**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет информационных технологий и прикладной математики

Кафедра математической кибернетики

**КУРСОВая работа**

**Выполнил:**

Студент учебной группы М8О-102Б-21

Яценко Александр Владимирович

Номер по списку: 20

Преподаватель:

доцент кафедры 805

Смерчинская С.О.

**Оценка:** \_\_\_\_

**Дата:** \_\_\_\_\_\_\_\_\_\_\_

Москва

2022

**Задания 20 варианта**

1. Определить для орграфа, заданного матрицей смежности:

A =

а) матрицу односторонней связности (2 способа, включая итерационный алгоритм);

б) матрицу сильной связности;

в) компоненты сильной связности;

г) матрицу контуров;

д) изображение графа и компонент сильной связности.

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



**3**

**2**

**4**

**1**

**5**

1. Используя алгоритм “фронта волны”, найти все минимальные пути из первой вершины в последнюю орграфа, заданного матрицей смежности.

A =

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

C =

1. Найти остовное дерево с минимальной суммой длин входящих в него ребер.
2. 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 𝐸1 и 𝐸2, а остальные элементы являются сопротивлениями. Используя закон Ома, и, предполагая внутренние сопротивления источников тока равными нулю, получить систему уравнений для токов.



**2**

**8**

**9**

**4**

**11**

**1 6**

**7**

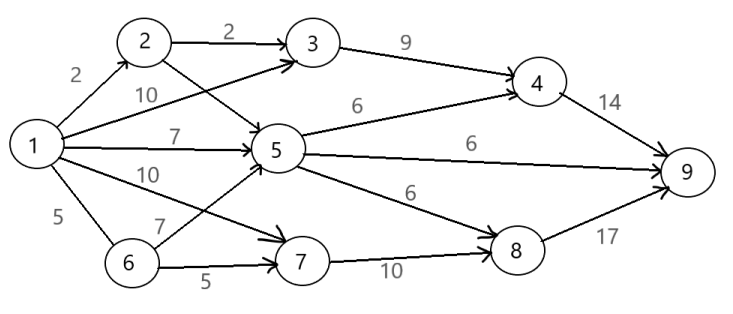
**3**

**10**

**12**

**5**

1. Построить максимальный поток по транспортной сети.



1. Построение таблицы Кэли группы, заданной образующими и определяющими соотношениям.

## *Выполнение курсовой работы.*

**Задание №1**. Связность в орграфе

Определить для графа, заданного матрицей смежности:

A =

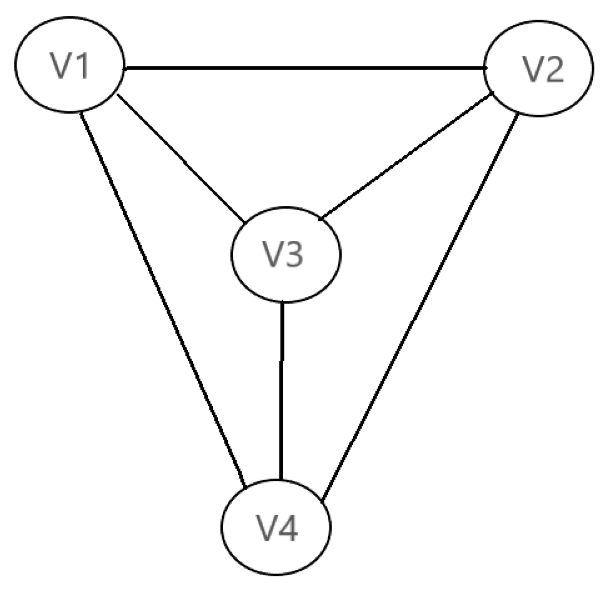
а) матрицу односторонней связности

б) матрицу сильной связности

в) компоненты сильной связности

г) матрицу контуров

д) изображение графа и компонент сильной связности.



**Решение:**

**а) Найдем матрицу односторонней связности по формуле: T = E\/А\/А2\/А3**

1) A2 = A\*A = \* =

2) A3 = A2\*A = \* =

3) T = E\/A\/A2\/A3 = \/ \/ \/ = = T - матрица односторонней связности.

Найдем матрицу односторонней связности по итерационному алгоритму Уоршалла:

k = 0

T(0) = E \/ A = \/ =

k = 1, k-1=0

T(1) = \/ =

k = 2, k-1=1

Следовательно, все последующие матрицы будут равны.

T(1) = T(2) = T(3) = T(4) = = T

**б) Матрица сильной связности = T & TT**

= T & TT = & =

= - матрица сильной связности.

**в) Компоненты сильной связности**

Выбираем первую строку, как ненулевую в матрице сильной связности

=

Номера вершин первой компоненты сильной связности соответствуют номерам столбцов

матрицы 𝑆, в которых в первой строке стоят единицы:

1) 1-ая компонента сильной связности {V1}. Обнуляем первый столбец матрицы

Получаем матрицу:

1 =

2) Ищем ненулевую строку матрицы 1: это вторая строка. Единица одна – во втором

столбце. Следовательно, 2-ая компонента сильной связности {V2}. Обнуляем второй столбец матрицы 1.

Получаем матрицу:

2 =

3) Ищем ненулевую строку матрицы 2.: это третья строка. Единица одна – в третьем

столбце. Следовательно, 3-ая компонента сильной связности {V3}. Обнуляем третий столбец матрицы 2.

Получаем матрицу:

3 =

4) Обнуляем четвёртый столбец матрицы3, получаем нулевую матрицу. Следовательно,

других компонент сильной связности нет.

**г) Матрица контуров k = & A**

k = & =

Следовательно, нет дуг, которые принадлежат какому-либо контуру исходного графа.

# **Задание №2**. Алгоритм Терри

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



**3**

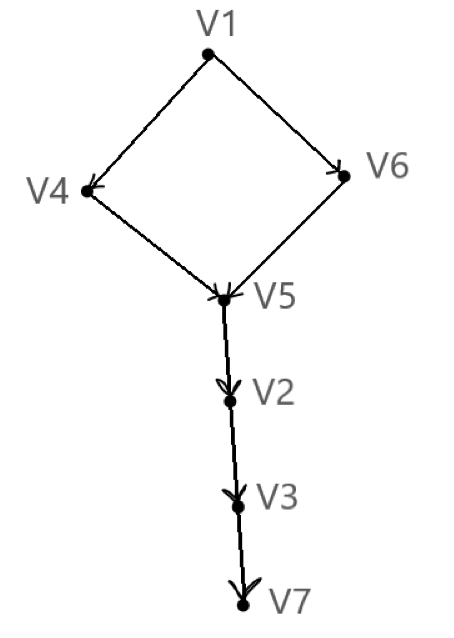
**2**

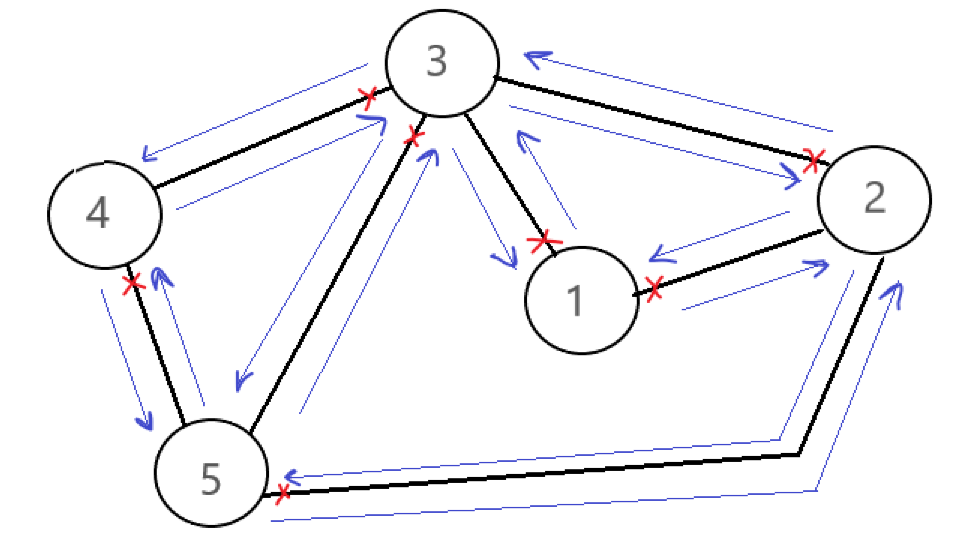
**4**

**1**

**5**

**Решение:**





Маршрут обхода:

1 🡪 2 🡪 3 🡪 4 🡪 5 🡪 2 🡪 1 🡪 3 🡪 2 🡪 5 🡪 4 🡪 3 🡪 5 🡪 3 🡪 1

# **Задание №3**. Алгоритм «фронта волны»

Используя алгоритм «фронта волны», найти все кратчайшие пути из первой вершины в остальные вершины орграфа, заданного матрицей смежности.

A =

Таблица путей:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | V1 | V2 | V3 | V4 | V5 | V6 | V7 |
| V1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| V2 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| V3 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| V4 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| V5 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| V6 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| V7 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

**Решение:**

I этап. Создание волн

1) вершина V1 c индексом 0; V1 ∈ W0(V1)

2) Г V1 = Г W0(V1) = {V4, V6}

вершины V4, V6 с индексом 1; V4, V6 ∈ W1(V1)

3) Г W1(V1) = Г {V4, V6} = {V5}

вершина V5 с индексом 2; V5, V7 ∈ W2(V1)

4) Г W2(V1) = Г {V5} = {V2}

вершина V2 с индексом 3; V2 ∈ W3(V1)

5) Г W3(V1) = Г {V2} = {V3}

вершина V3 с индексом 4; V3 ∈ W4(V1)

6) Г W4(V1) = Г {V3} = {V7}

вершина V7 с индексом 5; V7 ∈ W5(V1)

Вершина V7 достигнута 🡪 длина кратчайшего пути из V1 в V7 равна 5

II этап. Восстановление пути

1) V7

2) W4(V1)**∩**Г-1(V7) = {V3}**∩**{V3} = {V3}

3) W3(V1)**∩**Г-1(V3) = {V2}**∩**{ V2, V7} = {V2}

4) W2(V1)**∩**Г-1(V2) = {V5}**∩**{ V3, V5} = {V5}

5) W1(V1)**∩**Г-1(V5) = { V4, V6}**∩**{ V2, V3 , V4, V6, V7} = {V5}

6.1) W0(V1)**∩**Г-1(V4) = { V4, V6}**∩**{ V1, V3 , V5, V6} = {V6}

6.2) W0(V1)**∩**Г-1(V6) = { V4, V6}**∩**{ V1, V3 , V4, V5, V7} = {V4}

2 кратчайших пути длины 5:

1. V1 🡪 V4 🡪 V5 🡪 V2 🡪 V3 🡪 V7
2. V1 🡪 V6 🡪 V5 🡪 V2 🡪 V3 🡪 V7

# **Задание №4**. Алгоритм Форда

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

C =

**Решение:**

1. Составим таблицу итераций

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | λi(0) | λi(1) | λi(2) | λi(3) | λi(4) | λi(5) | λi(6) | λi(7) |
| V1 | ∞ | 3 | ∞ | ∞ | 6 | ∞ | ∞ | ∞ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| V2 | 13 | ∞ | 3 | 9 | 2 | ∞ | ∞ | ∞ | ∞ | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| V3 | ∞ | ∞ | ∞ | 5 | ∞ | 2 | ∞ | ∞ | ∞ | ∞ | 6 | 6 | 6 | 6 | 6 | 6 |
| V4 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 1 | 3 | ∞ | ∞ | 12 | 11 | 10 | 10 | 10 | 10 |
| V5 | ∞ | 2 | ∞ | ∞ | ∞ | 2 | 4 | ∞ | ∞ | 6 | 5 | 5 | 5 | 5 | 5 | 5 |
| V6 | ∞ | ∞ | 2 | ∞ | ∞ | ∞ | 5 | ∞ | ∞ | ∞ | 8 | 7 | 7 | 7 | 7 | 7 |
| V7 | ∞ | ∞ | ∞ | 1 | ∞ | ∞ | ∞ | 5 | ∞ | ∞ | 10 | 9 | 9 | 9 | 9 | 9 |
| V8 | 2 | ∞ | 5 | 7 | 4 | ∞ | 8 | ∞ | ∞ | ∞ | ∞ | 15 | 14 | 13 | 13 | 13 |

1. Длины минимальных путей из вершины 𝑣1 во все остальные вершины определены в последнем столбце таблицы.
2. Найдем вершины, входящие в минимальные пути из 𝑣1 во все остальные вершины графа.
   1. Минимальный путь из v1 в v2: v1 – v2, его длина равна 3.

λ1(0) + С12 = 0 + 3 = 3 = λ2(1)

* 1. Минимальный путь из v1 в v3: v1 – v2 − v3, его длина равна 6.

λ2(1) + С23 = 3 + 3 = 6 = λ3(2)

λ1(0) + С12 = 0 + 3 = 3 = λ2(1)

* 1. Минимальный путь из v1 в v4: v1 − v2 – v5 – v7 – v4, его длина равна 10.

λ7(3) + С74 = 9 + 1 = 10 = λ4(4)

λ5(2) + С57 = 5 + 4 = 9 = λ7(3)

λ2(1) + С25 = 3 + 2 = 5 = λ5(2)

λ1(0) + С12 = 0 + 3 = 3 = λ2(1)

* 1. Минимальный путь из v1 в v5: v1 − v2 – v5, его длина равна 5.

λ2(1) + С25 = 3 + 2 = 5 = λ5(2)

λ1(0) + С12 = 0 + 3 = 3 = λ2(1)

* 1. Минимальный путь из v1 в v6: v1 − v2 – v5– v6, его длина равна 7.

λ5(2) + С56 = 5 + 2 = 7 = λ6(3)

λ2(1) + С25 = 3 + 2 = 5 = λ5(2)

λ1(0) + С12 = 0 + 3 = 3 = λ2(1)

* 1. Минимальный путь из v1 в v7: v1 − v2 – v5 – v7, его длина равна 9.

λ5(2) + С57 = 5 + 4 = 9 = λ7(3)

λ2(1) + С25 = 3 + 2 = 5 = λ5(2)

λ1(0) + С12 = 0 + 3 = 3 = λ2(1)

* 1. Минимальный путь из v1 в v8: v1 − v2 – v5 – v7 – v4 – v8, его длина равна 13.

λ4(4) + С48 = 10 + 3 = 13 = λ8(5)

λ7(3) + С74 = 9 + 1 = 10 = λ4(4)

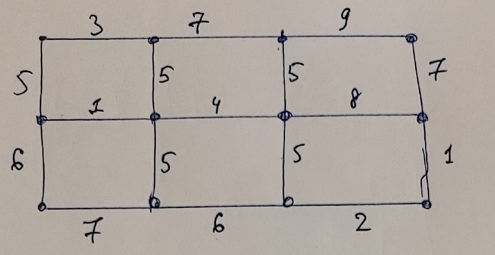
λ5(2) + С57 = 5 + 4 = 9 = λ7(3)

λ2(1) + С25 = 3 + 2 = 5 = λ5(2)

λ1(0) + С12 = 0 + 3 = 3 = λ2(1)

# **Задание №5**. Остовное дерево минимальной длины

Найти остовное дерево с минимальной суммой длин входящих в него ребер



**Решение:**

1) выбираем все вершины графа

2) добавляем все дуги, имеющие минимальный вес – 1

Циклов нет

3) добавляем все дуги, имеющие минимальный вес среди оставшихся – 2

Циклов нет

4) добавляем все дуги, имеющие минимальный вес среди оставшихся – 3

Циклов нет

5) добавляем все дуги, имеющие минимальный вес среди оставшихся – 4

Циклов нет

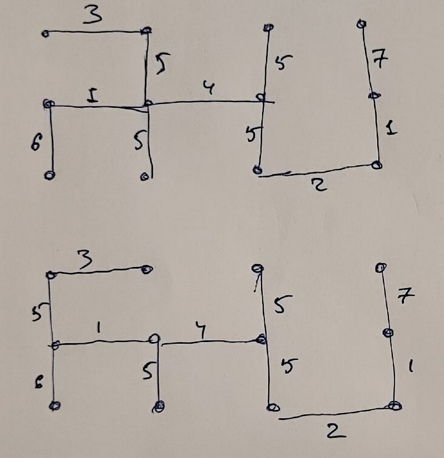
6) добавляем все дуги, имеющие минимальный вес – 5, так, чтобы не было циклов

7) добавляем все дуги, имеющие минимальный вес среди оставшихся – 6, так, чтобы не было циклов

8) добавляем все дуги, имеющие минимальный вес среди оставшихся – 7, так, чтобы не было циклов

Минимальный вес остовного дерева L(D) = 1\*2 + 2\*1 + 3\*1 + 4\*1 + 5\*4 + 6\*1 + 7\*1 = 44

Итого получаем 2 возможных варианта остовного дерева минимального веса.



# **Задание №6**. Деревья и циклы. Законы Кирхгофа

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 𝐸1 и 𝐸2, а остальные элементы являются сопротивлениями. Используя закон Ома, и, предполагая внутренние сопротивления источников тока равными нулю, получить систему уравнений для токов.



**2**

**8**

**9**

**4**

**11**

**1 6**

**7**

**3**

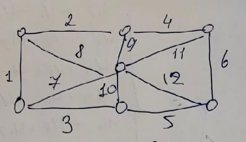
**10**

**12**

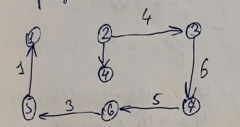
**5**

**Решение:**

1. Задаем на графе произвольную ориентацию



1. Строим произвольное остовное дерево D заданного графа



1. Находим базис циклов, добавляя к остовному дереву по одному не вошедшему в него ребру. Затем найдем соответствующие вектор-циклы.

(D+𝑞2): 𝜇1: 𝑣1−𝑣2–𝑣3−𝑣7− 𝑣6−𝑣5−𝑣1 ⟹ C(𝜇1) = (1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0);

(D+𝑞7): 𝜇2: 𝑣5 – 𝑣4 – 𝑣2 – 𝑣3 – 𝑣7− 𝑣6− 𝑣5 ⟹ C(𝜇2) = (0, 0, 1, 1, 1, 1, 1, 0, -1, 0, 0, 0);

(D+𝑞8): 𝜇3: 𝑣1 – 𝑣4 – 𝑣2 – 𝑣3 – 𝑣7− 𝑣6− 𝑣5 ⟹ C(𝜇3) = (1, 0, 1, 1, 1, 1, 0, 1, -1, 0, 0, 0);

(D+𝑞10): 𝜇4: 𝑣6 – 𝑣4 – 𝑣2 – 𝑣3 – 𝑣7− 𝑣6 ⟹ C(𝜇4) = (0, 0, 0, 1, 1, 1, 0, 0, -1, 1, 0, 0);

(D+𝑞11): 𝜇5: 𝑣3 – 𝑣4 – 𝑣2 – 𝑣3 ⟹ C(𝜇5) = (0, 0, 0, 1, 0, 0, 0, 0, -1, 0, 1, 0);

(D+𝑞12): 𝜇6: 𝑣7 – 𝑣4 – 𝑣2 – 𝑣3 – 𝑣7 ⟹ C(𝜇6) = (0, 0, 0, 1, 0, 1, 0, 0, -1, 0, 0, 1);

4) Цикломатическая матрица графа имеет вид:

C =

5) Выпишем закон Кирхгофа для напряжений:

\* = 0

Напряжения, соответствующие ребрам, не вошедшим в островное дерево 𝑢2, 𝑢7, 𝑢8, 𝑢10, 𝑢11, 𝑢12 – базисные переменные системы, остальные – свободные. Выражаем базисные переменные через свободные.

6) Выпишем уравнения Кирхгофа для токов. Найдем матрицу инцидентности 𝐵орграфа

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **q1** | **q2** | **q3** | **q4** | **q5** | **q6** | **q7** | **q8** | **q9** | **q10** | **q11** | **q12** |
| **v1** | 1 | -1 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 |
| **v2** | 0 | 1 | 0 | -1 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 |
| **v3** | 0 | 0 | 0 | 1 | 0 | -1 | 0 | 0 | 0 | 0 | -1 | 0 |
| **v4** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| **v5** | -1 | 0 | 1 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| **v6** | 0 | 0 | -1 | 0 | 1 | 0 | 0 | 0 | 0 | -1 | 0 | 0 |
| **v7** | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | -1 |

7) Запишем закон Кирхгофа для токов:

\* = 0

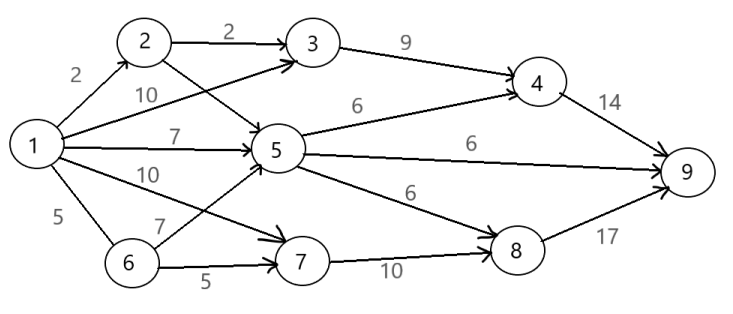
->

1. Подставляем формулы закона Ома
2. Совместная система имеет вид

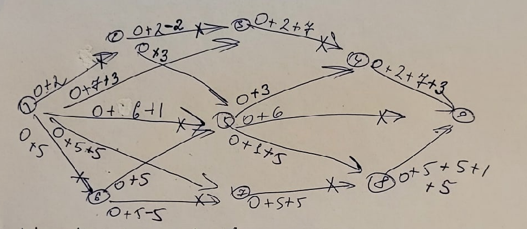
12 уравнений и 12 неизвестных – токи 𝐼1, 𝐼2, 𝐼3, 𝐼4, 𝐼5, 𝐼6, 𝐼7, 𝐼8, 𝐼9; ЭДС - 𝐸1, 𝐸2 и сопротивления 𝑅2, 𝑅3, 𝑅4, 𝑅6, 𝑅7, 𝑅8, 𝑅9 известны.

# **Задание №7**. Транспортные сети

Построить максимальный поток по данной транспортной сети:



**Решение:**



1) Построение полного потока.

Ищем пути из источника в сток, не содержащие насыщенных дуг.

1. 𝑣1 − 𝑣2 − 𝑣3 − 𝑣4 − 𝑣9

min {2, 2, 9, 14} = 2

2. 𝑣1 – 𝑣6 – 𝑣7 – 𝑣8 − 𝑣9

min {5, 5, 10, 17} = 5

3. 𝑣1 − 𝑣5 − 𝑣9

min {7, 6} = 6

4. 𝑣1 – 𝑣3 − 𝑣4 − 𝑣9

min {10, 9-2, 14-2} = 7

5. 𝑣1 – 𝑣7 – 𝑣8 − 𝑣9

min {10, 10-5, 17-5} = 5

6. 𝑣1 – 𝑣5 – 𝑣8 − 𝑣9

min {7-6, 6, 17-10} = 1

Величина полного потока Фполн = 9 + 6 + 11 = 26

2) Построение максимального потока.

Найдем увеличивающие цепи

1. 𝑣1 − 𝑣3 − 𝑣2 − 𝑣5 − 𝑣4 − 𝑣9

Δ1 = min {10 − 7, 2, 5, 6, 14 − 9} = 2

2. 𝑣1 − 𝑣7 − 𝑣6 − 𝑣5 − 𝑣8 − 𝑣9

Δ2 = min {10 − 5, 5, 7, 6 − 1, 17 − 11} = 5

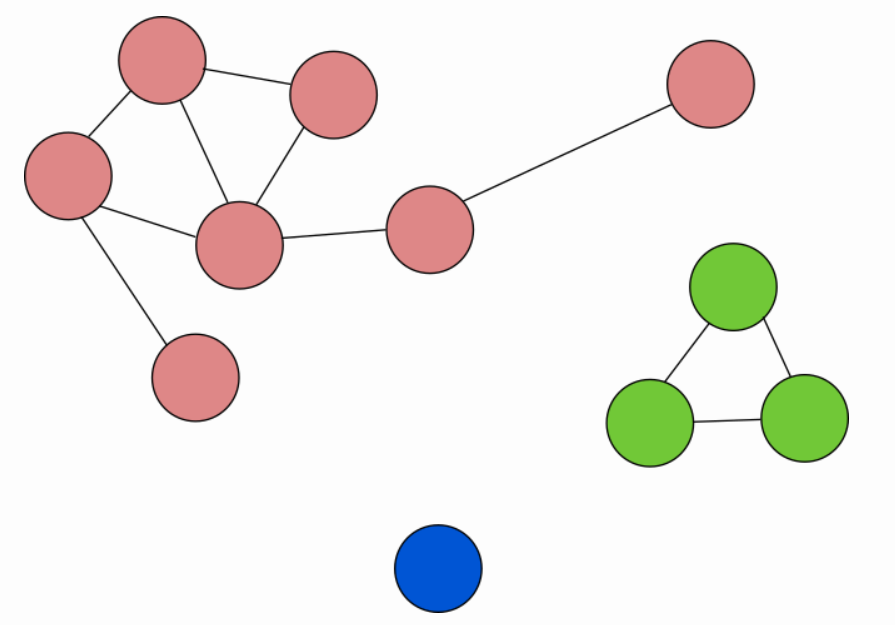
Величина максимального потока Фмакс = 11 + 6 + 16 = 33

**Задание №8 (вариант 30)**. Нахождение компонент связности неориентированного графа.

1. *Основные определения*

Понятие компоненты связности вытекает из понятия связности графа. Попросту говоря, компонента связности - часть графа (подграф), являющаяся связной. Формально, компонента связности - набор вершин графа, между любой парой которых существует путь.

Следующий содержит три компоненты связности, закрашенные разными цветами. Можно заметить, что даже одна вершина, изолированная от остального графа, составляет компоненту связности.



1. *Описание алгоритма*

А) На ввод подается количество вершина графа.

Б) Далее происходит ввод номеров вершин, в которые ведет данная вершина. Информация о ребрах хранится в множестве ( set() )

В) Создается массив, который будет хранить информацию о вершинах, которые были пройдены в результате работы функции DFS(). Также создается массив, хранящий информацию о компоненте связности вершины графа, массив заполняется нулями.

Г) Пока в этом массиве есть элемент, равный нулю, вызывается функция DFS. Каждый запуск функции увеличивает номер компонента связности графа.

В) В результате работы алгоритма получается массив, который хранит информацию о номерах компонентов связности вершин графа. В соответствие с этими данными будет происходить рисовка графа.

Описание функции DFS():

Входные параметры: graph, node, VisitedNodes, dfsVisited

graph – множество с информацией о ребрах графа

node – номер вершины

VisitedNodes – посещенные вершины

DfsVisited – вершины, которые будут посещённые в результате выполнения функции, этот массив будет возвращаться из функции

Если вершина, для которой вызвана функция еще никогда не было посещена, то она добавляется в массив посещенных вершин, также она добавляется в массив dfsVisited. Далее происходит поиск ребра в другую вершину. Если оно существует, то функция DFS() вызывается уже для нее. Если же проверяемая вершина уже была посещена ранее, то функция возвращает пустой массив.

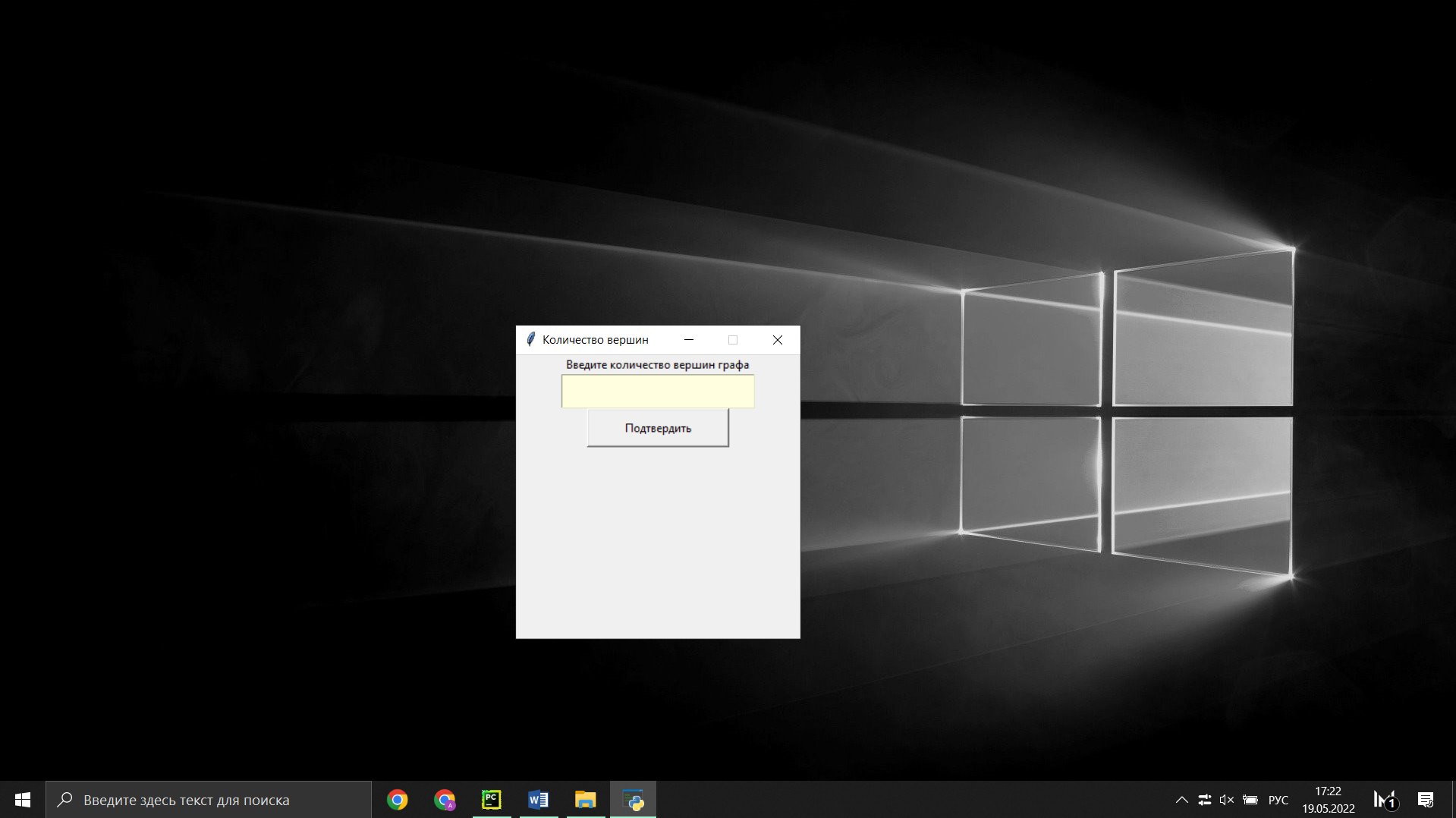
В результате алгоритма получается массив компонент связности. Далее происходит настройка отрисовки графа в соответствие с параметрами.

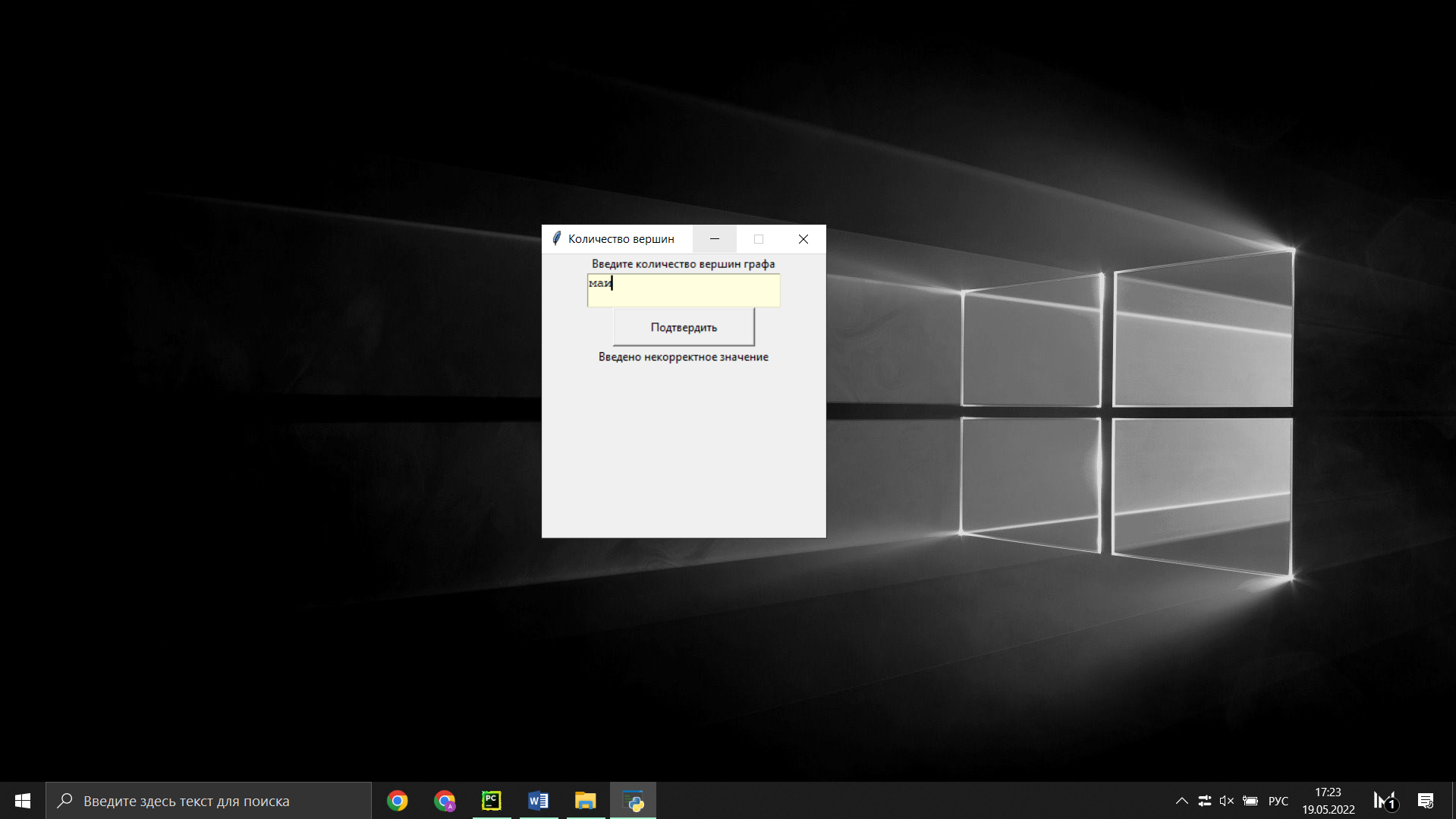
1. *Блок-схема*



1. *Описание программы и инструкция по работе с ней*

На первом этапе происходит ввод количества вершин графа. Выполняется проверка на корректность ввода





Далее необходимо ввести данные о связанности вершин графа.

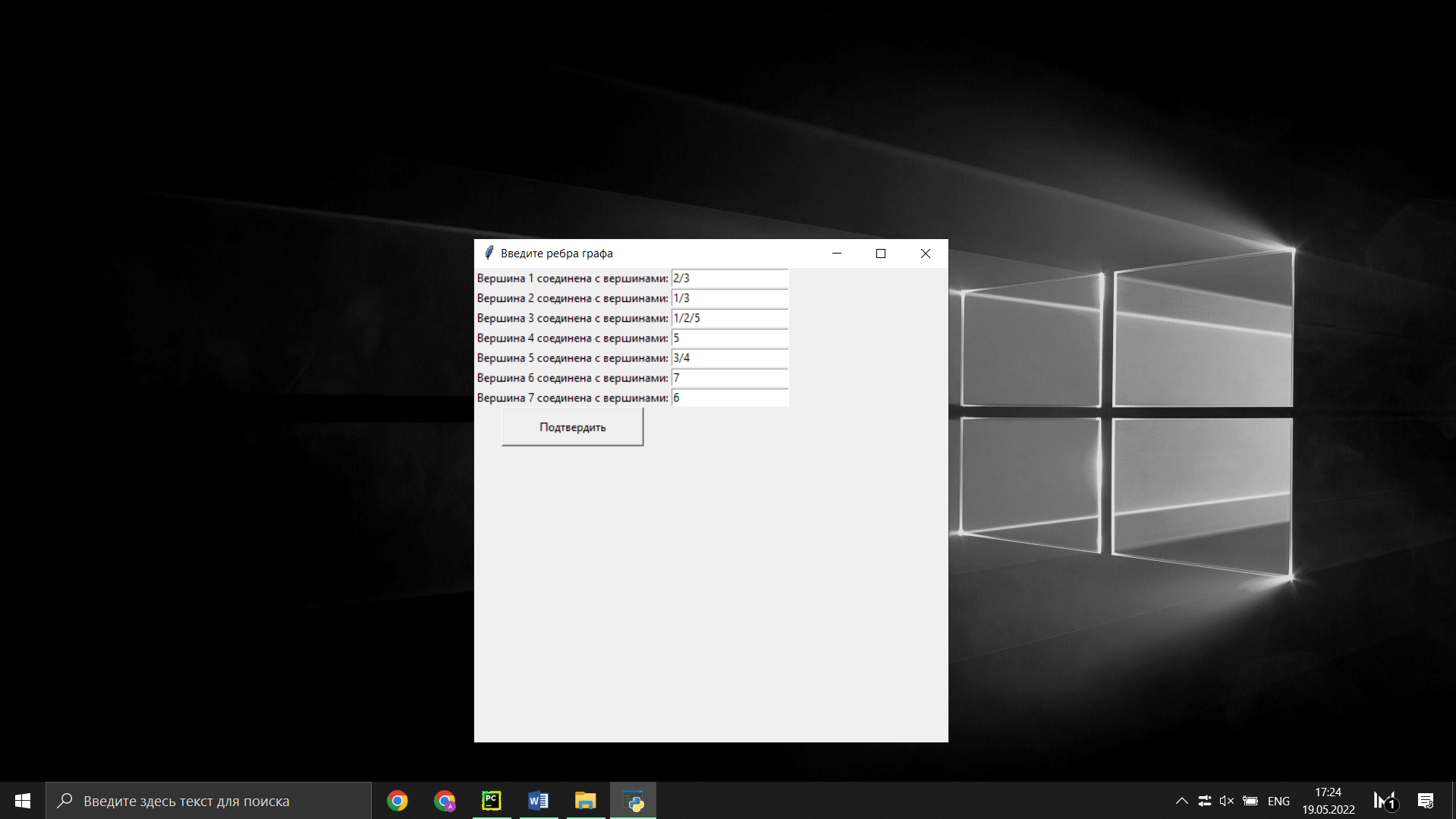
Если указывается, что вершина **a** соединена с вершиной **b**, то необходимо также указывать, что вершина **b** соединена с вершиной **a**.

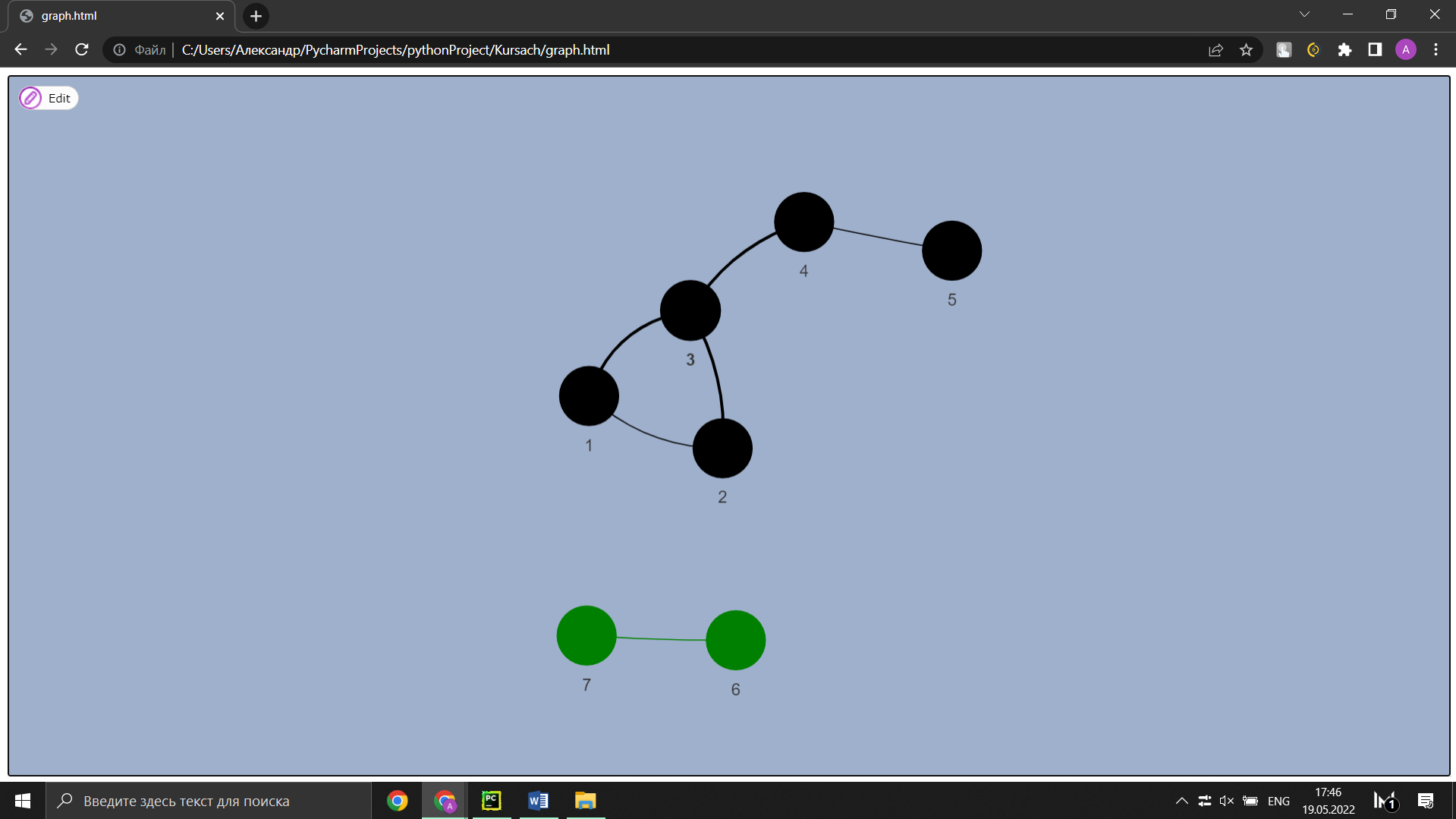
Вершина может не иметь ребер в другие вершины, тогда в поле ввода данные не вводятся

Номера вершин, в которые есть путь из данной вводятся по порядку

Если из одной вершины следует путь в несколько, то номер каждой вершины следует разделять символом **“ / ”**

*Пример:*



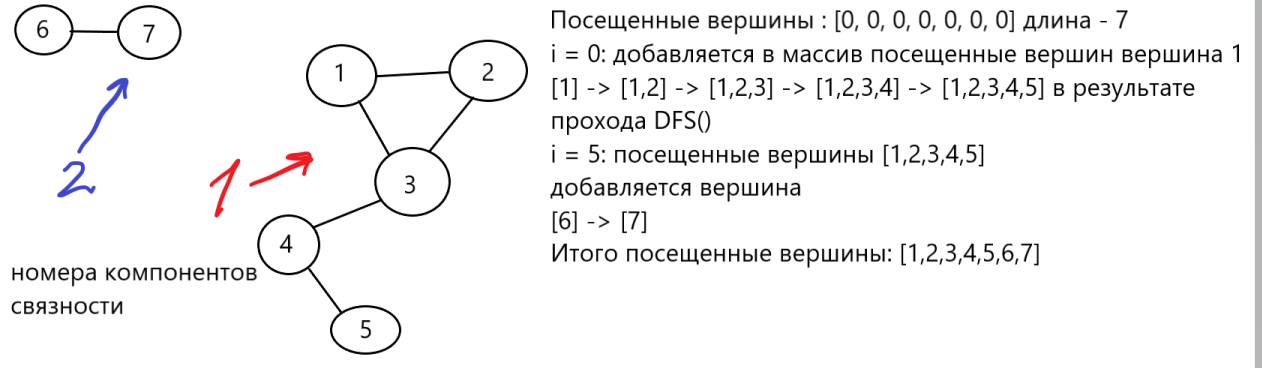


Разным цветом выделяются компоненты связности.

1. *Сложность*

Сложность алгоритма – **О(n+m)**, где *n* – количество вершин в графе, *m* – количество ребер в графе.

1. *Текстовый пример с решением*



1. Скриншот программы для данного примера

