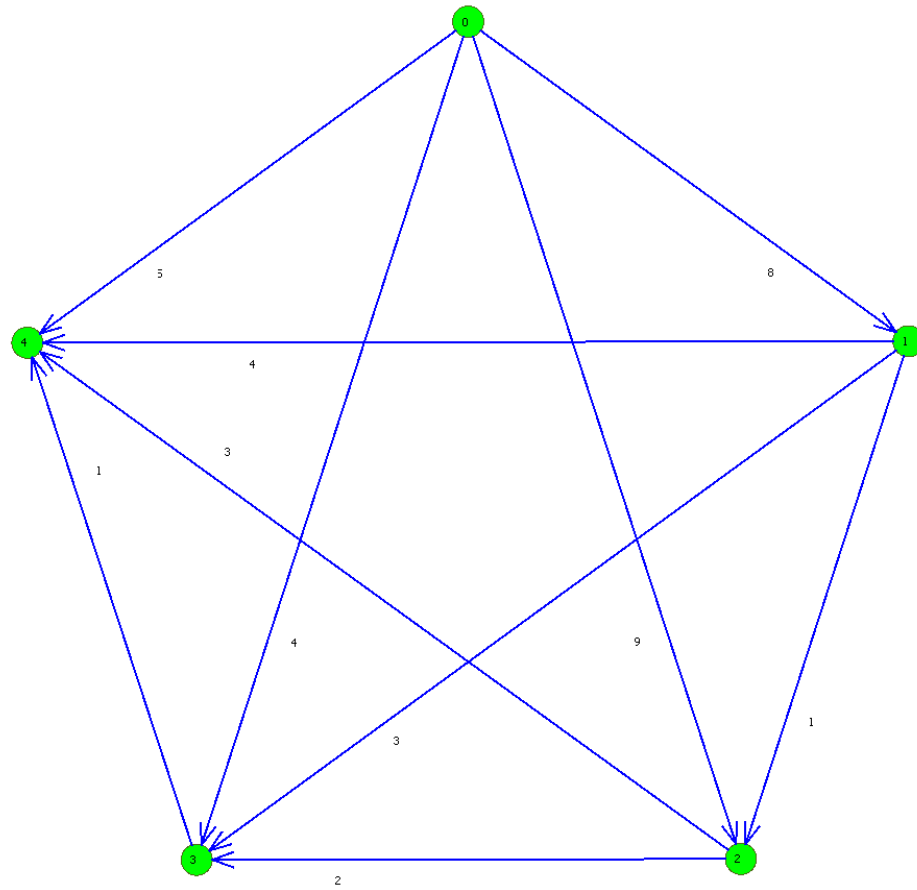
Результат работы:



Приложение 3

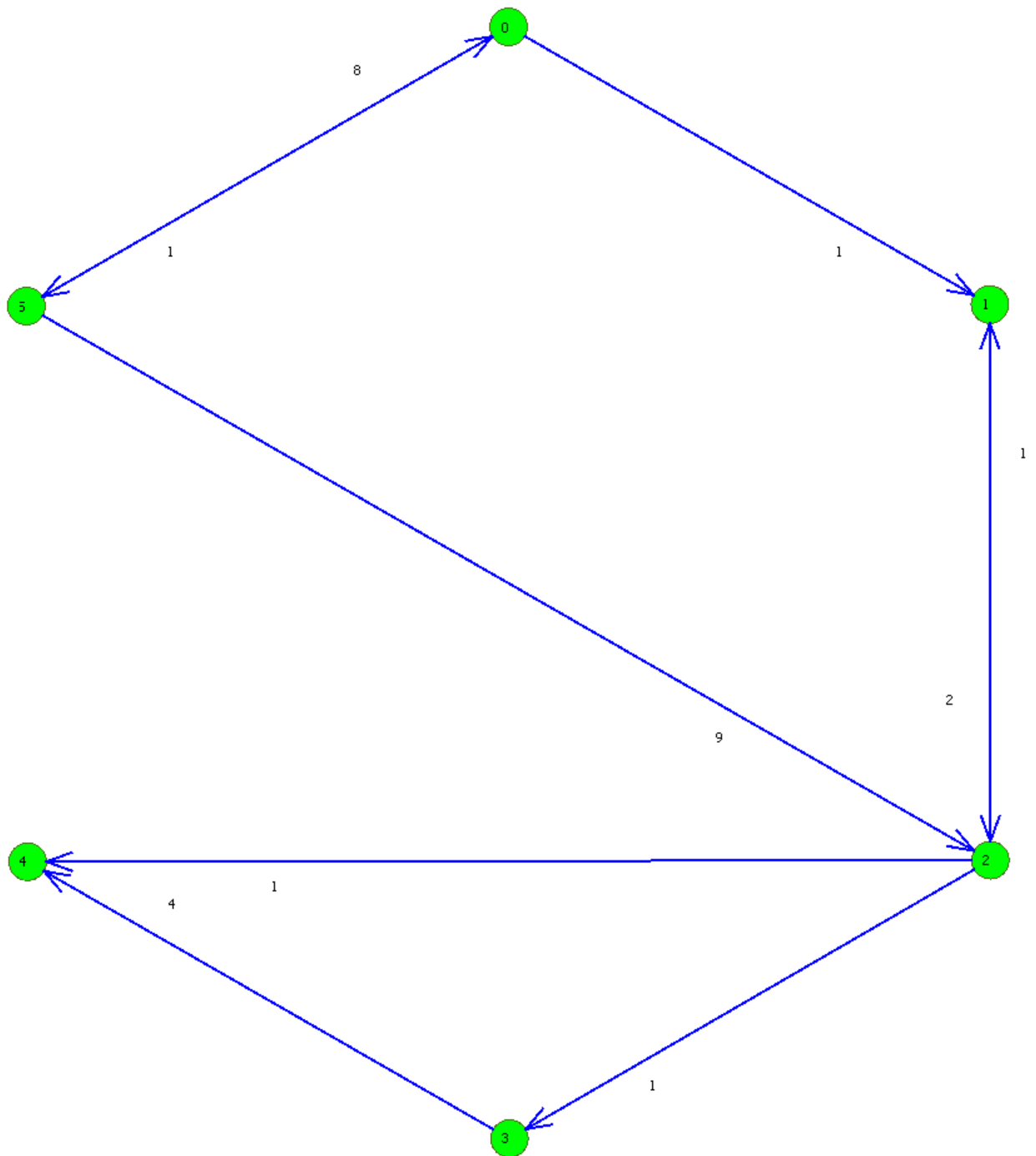
Граф:

Результат работы:



Приложение 4

Граф:



Результат работы:

