Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и

программной инженерии

**ОТЧЕТ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ ПРАКТИКИ**

**(ознакомительной практики)**

**по направлению обучения**

**09.03.04 – Программная инженерия**

**с \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ по \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Выполнил:

студент группы ПРИ-120

Грачев Д.А.

Руководитель практики:

Шамышева О.Н.

Владимир, 2021

УТВЕРЖДАЮ

Зав. кафедрой ИСПИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.Е. Жигалов

“ \_\_\_\_\_“ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

**ЗАДАНИЕ** на ***учебную*** практику

(***ознакомительную практику***)

студента \_Грачева Даниила Алексеевича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

***1*** курса бакалавриата по направлению *09.03.04 «Программная инженерия»* ,

группы ***\_*** *ПРИ-120****\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Срок проведения практики *с 28.06.2021 по 11.07.2021*

Предприятие \_ВлГУ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Последовательность прохождения практики *кафедра ИСПИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

За время прохождения практики необходимо:

Изучить вопросы, предусмотренные программой по всем разделам.

1 . Задание по ознакомлению с предприятием (организацией) – базой практики: *Изучение комплекса технических средств базы практики* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2 . Задание по получению первичных профессиональных умений и навыков: *Выполнение трудовых действий по профессии в области ИТ (в соответствии с профессиональным стандартом)*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Задание по программированию:

3.1. *Рекурсивные алгоритмы. Рекурсивный перебор. Подсчёт значения арифметического выражения методом рекурсивного спуска. Компилятор математических формул.*\_\_\_\_

3.2. *Алгоритмы на графах. Выделение компонент связности. Поиск в ширину, поиск в глубину. Поиск кратчайшего пути: Алгоритм Дейкстры, алгоритм Флойда. Остовы. Алгоритм Крускала. Поиск циклов. Эйлеров путь. Гамильтонов путь*

4. Задание по стандартизации: *изучить нормативные документы высшего образования, профессиональной деятельности, оформления научно-технических документов\_\_\_\_\_\_\_\_\_*

Отчёт по практике составить к *08 .07.2021*

Задание выдал: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шамышева О.Н.

Подпись и ФИО руководителя практики от университета

Задание принял:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись студента, дата

СОДЕРЖАНИЕ

[КАЛЕНДАРНЫЙ ГРАФИК ПРОХОЖДЕНИЯ ПРАКТИКИ 5](#_Toc81689692)

[ВВЕДЕНИЕ 6](#_Toc81689693)

[1. Задание 0. Графические примитивы 8](#_Toc81689694)

[1.1. ЦЕЛЬ РАБОТЫ 8](#_Toc81689695)

[1.2. ВЫПОЛНЕНИЕ РАБОТЫ 8](#_Toc81689696)

[1.2.1. Задание 1 8](#_Toc81689697)

[1.2.2. Задание 2 9](#_Toc81689698)

[1.2.3. Задание 3 10](#_Toc81689699)

[1.2.4. Задание 4 12](#_Toc81689700)

[1.3. ВЫВОД 13](#_Toc81689701)

[2. Задание 1. Чтение графа из файла. Рисование графа. DFS 14](#_Toc81689702)

[2.1. ЦЕЛЬ РАБОТЫ 14](#_Toc81689703)

[2.2. ВЫПОЛНЕНИЕ РАБОТЫ 14](#_Toc81689704)

[2.3. ВЫВОД 17](#_Toc81689705)

[3. Задание 2. Алгоритм Дейкстры. Чтение графа из файла. Веса ребер графа хранятся в таблице. Рисование графа 18](#_Toc81689706)

[3.1. ЦЕЛЬ РАБОТЫ 18](#_Toc81689707)

[3.2. ВЫПОЛНЕНИЕ РАБОТЫ 18](#_Toc81689708)

[3.3. ВЫВОД 19](#_Toc81689709)

[4. Задание 3. Топологическая сортировка 20](#_Toc81689710)

[4.1. ЦЕЛЬ РАБОТЫ 20](#_Toc81689711)

[4.2. ВЫПОЛНЕНИЕ РАБОТЫ 20](#_Toc81689712)

[4.3. ВЫВОД 22](#_Toc81689713)

[5. Задание 4. Мосты 23](#_Toc81689714)

[5.1. ЦЕЛЬ РАБОТЫ 23](#_Toc81689715)

[5.2. ВЫПОЛНЕНИЕ РАБОТЫ 23](#_Toc81689716)

[5.3. ВЫВОД 25](#_Toc81689717)

[6. Задание 5. Волновой алгоритм 26](#_Toc81689718)

[6.1. ЦЕЛЬ РАБОТЫ 26](#_Toc81689719)

[6.2. ВЫПОЛНЕНИЕ РАБОТЫ 26](#_Toc81689720)

[6.3. ВЫВОД 28](#_Toc81689721)

[7. Задание 6. Остовное дерево.Алгоритм Крускала 29](#_Toc81689722)

[7.1. ЦЕЛЬ РАБОТЫ 29](#_Toc81689723)

[7.2. ВЫПОЛНЕНИЕ РАБОТЫ 29](#_Toc81689724)

[7.3. ВЫВОД 31](#_Toc81689725)

[8. Задание 7. Циклы 32](#_Toc81689726)

[8.1. ЦЕЛЬ РАБОТЫ 32](#_Toc81689727)

[8.2. ВЫПОЛНЕНИЕ РАБОТЫ 32](#_Toc81689728)

[8.3. ВЫВОД 34](#_Toc81689729)

[9. Задание 8. Гамильтонов цикл 35](#_Toc81689730)

[9.1. ЦЕЛЬ РАБОТЫ 35](#_Toc81689731)

[9.2. ВЫПОЛНЕНИЕ РАБОТЫ 35](#_Toc81689732)

[9.3. ВЫВОД 37](#_Toc81689733)

[10. Задание 9. Задание по выбору 38](#_Toc81689734)

[10.1. ЦЕЛЬ РАБОТЫ 38](#_Toc81689735)

[10.2. ВЫПОЛНЕНИЕ РАБОТЫ 38](#_Toc81689736)

[10.3. ВЫВОД 40](#_Toc81689737)

[ЗАКЛЮЧЕНИЕ 41](#_Toc81689738)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 42](#_Toc81689739)

# КАЛЕНДАРНЫЙ ГРАФИК ПРОХОЖДЕНИЯ ПРАКТИКИ

|  |  |
| --- | --- |
| Вид работы | Время выполнения |
| Получение задания. Ознакомление с техникой безопасности. | 28.06.2021 |
| Выполнение индивидуальных заданий | 29.06.2021 – 07.07.2021 |
| Оформление отчета | 08.07.2021 – 10.07.2021 |
| Защита отчета руководителю практики | 11.07.2021 |

# ВВЕДЕНИЕ

Целью учебной практики является закрепление и углубление теоретической подготовки обучающегося, расширение профессионального и общего кругозора студента, получение первичных профессиональных умений и навыков. Практика должна способствовать пониманию теоретических и практических проблем программной инженерии, профессиональной деятельности в информационном обществе, адаптации к рынку труда по направлению подготовки.

Цель практики соотнесена с общими целями ОПОП ВО, в соответствии с которой область профессиональной деятельности выпускников, освоивших программу бакалавриата, включает:

06 Связь, информационные и коммуникационные технологии (в сфере индустриального производства программного обеспечения для информационно-вычислительных систем различного назначения).

Основными объектами профессиональной деятельности выпускников являются:

* прикладные и информационные процессы;
* информационные технологии;
* программное обеспечение.

Практика в соответствии с ОПОП должна способствовать формированию готовности выпускника, освоившего программу бакалавриата, решать задачи профессиональной деятельности следующих типов: производственно технологического, проектного.

При прохождении учебной практики студенты закрепляют и углубляют теоретическую подготовку по дисциплинам первого года обучения, приобретают навыки практического использования вычислительной техники, первичные профессиональные умения и навыки в области программной инженерии за счёт решения следующих задач:

* изучение специальной литературы и другой научно-технической информации, достижений отечественной и зарубежной науки и техники в области программной инженерии;
* изучение и освоение комплекса технических и программных средств базы практики;
* выполнение практических заданий по углублённым темам дисциплин первого года обучения;
* участие в эксплуатации, сопровождении программно-информационных систем;
* оформление результатов анализа информации по заданной теме и собственных разработок в виде отчета.

# Задание 0. Графические примитивы

## ЦЕЛЬ РАБОТЫ

На языке Python научиться создавать фото и видео с помощью библиотек Pillow, numpy и opencv-python.

## ВЫПОЛНЕНИЕ РАБОТЫ

### Задание 1

Рисование графических примитивов на языке Python.

Листинг программы:

from PIL import Image,ImageDraw

img = Image.new("RGB",(500,500),"white")

draw = ImageDraw.Draw(img)

draw.line([(0,0),(100,100),(200,100)],"red",5)

draw.arc([(200,100),(300,200)],0,90,"blue",2)

draw.chord([(250,150),(350,250)],0,90,None,"blue",2)

draw.ellipse([(50,150),(200,200)],"green",None,0)

draw.point([(250,250)],"magenta")

draw.rectangle([(50,300),(100,350)],"magenta","green",2)

draw.polygon([(400,400),(400,500),(500,500)],"magenta",None)

img.show()

Скриншот работы программы представлен на Рис. 1.

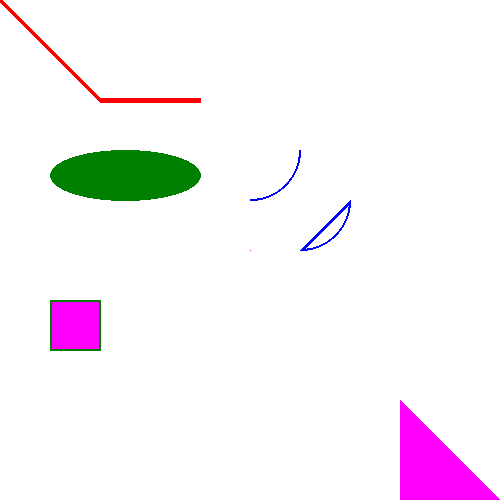


Рисунок 1. Графические примитивы

### Задание 2

Рисование фракталов

Листинг программы:

from PIL import Image, ImageDraw

*def* sq\_fr(*A*,*B*,*C*,*D*,*deep*=10,*alpha*=0.1):

    '''

    Построение фрактала квадрата по 4 точкам\n

    deep - глубина\n

    alpha - коэффициент погружения(>1 - расширяется, <1 - уменьшается)

    '''

    if deep < 1:

        return

    draw.polygon([A,B,C,D], None, 3)

    A1 = (A[0]\*(1-alpha)+B[0]\*alpha, A[1]\*(1-alpha)+B[1]\*alpha)

    B1 = (B[0]\*(1-alpha)+C[0]\*alpha, B[1]\*(1-alpha)+C[1]\*alpha)

    C1 = (C[0]\*(1-alpha)+D[0]\*alpha, C[1]\*(1-alpha)+D[1]\*alpha)

    D1 = (D[0]\*(1-alpha)+A[0]\*alpha, D[1]\*(1-alpha)+A[1]\*alpha)

    sq\_fr(A1,B1,C1,D1,deep-1,alpha)

img = Image.new("RGB", (700,700), "white")

draw = ImageDraw.Draw(img)

A = (5,5)

B = (695,5)

C = (695,695)

D = (5,695)

sq\_fr(A,B,C,D,*deep*=994,*alpha*=0.01)

img.show()

Скриншот работы программы представлен на Рис. 2.



Рисунок 2. Фрактал

### Задание 3

Передвижение круга по горизонтали

Листинг программы:

import numpy as np

from PIL import Image, ImageDraw

import cv2

videoDimensions = (1280,720)

videoFps = 75

videoCodec = "avc1"

videoFilename = "output.mp4"

videoLength = 3

videoFourcc = cv2.VideoWriter\_fourcc(\*videoCodec)

video = cv2.VideoWriter(videoFilename, videoFourcc, videoFps, videoDimensions)

img = Image.new("RGB", videoDimensions, *color* = "#FFBD9B")

imgDrawer = ImageDraw.Draw(img)

imgDrawer.line([(0,360), (1280,360)], *fill* = "#0A1D37", *width* = 1)

totalPercent = 0

for frameNumber in range(0, videoFps\*videoLength):

    frame = img.copy()

    frameDrawer = ImageDraw.Draw(frame)

    shift = (frameNumber / videoFps) \* (1330 / videoLength)

    frameDrawer.ellipse([(-50 + shift, 310), (shift, 360)], *fill* = "#0A1D37")

    video.write(cv2.cvtColor(np.array(frame), cv2.COLOR\_RGB2BGR))

    currentPercent = round(frameNumber / (videoFps\*videoLength)\*100)

    if currentPercent > totalPercent:

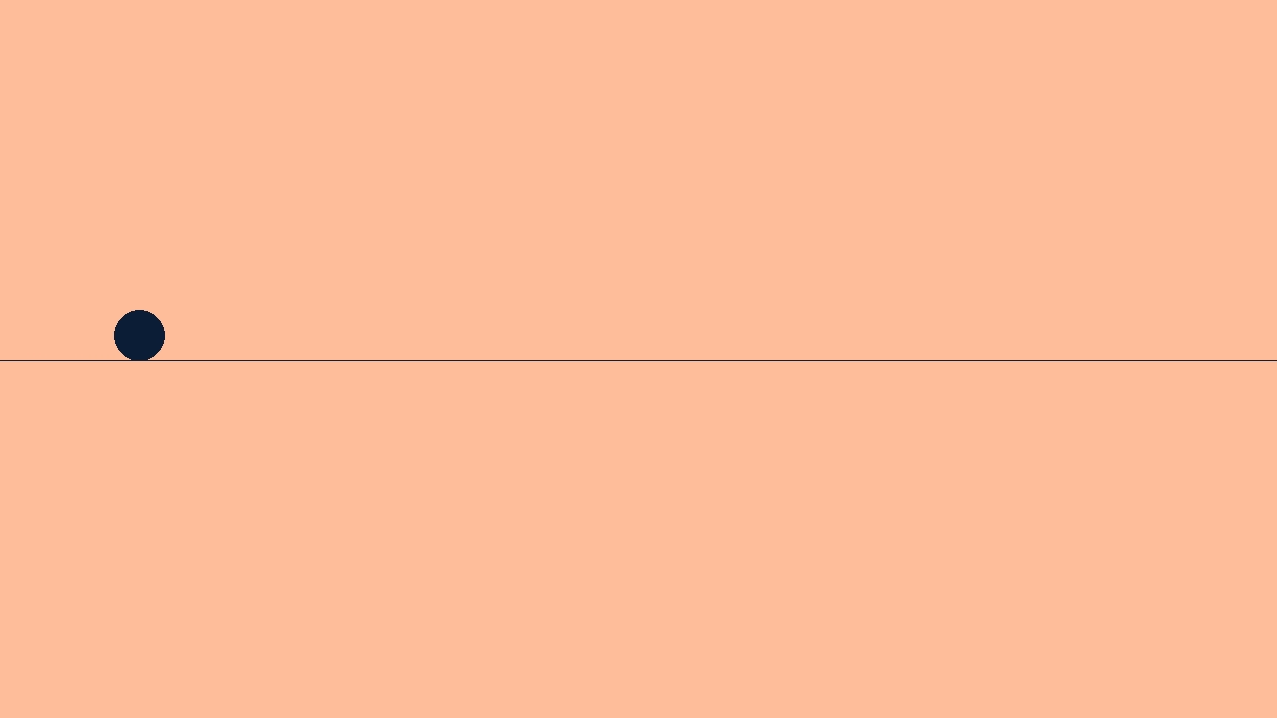
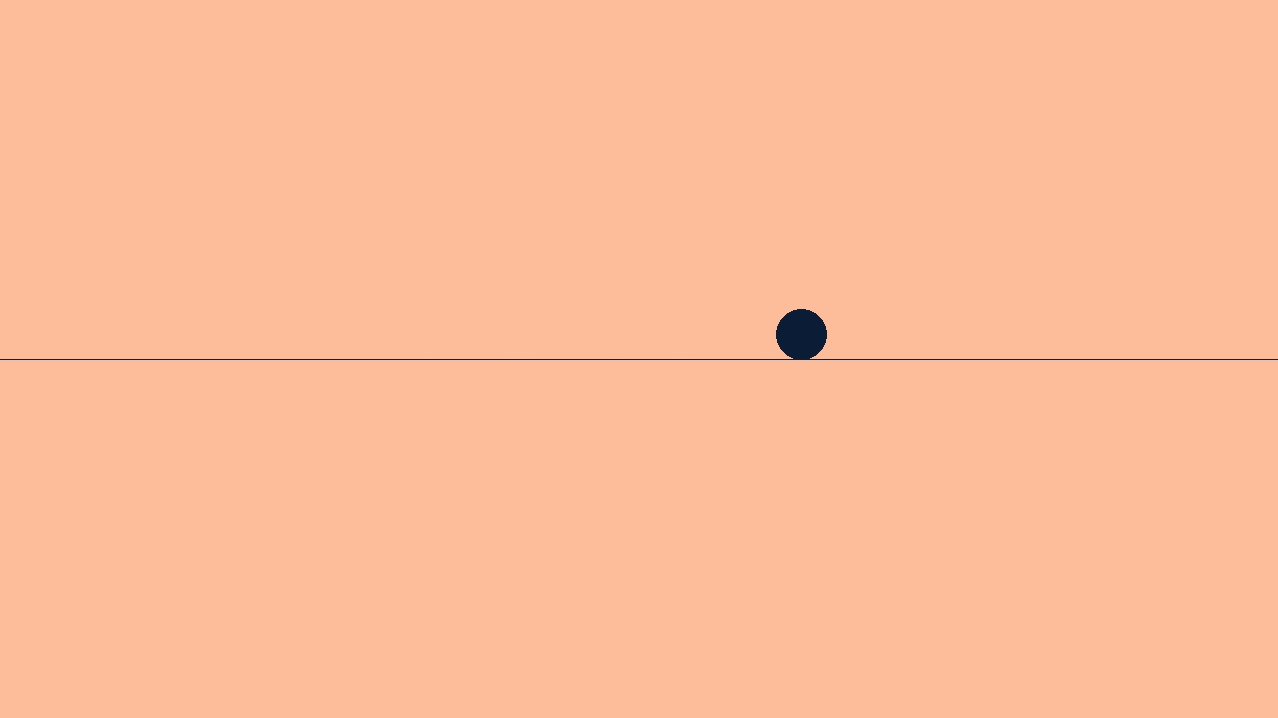
        totalPercent = currentPercent

        print(*f*"{currentPercent}% выполнено ")

video.release()

print(*f*"\nВидео с названием {videoFilename} готово!")

Скриншот работы программы представлен на Рис. 3.1 и Рис. 3.2.

Рисунки 3.1 и 3.2. Движение круга по горизонтали

### Задание 4

Визуализация фрактала

Листинг программы:

from PIL import Image, ImageDraw

import numpy as np

import cv2

videoDimensions = (1280,720)

videoFps = 75

videoCodec = "avc1"

videoFilename = "frac.mp4"

videoFourcc = cv2.VideoWriter\_fourcc(\*videoCodec)

video = cv2.VideoWriter(videoFilename, videoFourcc, videoFps, videoDimensions)

img = Image.new("RGB", videoDimensions, *color* = "#FFBD9B")

imgDrawer = ImageDraw.Draw(img)

frame = img.copy()

frameDrawer = ImageDraw.Draw(frame)

*def* sq\_fr(*A*,*B*,*C*,*D*,*deep*=10,*alpha*=0.1,*k*=0,*totalPercent*=0):

    '''

    Построение фрактала квадрата по 4 точкам\n

    deep - глубина\n

    alpha - коэффициент погружения(>1 - расширяется, <1 - уменьшается)

    '''

    if deep < 1:

        return

    draw.polygon([A,B,C,D], None, 3)

    A1 = (A[0]\*(1-alpha)+B[0]\*alpha, A[1]\*(1-alpha)+B[1]\*alpha)

    B1 = (B[0]\*(1-alpha)+C[0]\*alpha, B[1]\*(1-alpha)+C[1]\*alpha)

    C1 = (C[0]\*(1-alpha)+D[0]\*alpha, C[1]\*(1-alpha)+D[1]\*alpha)

    D1 = (D[0]\*(1-alpha)+A[0]\*alpha, D[1]\*(1-alpha)+A[1]\*alpha)

    frameDrawer.polygon([A,B,C,D], None, 3)

    video.write(cv2.cvtColor(np.array(frame), cv2.COLOR\_RGB2BGR))

    currentPercent = round(k / 900 \* 100)

    if currentPercent > totalPercent:

        totalPercent = currentPercent

        print(*f*"{currentPercent}% выполнено ")

    sq\_fr(A1,B1,C1,D1,deep-1,alpha,k+1,totalPercent)

img = Image.new("RGB", (700,700), "white")

draw = ImageDraw.Draw(img)

A = (5,5)

B = (695,5)

C = (695,695)

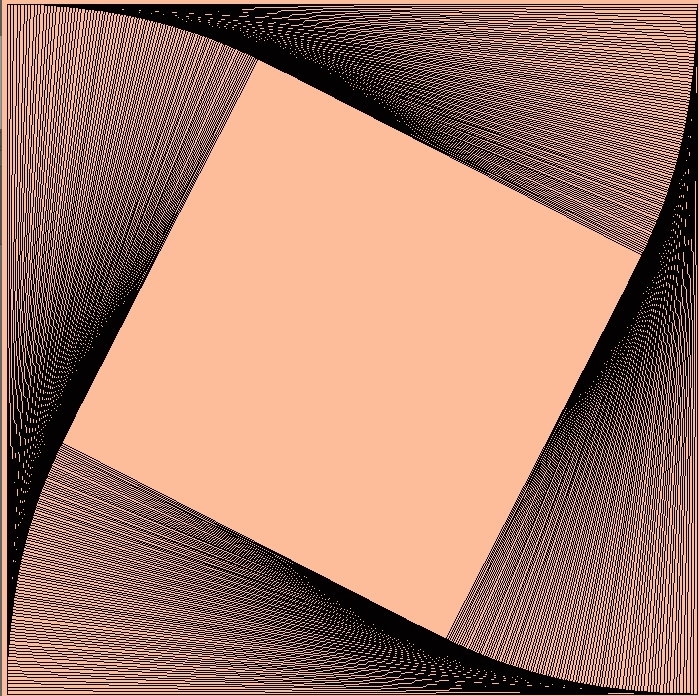
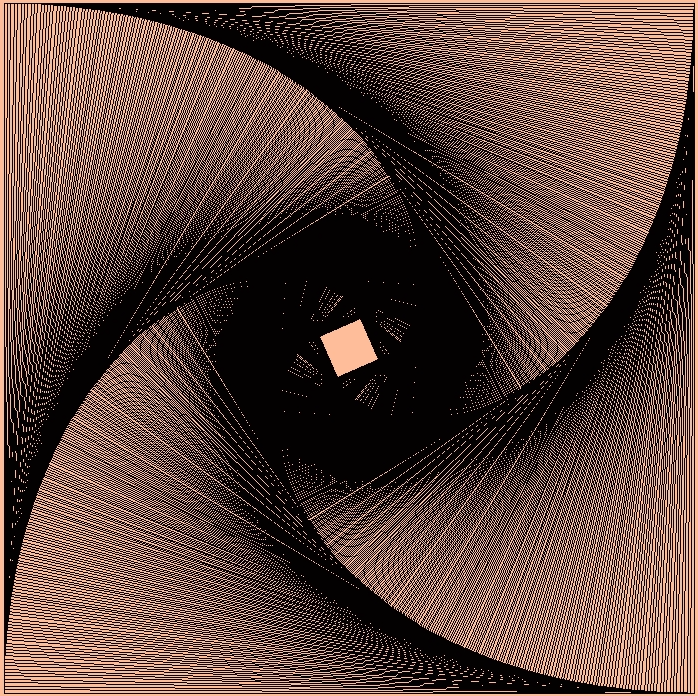
D = (5,695)

sq\_fr(A,B,C,D,*deep*=900,*alpha*=0.005)

video.release()

print(*f*"\nВидео с названием {videoFilename} готово!")

Скриншот работы программы представлен на Рис. 4.1 и Рис. 4.2.

Рисунки 4.1 и 4.2. Визуализация фрактала

## ВЫВОД

В ходе выполнения работы были получены практические навыки по созданию фото и видео с помощью библиотек Pillow, numpy и opencv-python.

# Задание 1. Чтение графа из файла. Рисование графа. DFS

## ЦЕЛЬ РАБОТЫ

Реализовать алгоритм обхода графа DFS, визуализировать граф на языке программирования Python.

## ВЫПОЛНЕНИЕ РАБОТЫ

Листинг программы:

from PIL import Image, ImageDraw, ImageFont

from random import randint

import numpy as np

import cv2

videoDimensions = (1280, 720)

videoFps = 75

videoCodec = "avc1"

videoFilename = "output.mp4"

videoLength = 3

videoFourcc = cv2.VideoWriter\_fourcc(\*videoCodec)

video = cv2.VideoWriter(videoFilename, videoFourcc, videoFps, videoDimensions)

img = Image.new("RGB", videoDimensions, *color* = "white")

imgDrawer = ImageDraw.Draw(img)

font = ImageFont.truetype("impact.ttf", 18) #шрифт для цифр

font\_text = ImageFont.truetype("impact.ttf", 30) #заголовки

#Чтение графа из файла

*def* graph\_from\_file():

    file = open("data.txt", "r")

    kolvo = *int*(file.readline())

    graf = [[0]\*kolvo for i in range(kolvo)]

    for i in range(1, kolvo):

        l = file.readline().rsplit()

        graf[*int*(l[0])][*int*(l[1])] = 1

        graf[*int*(l[1])][*int*(l[0])] = 1

    return graf, kolvo

#Создание вершин графа

*def* new\_coord\_1(*X*,*Y*):

    n = len(X)

    ni = *int*(n \*\* (1/2))

    nj = ni + 1

    if ni \*\* 2 == n:

        nj = ni

    elif ni\*nj < n:

        ni += 1

    a = (videoDimensions[0] - 100) // nj

    b = (videoDimensions[1] - 100) // ni

    x = 50

    y = 50

    z = 1

    for i in range(ni):

        for j in range(nj):

            print(z)

            X[z] = randint(x, x+a)

            Y[z] = randint(y, y+b)

            x = (x+a) % (videoDimensions[0] - 100)

            z += 1

            if z>=n:

                return X,Y

        y += b

#Рисование графа

*def* draw\_Graph(*A*,*X*,*Y*):

    for i in range(len(A)):

        for j in range(i+1, len(A)):

            if A[i][j] == 1:

                frame = img.copy()

                imgDrawer.line([(X[i], Y[i]), (X[j], Y[j])], "black", 2)

                frameDrawer = ImageDraw.Draw(frame)

                video.write(cv2.cvtColor(np.array(frame), cv2.COLOR\_RGB2BGR))

    for i in range(1, len(X)):

        frame = img.copy()

        imgDrawer.ellipse([(X[i] - 15, Y[i] - 15), (X[i] + 15, Y[i] + 15)], "white", "black", 2)

        imgDrawer.text((X[i] - 5, Y[i] - 5), *str*(5), (0,0,0), font)

        frameDrawer = ImageDraw.Draw(frame)

        video.write(cv2.cvtColor(np.array(frame), cv2.COLOR\_RGB2BGR))

#Обход графа dfs

*def* dfs(*A*,*v*,*color*,*p*,*z*):

    color[v] = 1

    z.append(v)

    for u in range(1, len(A)):

        if A[v][u] != 0 and color[u] == 0:

            p[u] = v

            dfs(A,u,color,p,z)

    color[v] = 2

#рисование линий, кругов и цифр

*def* draw\_dfs(*p*):

    for u in range(1, len(p)):

        if p[u] != 0:

            frame = img.copy()

            imgDrawer.line([(X[p[u]], Y[p[u]]), (X[u], Y[u])], "red", 5)

            frameDrawer = ImageDraw.Draw(frame)

            video.write(cv2.cvtColor(np.array(frame), cv2.COLOR\_RGB2BGR))

    for i in range(1, len(X)):

            frame = img.copy()

            imgDrawer.ellipse([(X[i] - 15, Y[i] - 15), (X[i] + 15, Y[i] + 15)], "white", "black", 2)

            imgDrawer.text((X[i] - 4, Y[i] - 10), *str*(i), (0,0,0), font)

            frameDrawer = ImageDraw.Draw(frame)

            video.write(cv2.cvtColor(np.array(frame), cv2.COLOR\_RGB2BGR))

A, N = graph\_from\_file()

color = [0]\*N

p = [0]\*N

d = [0]\*N

fi = [0]\*N

X = [0]\*N

Y = [0]\*N

z = []

new\_coord\_1(X,Y)

draw\_Graph(A,X,Y)

for v in range(1, len(A)):

    if color[v] == 0:

        dfs(A,v,color,p,z)

draw\_dfs(p)

frame = img.copy()

imgDrawer.text((100, 0), 'Graph: ', (0,0,0), font\_text)

frameDrawer = ImageDraw.Draw(frame)

video.write(cv2.cvtColor(np.array(frame), cv2.COLOR\_RGB2BGR))

for i in range(len(z)):

    frame = img.copy()

    imgDrawer.text((200,650), 'DFS: ', (0,0,0), font\_text)

    imgDrawer.text((400 + i\*20,650), *str*(z[i]), (0,0,0), font\_text)

    frameDrawer = ImageDraw.Draw(frame)

    video.write(cv2.cvtColor(np.array(frame), cv2.COLOR\_RGB2BGR))

img.save("draw\_dfs.png", "PNG")

img.show()

video.release()

Скриншот работы программы и данные из файла data.txt представлены на Рис.5.1 и 5.2.

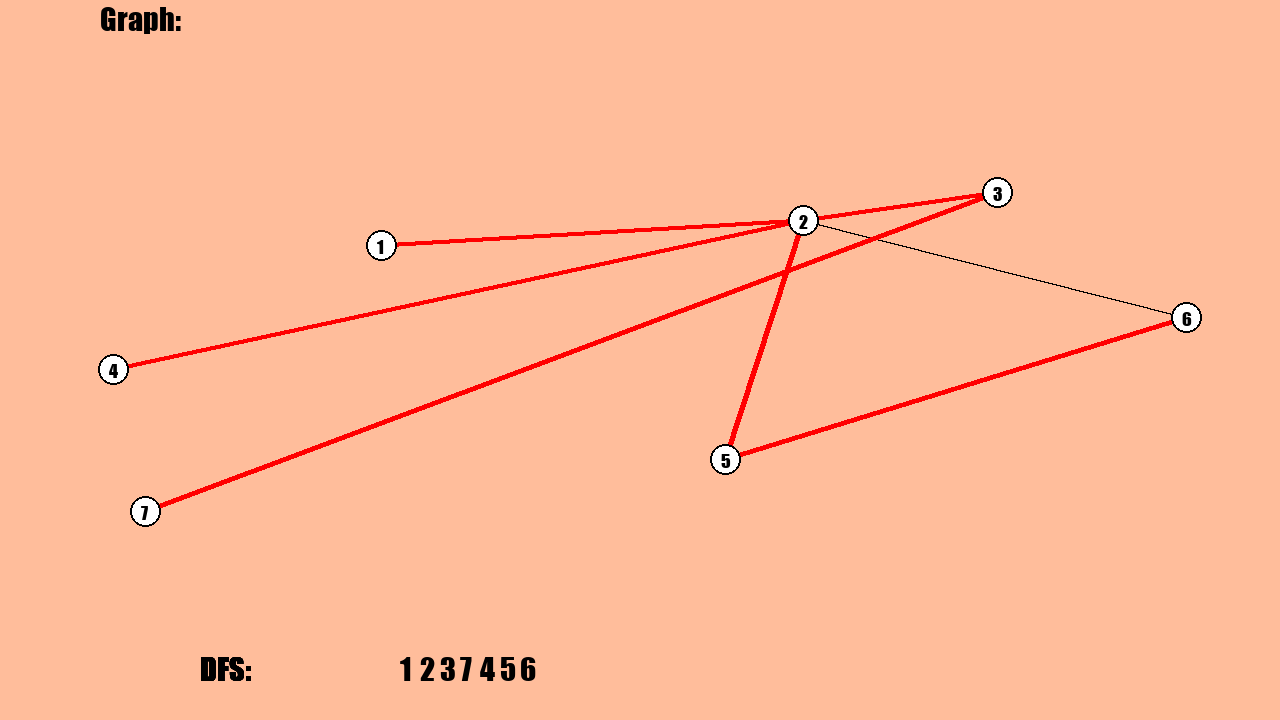
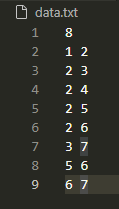
 

Рисунок 5.1 и 5.2. Результат работы программы и data.txt

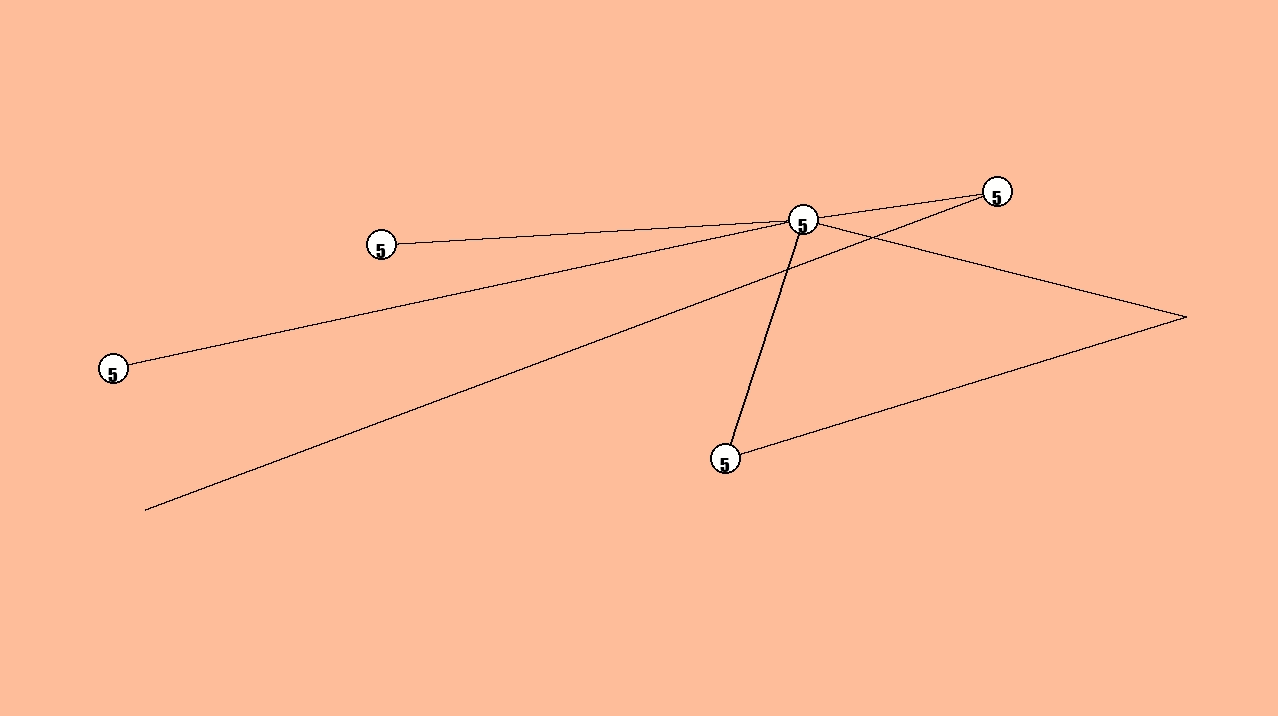
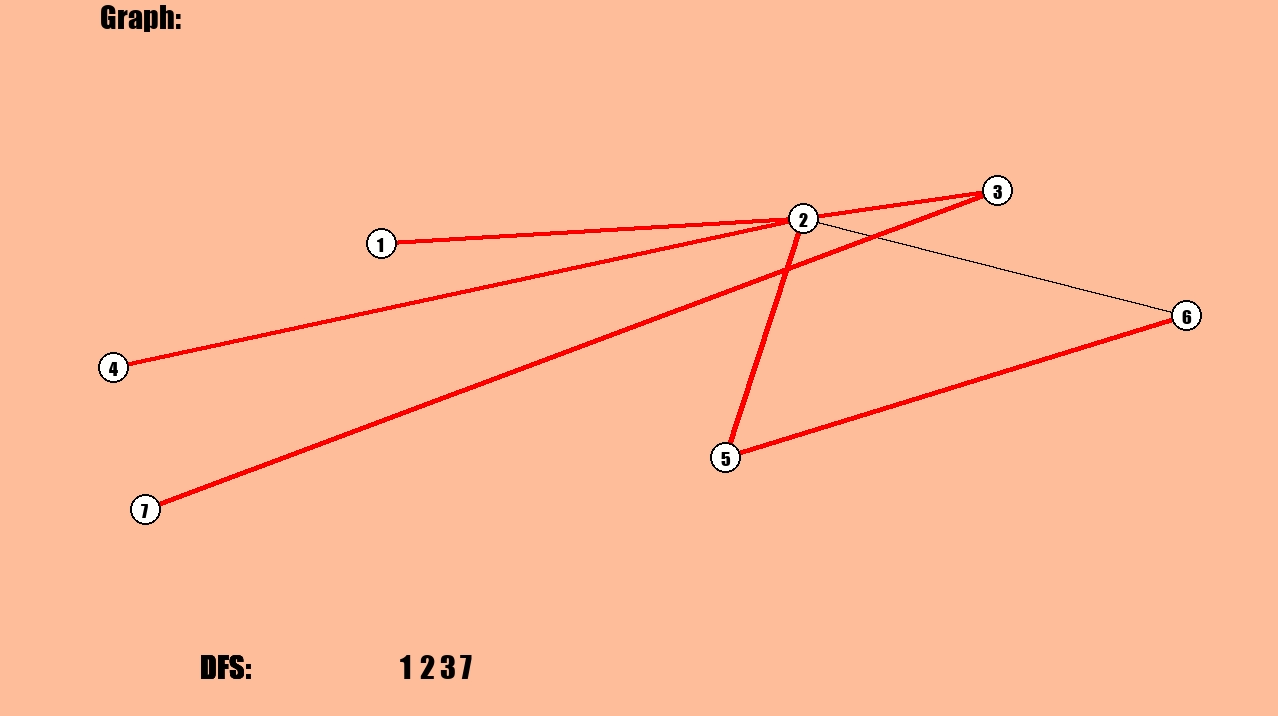
 

Рисунок 6. Скриншоты из видео

## ВЫВОД

В ходе выполнения работы были получены практические навыки по реализации алгоритма обхода графа DFS, визуализирован граф на языке программирования Python.

# Задание 2. Алгоритм Дейкстры. Чтение графа из файла. Веса ребер графа хранятся в таблице. Рисование графа

## ЦЕЛЬ РАБОТЫ

Реализовать алгоритм Дейкстры на языке Python

## ВЫПОЛНЕНИЕ РАБОТЫ

Листинг программы:

def Dijkstra(W,N,start):

    F = [100000]\*N

    F[start] = 0

    V = [True]\*N

    P = [0]\*n

    while True:

        min\_R = 100000

        min\_v = None

        for i in range(N):

            if V[i] and F[i] < min\_R:

                min\_R  = F[i]

                min\_v = i

        if min\_v is None:

            break

        for j in range(N):

            if V[j] and W[min\_v][j] != 0 and F[j] > F[min\_v]+W[min\_v][j]:

                F[j] = F[min\_v]+W[min\_v][j]

                P[j] = min\_v

        V[min\_v] = False

    puti = [[i] for i in range(n)]

    for i in range(n):

        min\_v = i

        while min\_v != start:

            min\_v = P[min\_v]

            puti[i].insert(0,min\_v)

    return F,puti

def vivod(matrix):

    print("Путь:")

    for i in range(len(matrix)):

        print("Путь до вершины ", i)

        for j in range(len(matrix[i])):

            print ( "{:4d}".format(matrix[i][j]), end = "" )

        print ()

f = open('2 semestr/practice/Zadanie2/vesa.txt', 'r')

line = [line.strip() for line in f]

n = int(line[0])

A = [[0]\*n for i in range(n)]

for i in range(1, len(line)):

    k = int(line[i][4])\*10 + int(line[i][5])

    A[int(line[i][0])][int(line[i][2])] = k

A[0][2] = 6

A[0][3] = 99

A[0][4] = 25

A[1][0] = 34

A[1][4] = 37

A[2][0] = 7

A[2][1] = 23

A[3][1] = 29

A[4][2] = 11

start = int(input('Введите начальное значение: '))

F,path = Dijkstra(A,n,start)

print(\*F)

vivod(path)

Скриншот работы программы и данные из файла vesa.txt представлены на Рис.7.1 и 7.2.

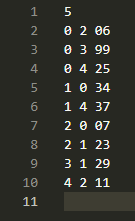
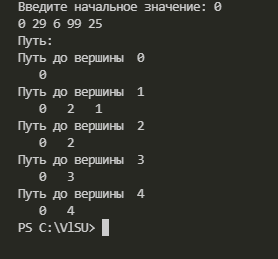


Рисунок 7.1 и 7.2. Результат работы программы и vesa.txt

## ВЫВОД

В ходе выполнения работы были получены практические навыки по реализации алгоритма Дейкстры на языке программирования Python.

# Задание 3. Топологическая сортировка

## ЦЕЛЬ РАБОТЫ

Реализовать алгоритм топологической сортировки в ориентированном графе.

## ВЫПОЛНЕНИЕ РАБОТЫ

Листинг программы:

from PIL import Image, ImageDraw, ImageFont

from random import randint

videoDimensions = (1280, 720)

img = Image.new("RGB", videoDimensions, color = "white")

imgDrawer = ImageDraw.Draw(img)

font = ImageFont.truetype("arial.ttf", 16)

font\_text = ImageFont.truetype("arial.ttf", 30)

def graph\_from\_file():

   f = open("2 semestr/practice/Zadanie3/graph.txt", "r")

   l = f.readline().rsplit()

   N = int(l[0])

   N\_R = int(l[1])

   A = [[0]\*N for i in range(N)]

   for i in range(1, N\_R + 1):

      l = f.readline().rsplit()

      A[int(l[0])][int(l[1])] = 1

   return A, N

def new\_coord\_1(X,Y):

   n = len(X)

   ni = int(n \*\* (1/2))

   nj = ni + 1

   if ni \*\* 2 == n:

      nj = ni

   elif ni\*nj < n:

      ni += 1

   a = (videoDimensions[0] - 100) // nj

   b = (videoDimensions[1] - 100) // ni

   x = 50

   y = 50

   z = 1

   for i in range(ni):

      for j in range(nj):

         print(z)

         X[z] = randint(x, x+a)

         Y[z] = randint(y, y+b)

         x = (x+a) % (videoDimensions[0] - 100)

         z += 1

         if z>=n:

            return X,Y

      y += b

def draw\_Graph(A,X,Y):

   for i in range(len(A)):

      for j in range(1, len(A)):

         if A[i][j] == 1:

            imgDrawer.line([(X[i], Y[i]), (X[j], Y[j])], "black", 2)

   for i in range(1, len(X)):

      imgDrawer.ellipse([(X[i] - 15, Y[i] - 15), (X[i] + 15, Y[i] + 15)], "white", "black", 2)

      imgDrawer.text((X[i] - 5, Y[i] - 5), str(i), (0,0,0), font)

def dfs(A,v, color, list):

   color[v] = 1

   for i in range(1, len(A)):

      if A[v][i] == 1 and color[i] == 0:

         dfs(A,i, color, list)

   k[0] += 1

   list[len(A)-k[0]] = v

   color[v] = 2

A, N = graph\_from\_file()

color = [0]\*N

list = [0]\*N

X = [0]\*N

Y = [0]\*N

new\_coord\_1(X,Y)

draw\_Graph(A,X,Y)

k = [0]

for v in range(1, N):

   if color[v] == 0:

      dfs(A, v, color, list)

imgDrawer.text((200,650), 'Результат: ', (0,0,0), font\_text)

for i in range(1, len(list)):

   imgDrawer.text((400 + i\*20,650), str(list[i]), (0,0,0), font\_text)

img.show()

img.save("2 semestr/practice/Zadanie3/top3.png", "PNG")

Скриншот работы программы и данные из файла graph.txt представлены на Рис.8.1 и 8.2.

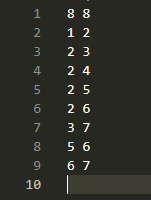
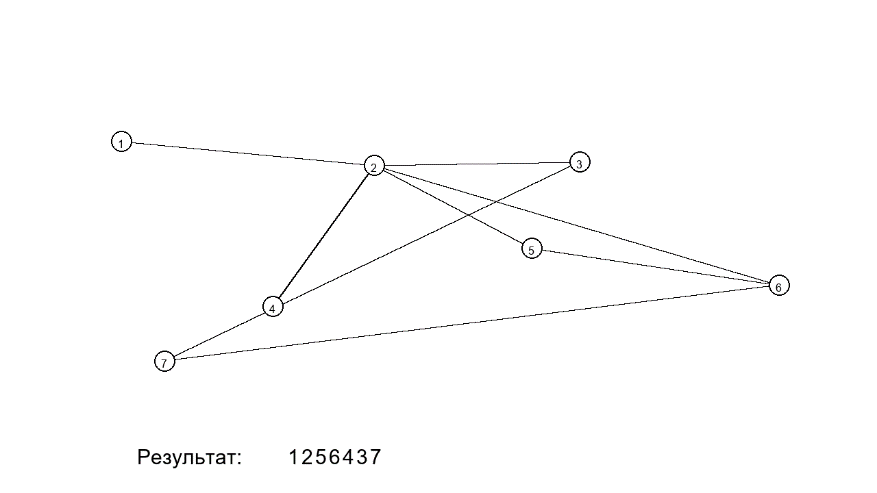


Рисунок 8.1 и 8.2. Результат работы программы и graph.txt

## ВЫВОД

В ходе выполнения работы были получены практические навыки по реализации алгоритма топологической сортировки на языке Python.

# Задание 4. Мосты

## ЦЕЛЬ РАБОТЫ

Реализовать алгоритм поиска мостов в графе используя DFS на языке программирования Python.

## ВЫПОЛНЕНИЕ РАБОТЫ

Листинг программы:

from PIL import Image, ImageDraw, ImageFont

from random import randint

videoDimensions = (1280, 1280)

img = Image.new("RGB", videoDimensions, color = '#FFE4C4')

imgDrawer = ImageDraw.Draw(img)

font = ImageFont.truetype('C:\Windows\Fonts\Arial.ttf', 30)

file = open('2 semestr/practice/Zadanie4/data.txt', 'r')

N = int(file.readline())

A = []

for i in range(N):

    A.append([])

while True:

    tmp = file.readline().rsplit()

    if tmp == []:

        break

    tmp[0], tmp[1] = int(tmp[0]) - 1, int(tmp[1]) - 1

    A[tmp[0]].append(tmp[1])

    A[tmp[1]].append(tmp[0])

used = [False]\*N

tin, fup = [0] \* N, [0] \* N

bridges = []

def dfs (v, timer, p = -1):

    used[v] = True

    tin[v] = timer

    fup[v] = timer

    for to in A[v]:

        if (to == p):

            continue

        if (used[to]):

            fup[v] = min (fup[v], tin[to])

        else:

            dfs (to, timer + 1, v)

            fup[v] = min (fup[v], fup[to])

            if (fup[to] > tin[v]):

                bridges.append([v, to])

def find():

    for i in range(N):

        if(not(used[i])):

            dfs(i, 0)

def new\_coord(X,Y):

    n = len(X)

    Y[0] = 10

    for i in range(n):

        X[i] = randint(50, videoDimensions[0] - 100)

        Y[i] = Y[i - 1] + videoDimensions[1] // n - 10

    return X, Y

def draw\_graph(x, y):

    for i in range(N):

        for j in A[i]:

            imgDrawer.line([(x[i], y[i]), (x[j], y[j])], 'black', 2)

    for i in range(N):

        imgDrawer.ellipse([(x[i] - 25, y[i] - 25), (x[i] + 25, y[i] + 25)], 'white', 'black', 2)

        imgDrawer.text((x[i] - 9, y[i] - 12), str(i), (0, 0, 0), ImageFont.truetype('C:\Windows\Fonts\Arial.ttf', 30))

def draw\_bridge(x, y):

    for i in bridges:

        imgDrawer.line([(x[i[0]], y[i[0]]), (x[i[1]], y[i[1]])], 'red', 4)

        imgDrawer.ellipse([(x[i[0]] - 25, y[i[0]] - 25), (x[i[0]] + 25, y[i[0]] + 25)], 'white', 'black', 2)

        imgDrawer.ellipse([(x[i[1]] - 25, y[i[1]] - 25), (x[i[1]] + 25, y[i[1]] + 25)], 'white', 'black', 2)

        imgDrawer.text((x[i[0]] - 9, y[i[0]] - 12), str(i[0]), (0, 0, 0), ImageFont.truetype('C:\Windows\Fonts\Arial.ttf', 30))

        imgDrawer.text((x[i[1]] - 9, y[i[1]] - 12), str(i[1]), (0, 0, 0), ImageFont.truetype('C:\Windows\Fonts\Arial.ttf', 30))

X = [0]\*N

Y = [0]\*N

new\_coord(X, Y)

draw\_graph(X, Y)

find()

draw\_bridge(X, Y)

imgDrawer.text((100, 1200), 'Путь: ', (0,0,0), font)

imgDrawer.text((200, 1200), str(bridges), (0,0,0), font)

img.show()

Скриншот работы программы и данные из файла data.txt представлены на Рис.9.1 и 9.2.

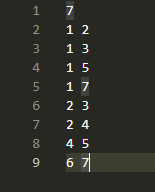
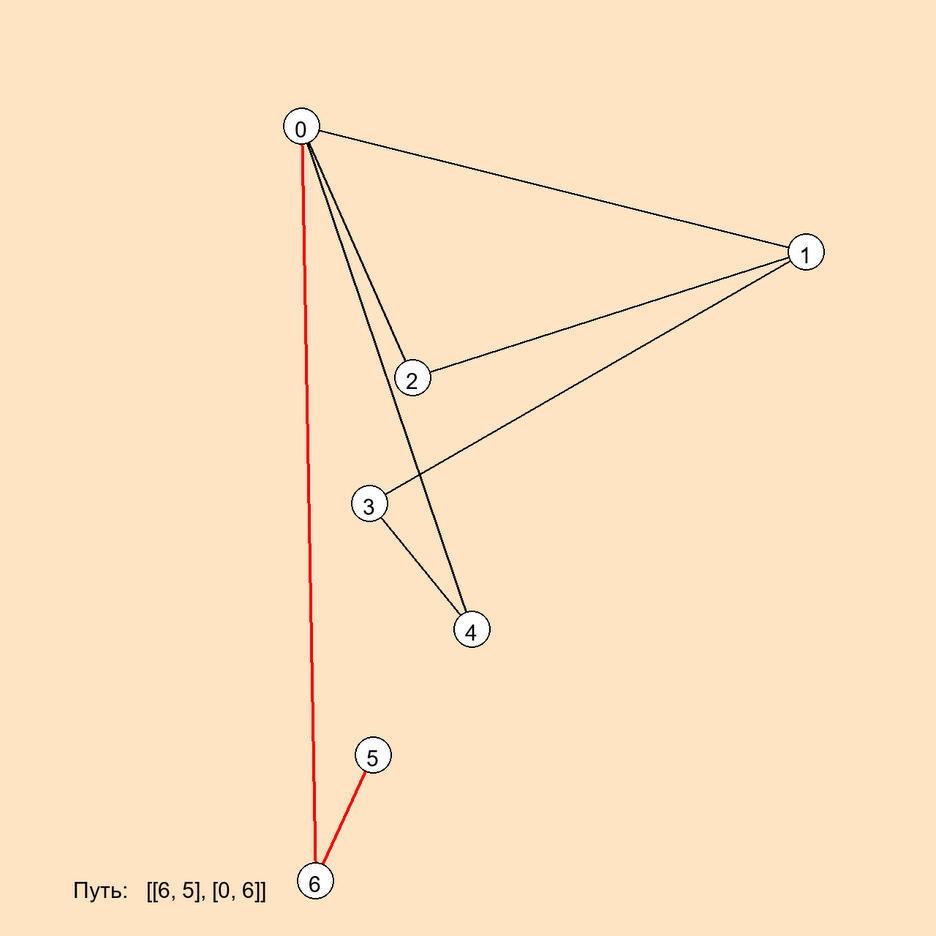


Рисунок 9.1 и 9.2. Результат работы программы и data.txt

## ВЫВОД

В ходе выполнения работы были получены практические навыки по использованию алгоритма DFS для поиска мостов в графе.

# Задание 5. Волновой алгоритм

## ЦЕЛЬ РАБОТЫ

Реализовать волновой алгоритм для поиска кратчайшего пути в графе на языке программирования Python.

## ВЫПОЛНЕНИЕ РАБОТЫ

Листинг программы:

from PIL import Image, ImageDraw, ImageFont

from random import randint

videoDimensions = (1280, 1280)

img = Image.new("RGB", videoDimensions, color = 'white')

imgDrawer = ImageDraw.Draw(img)

file = open('2 semestr/practice/Zadanie5/data.txt', 'r')

N = int(file.readline())

A = []

for i in range(N):

    A.append([])

while True:

    tmp = file.readline().rsplit()

    if tmp == []:

        break

    tmp[0], tmp[1] = int(tmp[0]) - 1, int(tmp[1]) - 1

    A[tmp[0]].append(tmp[1])

    A[tmp[1]].append(tmp[0])

def new\_coord(X,Y):

    n = len(X)

    Y[0] = 10

    for i in range(n):

        X[i] = randint(50, videoDimensions[0] - 100)

        Y[i] = Y[i - 1] + videoDimensions[1] // n - 10

    return X, Y

def draw\_graph(x, y):

    for i in range(N):

        for j in A[i]:

            imgDrawer.line([(x[i], y[i]), (x[j], y[j])], 'black', 2)

    for i in range(N):

        imgDrawer.ellipse([(x[i] - 25, y[i] - 25), (x[i] + 25, y[i] + 25)], 'white', 'black', 2)

        imgDrawer.text((x[i] - 9, y[i] - 12), str(i + 1), (0, 0, 0), ImageFont.truetype('C:\Windows\Fonts\Arial.ttf', 30))

def draw\_route():

    for i in range(len(route) - 1):

        imgDrawer.line([(X[route[i]], Y[route[i]]), (X[route[i + 1]], Y[route[i + 1]])], 'blue', 4)

    for i in route:

        imgDrawer.ellipse([(X[i] - 25, Y[i] - 25), (X[i] + 25, Y[i] + 25)], 'white', 'black', 2)

        imgDrawer.text((X[i] - 9, Y[i] - 12), str(i + 1), (0, 0, 0), ImageFont.truetype('C:\Windows\Fonts\Arial.ttf', 30))

def wave(d):

    used[d] = True

    for i in A[d]:

        if times[i] == 0:

            times[i] = times[d] + 1;

            if i == N - 1: return;

    for i in A[d]:

        if not(used[i]):

            wave(i)

def restore(d):

    k = times[d]

    for i in A[d]:

        if times[i] == k - 1:

            route.append(i)

            restore(i)

X = [0]\*N

Y = [0]\*N

used = [False]\*N

times = [0]\*N

times[0] = 1

route = []

route.append(N - 1)

wave(0)

restore(N - 1)

new\_coord(X, Y)

draw\_graph(X, Y)

draw\_route()

img.show()

Скриншот работы программы и данные из файла data.txt представлены на Рис.10.1 и 10.2.

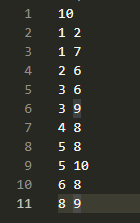
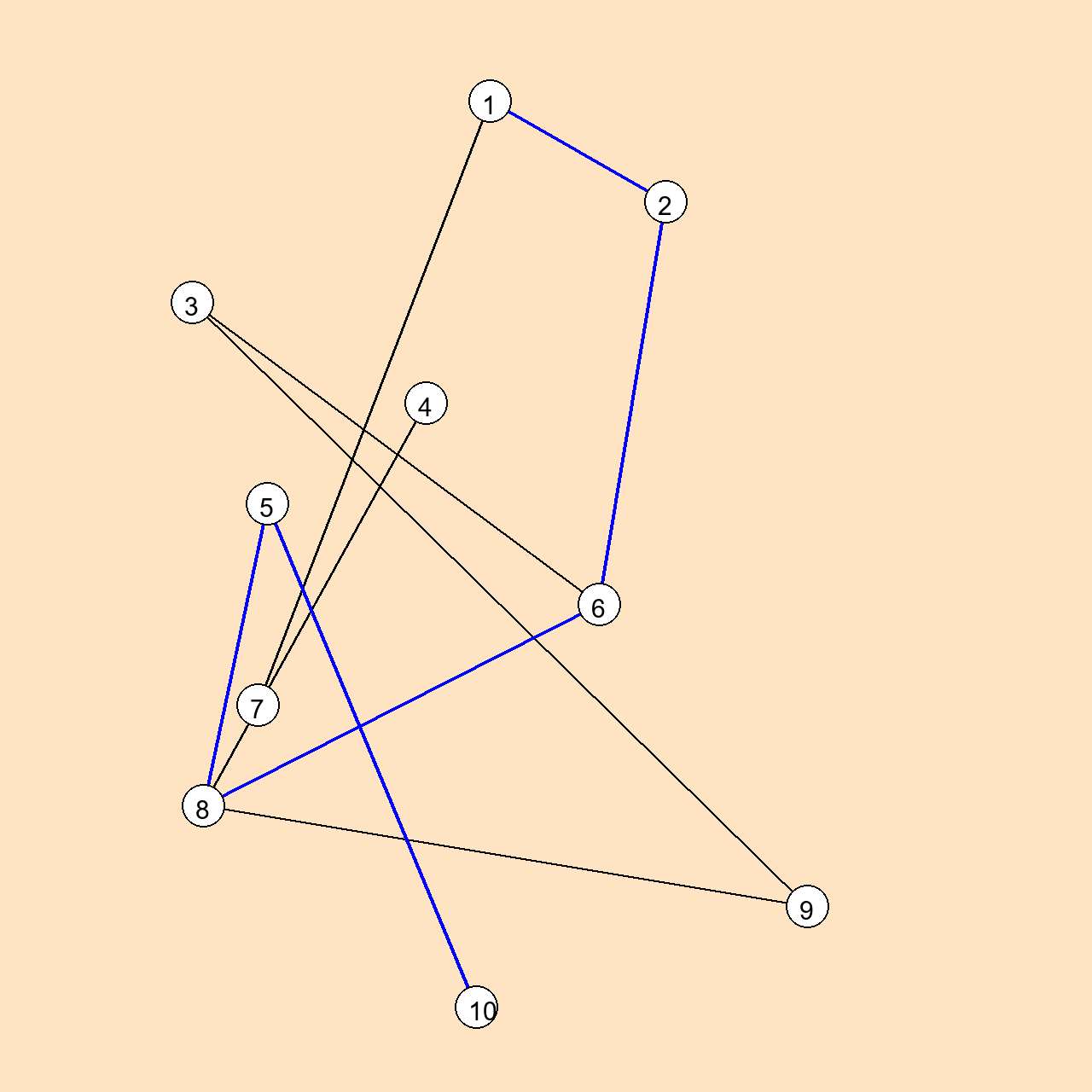


Рисунок 10.1 и 10.2. Результат работы программы и data.txt

## ВЫВОД

В ходе выполнения работы были получены практические навыки по реализации волноовго алгоритма для поиска кратчайшего пути в графе.

# Задание 6. Остовное дерево.Алгоритм Крускала

## ЦЕЛЬ РАБОТЫ

Реализовать алгоритм Крускала для создания остовного дерева из графа на языке программирования Python.

## ВЫПОЛНЕНИЕ РАБОТЫ

Листинг программы:

from PIL import Image, ImageDraw, ImageFont

from random import randint

videoDimensions = (1280, 1280)

img = Image.new("RGB", videoDimensions, *color* = '#FFE4C4')

imgDrawer = ImageDraw.Draw(img)

font = ImageFont.truetype('C:\Windows\Fonts\Arial.ttf', 30)

file = open('2 semestr/practice/Zadanie6/data.txt', 'r')

N = *int*(file.readline())

A = []

while True:

    tmp = file.readline().rsplit()

    tmp = [*int*(i) - 1 for i in tmp]

    if tmp == []:

        break

    A.append(tmp)

*def* new\_coord(*X*,*Y*):

    n = len(X)

    Y[0] = 10

    for i in range(n):

        X[i] = randint(50, videoDimensions[0] - 100)

        Y[i] = Y[i - 1] + videoDimensions[1] // n - 10

    return X, Y

*def* draw\_graph(*x*, *y*):

    for i in A:

        imgDrawer.line([(x[i[0]], y[i[0]]), (x[i[1]], y[i[1]])], 'black', 2)

        imgDrawer.text(((abs(x[i[0]] - x[i[1]]) // 2 + min(x[i[0]], x[i[1]]) + 5), (abs(y[i[0]] - y[i[1]]) // 2 + min(y[i[0]], y[i[1]]) + 5)), *str*(i[2]), (0, 0, 0), font)

        imgDrawer.ellipse([(x[i[0]] - 25, y[i[0]] - 25), (x[i[0]] + 25, y[i[0]] + 25)], 'white', 'black', 2)

        imgDrawer.ellipse([(x[i[1]] - 25, y[i[1]] - 25), (x[i[1]] + 25, y[i[1]] + 25)], 'white', 'black', 2)

        imgDrawer.text((x[i[0]] - 9, y[i[0]] - 12), *str*(i[0] + 1), (0, 0, 0), font)

        imgDrawer.text((x[i[1]] - 9, y[i[1]] - 12), *str*(i[1] + 1), (0, 0, 0), font)

Rs = sorted(A, *key*=*lambda* *x*: x[2])

U = *set*()

D = {}

T = []

*def* kr\_ostov():

    for r in Rs:

        if r[0] not in U or r[1] not in U:

            if r[0] not in U and r[1] not in U:

                D[r[0]] = [r[0], r[1]]

                D[r[1]] = D[r[0]]

            else:

                if not D.get(r[0]):

                    D[r[1]].append(r[0])

                    D[r[0]] = D[r[1]]

                else:

                    D[r[0]].append(r[1])

                    D[r[1]] = D[r[0]]

            T.append(r)

            U.add(r[0])

            U.add(r[1])

    for r in Rs:

        if r[1] not in D[r[0]]:

            T.append(r)

            gr0 = D[r[0]]

            D[r[0]] += D[r[1]]

            D[r[1]] += gr0

*def* draw\_ostov(*x*, *y*):

    for i in T:

        imgDrawer.line([(x[i[0]], y[i[0]]), (x[i[1]], y[i[1]])], 'red', 4)

        imgDrawer.ellipse([(x[i[0]] - 25, y[i[0]] - 25), (x[i[0]] + 25, y[i[0]] + 25)], 'white', 'black', 2)

        imgDrawer.ellipse([(x[i[1]] - 25, y[i[1]] - 25), (x[i[1]] + 25, y[i[1]] + 25)], 'white', 'black', 2)

        imgDrawer.text((x[i[0]] - 9, y[i[0]] - 12), *str*(i[0] + 1), (0, 0, 0), font)

        imgDrawer.text((x[i[1]] - 9, y[i[1]] - 12), *str*(i[1] + 1), (0, 0, 0), font)

X = [0]\*N

Y = [0]\*N

new\_coord(X, Y)

draw\_graph(X, Y)

kr\_ostov()

draw\_ostov(X, Y)

print(T)

img.show()

Скриншот работы программы и данные из файла data.txt представлены на Рис.11.1 и 11.2.

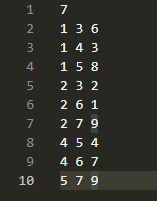
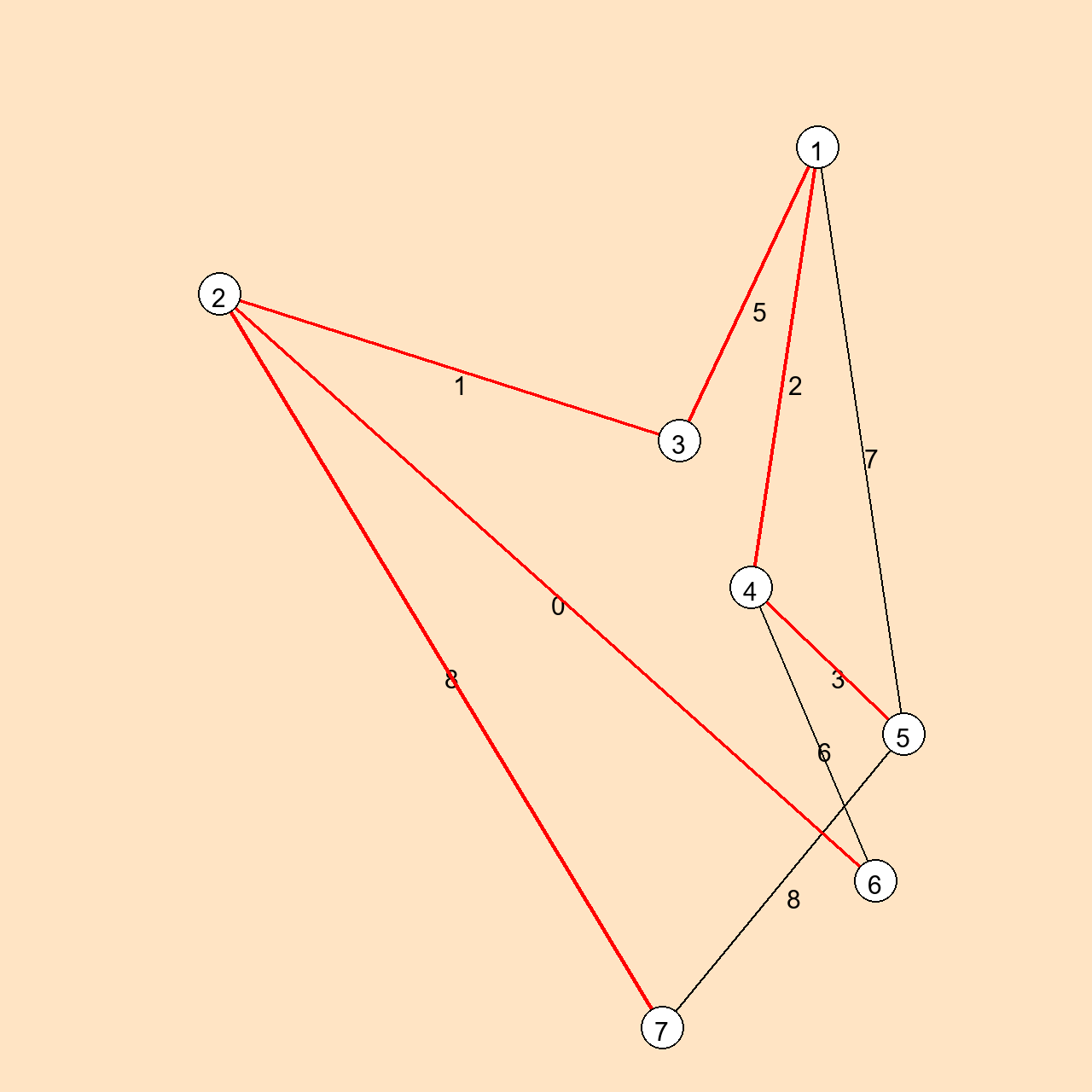


Рисунок 11.1 и 11.2. Результат работы программы и data.txt

## ВЫВОД

В ходе выполнения работы были получены практические навыки по реализации алгоритма Крусала для создания остовного дерева из графа.

# Задание 7. Циклы

## ЦЕЛЬ РАБОТЫ

Реализовать алгоритм поиска циклов в графе на языке программирования Python.

## ВЫПОЛНЕНИЕ РАБОТЫ

Листинг программы:

from PIL import Image, ImageDraw, ImageFont

from random import randint

videoDimensions = (1280, 1280)

img = Image.new("RGB", videoDimensions, color = 'white')

imgDrawer = ImageDraw.Draw(img)

font = ImageFont.truetype('C:\Windows\Fonts\Arial.ttf', 30)

file = open('2 semestr/practice/Zadanie7/data.txt', 'r')

N = int(file.readline())

A = {}

for i in range(N): A[i] = []

while True:

    tmp = file.readline().rsplit()

    if tmp == []:

        break

    tmp[0], tmp[1] = int(tmp[0]) - 1, int(tmp[1]) - 1

    A[tmp[0]].append(tmp[1])

    A[tmp[1]].append(tmp[0])

def new\_coord():

    Y[0] = 10

    for i in range(N):

        X[i] = randint(80, videoDimensions[0] - 100)

        Y[i] = Y[i - 1] + videoDimensions[1] // N - 10

def draw\_graph(x, y):

    for i in range(N):

        for j in A[i]:

            imgDrawer.line([(x[i], y[i]), (x[j], y[j])], 'black', 2)

    for i in range(N):

        imgDrawer.ellipse([(x[i] - 25, y[i] - 25), (x[i] + 25, y[i] + 25)], 'white', 'black', 2)

        imgDrawer.text((x[i] - 9, y[i] - 12), str(i + 1), (0, 0, 0), font)

def dfs(graph, start, end):

    fringe = [(start, [])]

    while fringe:

        state, path = fringe.pop()

        if path and state == end:

            yield path

            continue

        for next\_state in graph[state]:

            if next\_state in path:

                continue

            fringe.append((next\_state, path+[next\_state]))

X = [0]\*N

Y = [0]\*N

new\_coord()

draw\_graph(X, Y)

cycle = []

for node in A:

    for path in dfs(A, node, node):

        point = False

        t = sorted(path)

        for i in cycle:

            if sorted(i) == t or len(t) < 3: point = True

        if not point:

            cycle.append(t)

for i in range(len(cycle)):

    for j in range(len(cycle[i])):

        imgDrawer.text((20 + 30 \* j, 20 + 30 \* i), str(cycle[i][j] + 1), (0, 0, 0), font)

print(cycle)

img.show()

Скриншот работы программы и данные из файла data.txt представлены на Рис.12.1 и 12.2.

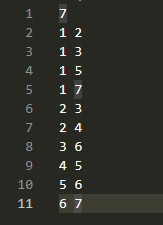
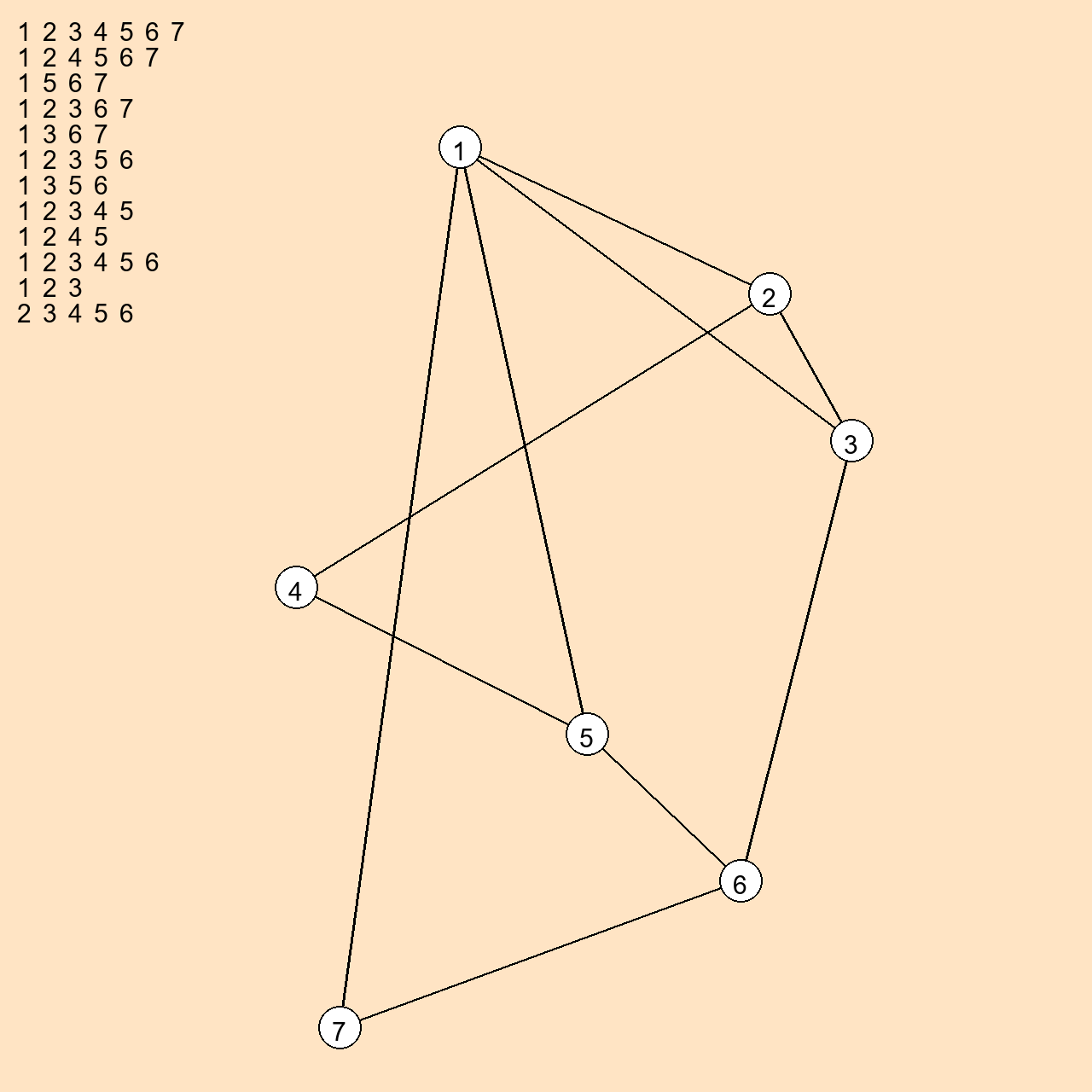


Рисунок 12.1 и 12.2. Результат работы программы и data.txt

## ВЫВОД

В результате выполнения работы были получены практические навыки по реализации алгоритма поиска циклов в графе.

# Задание 8. Гамильтонов цикл

## ЦЕЛЬ РАБОТЫ

Реализовать Гамильтонов цикл в графе на языке программирования Python.

## ВЫПОЛНЕНИЕ РАБОТЫ

Листинг программы:

from PIL import Image, ImageDraw, ImageFont

from random import randint

videoDimensions = (1280, 1280)

img = Image.new('RGB', videoDimensions, *color* = '#FFE4C4')

imgDrawer = ImageDraw.Draw(img)

font = ImageFont.truetype('C:\Windows\Fonts\Arial.ttf', 30)

file = open('2 semestr/practice/Zadanie8/data.txt', 'r')

N = *int*(file.readline())

A = [0] \* N

for i in range(N):

    A[i] = [0] \* N

while True:

    tmp = file.readline().rsplit()

    if tmp == []:

        break

    tmp[0], tmp[1] = *int*(tmp[0]) - 1, *int*(tmp[1]) - 1

    A[tmp[0]][tmp[1]] = 1

    A[tmp[1]][tmp[0]] = 1

*def* new\_coord():

    Y[0] = 10

    for i in range(N):

        X[i] = randint(80, videoDimensions[0] - 100)

        Y[i] = Y[i - 1] + videoDimensions[1] // N - 10

*def* draw\_graph(*x*, *y*):

    for i in range(N):

        for j in range(i + 1, N):

            if A[i][j] == 1:

                imgDrawer.line([(x[i], y[i]), (x[j], y[j])], 'black', 2)

    for i in range(1, N):

        imgDrawer.ellipse([(x[i] - 25, y[i] - 25), (x[i] + 25, y[i] + 25)], 'white', 'black', 2)

        imgDrawer.text((x[i] - 9, y[i] - 12), *str*(i), (0, 0, 0), font)

*def* draw\_hamilton():

    for i in range(len(path) - 1):

        imgDrawer.line([(X[path[i]], Y[path[i]]), (X[path[i + 1]], Y[path[i + 1]])], 'red', 4)

    for i in path:

        imgDrawer.ellipse([(X[i] - 25, Y[i] - 25), (X[i] + 25, Y[i] + 25)], 'white', 'black', 2)

        imgDrawer.text((X[i] - 9, Y[i] - 12), *str*(i), (0, 0, 0), font)

*def* GamCycle(*gg*):

    path.append(gg)

    if len(path) == N:

        if A[path[0]][path[-1]] == 1:

            return True

        else:

            path.pop()

            return False

    visited[gg] = True

    for next in range(N):

        if A[gg][next] == 1 and not visited[next]:

            if GamCycle(next):

                return True

    visited[gg] = False

    path.pop()

    return False

X = [0]\*N

Y = [0]\*N

visited = [False] \* N

path = []

new\_coord()

draw\_graph(X, Y)

GamCycle(0)

draw\_hamilton()

imgDrawer.text((100, 1200), 'Путь: ', (0,0,0), font)

imgDrawer.text((200, 1200), *str*(path), (0,0,0), font)

img.show()

Скриншот работы программы и данные из файла data.txt представлены на Рис.13.1 и 13.2.

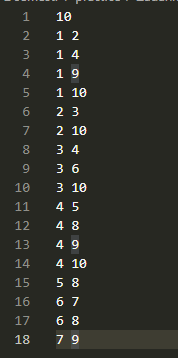
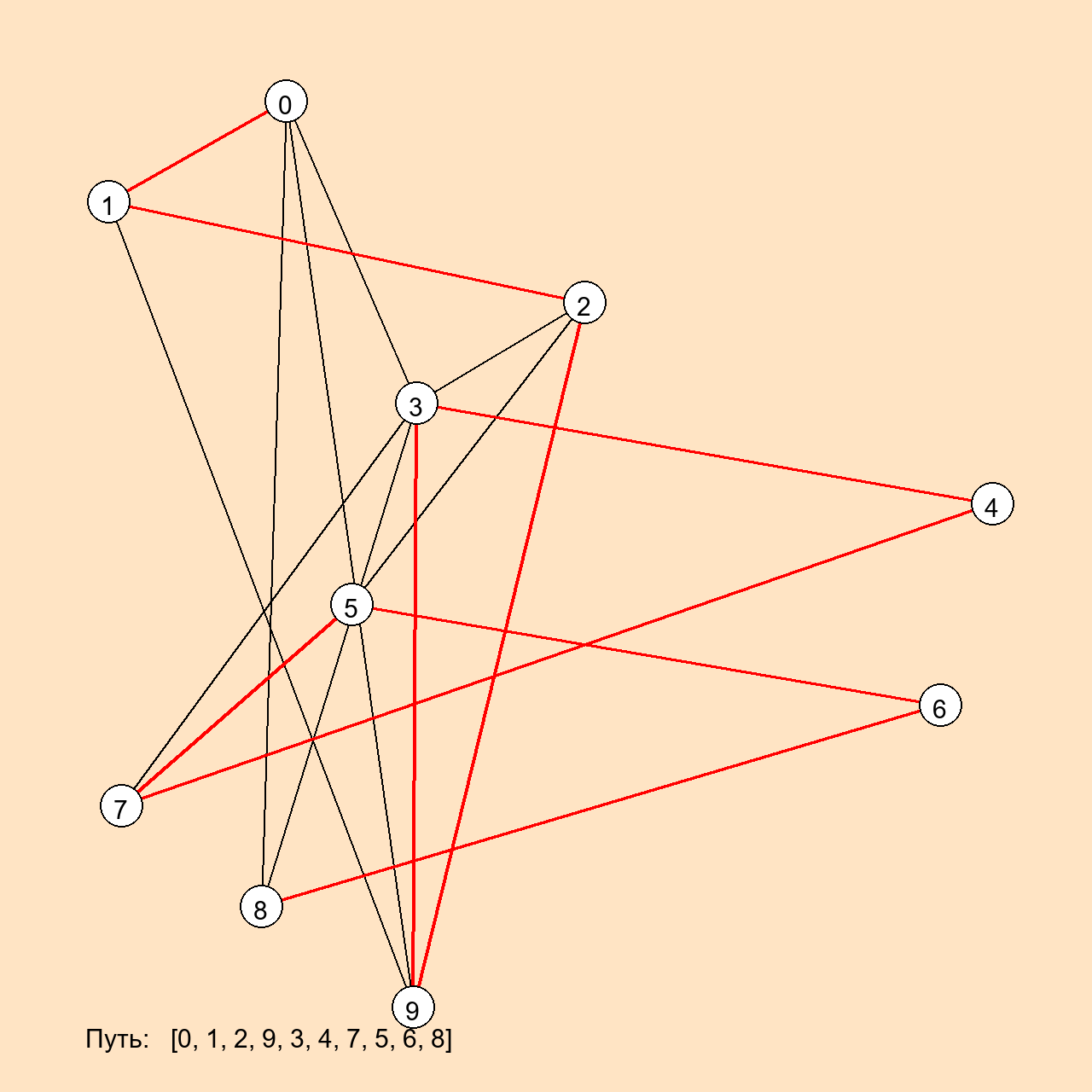


Рисунок 13.1 и 13.2. Результат работы программы и data.txt

## ВЫВОД

В ходе выполнения работы были получены практические навыки по реализации алгоритма Гамильтонова цикла.

# Задание 9. Задание по выбору

## ЦЕЛЬ РАБОТЫ

Реализовать алгоритм обработки графов на языке программирования Python

## ВЫПОЛНЕНИЕ РАБОТЫ

Определить все мосты графа(Рис. 14)

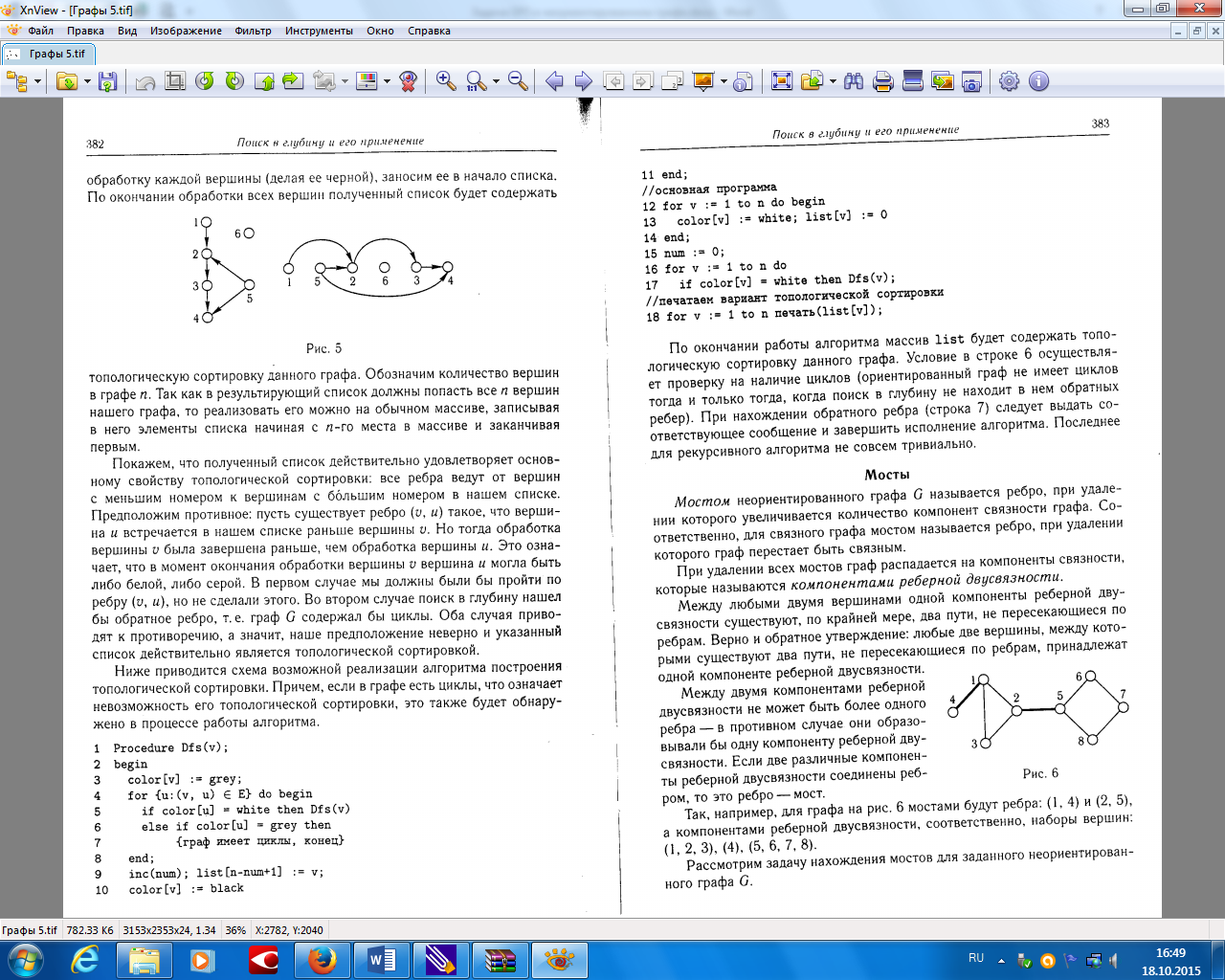


Рисунок 14. Граф

Листинг программы:

from collections import defaultdict

class Graph:

    def \_\_init\_\_(self,vertices):

        self.V= vertices

        self.graph = defaultdict(list)

        self.Time = 0

    def addEdge(self,u,v):

        self.graph[u].append(v)

        self.graph[v].append(u)

    def bridgeUtil(self,u, visited, parent, low, disc):

        visited[u]= True

        disc[u] = self.Time

        low[u] = self.Time

        self.Time += 1

        for v in self.graph[u]:

            if visited[v] == False :

                parent[v] = u

                self.bridgeUtil(v, visited, parent, low, disc)

                low[u] = min(low[u], low[v])

                if low[v] > disc[u]:

                    print ("%d %d" %(u,v))

            elif v != parent[u]:

                low[u] = min(low[u], disc[v])

    def bridge(self):

        visited = [False] \* (self.V)

        disc = [float("Inf")] \* (self.V)

        low = [float("Inf")] \* (self.V)

        parent = [-1] \* (self.V)

        for i in range(self.V):

            if visited[i] == False:

                self.bridgeUtil(i, visited, parent, low, disc)

graf = Graph(8)

graf.addEdge(0, 1)

graf.addEdge(0, 2)

graf.addEdge(0, 3)

graf.addEdge(1, 2)

graf.addEdge(1, 4)

graf.addEdge(4, 5)

graf.addEdge(4, 7)

graf.addEdge(5, 6)

graf.addEdge(7, 6)

print ("Bridges in graph ")

graf.bridge()

Скриншот работы программы представлен на Рис.15

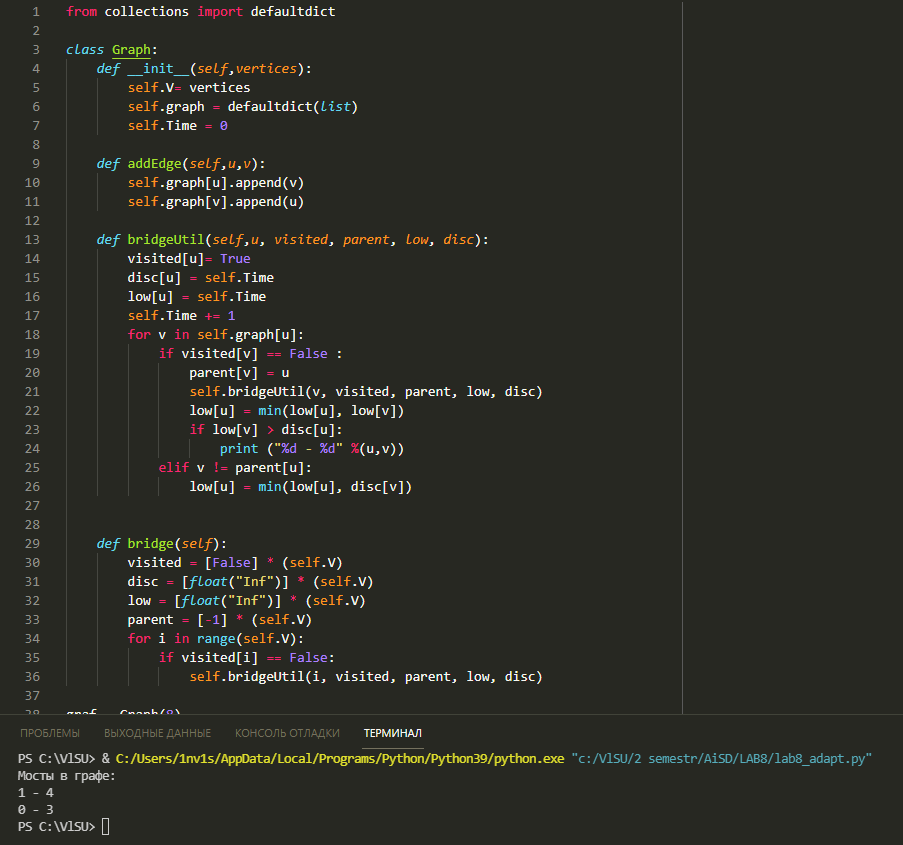


Рисунок 15. Скриншот работы программы

## ВЫВОД

В ходе выполнения работы были получены практические навыки по разработке алгоритма обработки графа.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения практики была закреплена и углублена теоретическую подготовку, расширен профессиональный и общий кругозор, получены первичные профессиональные умения и навыки практического использования вычислительной техники, были решены следующие задачи:

* изучение специальной литературы и другой научно-технической информации, достижений отечественной и зарубежной науки и техники в области программной инженерии;
* изучение и освоение комплекса технических и программных средств базы практики;
* выполнение практических заданий по углублённым темам дисциплин первого года обучения;
* участие в эксплуатации, сопровождении программно-информационных систем;
* оформление результатов анализа информации по заданной теме и собственных разработок в виде отчета.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Королев Л.Н. Информатика. Введение в компьютерные науки: Учебник [Электронный ресурс] / Л.Н. Королев, А.И. Миков. - М.: Абрис, 2012.- 367 с.: ил. - ISBN 978-5- 4372-0042-1. <https://www.studentlibrary.ru/book/ISBN9785437200421.html>
2. Информационные технологии в профессиональной деятельности : учеб. пособие. [Электронный ресурс] - Москва : Проспект, 2014. - 448 с. - ISBN 978-5-392-12318-6. <http://www.studentlibrary.ru/book/ISBN9785392123186.html>
3. Практикум по информационным технологиям в профессиональной деятельности: учеб. пособие. [Электронный ресурс] - Москва : Проспект, 2015. - 288 с. - ISBN 978-5-392-16901-6. <http://www.studentlibrary.ru/book/ISBN9785392169016.html>
4. Основы программирования [Электронный ресурс] / С. М. Окулов. - 6-е изд., перераб. (эл.). - М. : БИНОМ. Лаборатория знаний, 2012.- 336 с. : ил. - (Развитие интеллекта школьников). - ISBN 978-5-9963-1094-4. <http://www.studentlibrary.ru/book/ISBN9785392169016.html>
5. Задачи по программированию [Электрон-ный ресурс] / С. М. Окулов [и др.] ; под ред. С. М. Окулова. - 2-е изд., испр. (эл.).-М. : БИНОМ. Лаборатория знаний, 2014. - 823 с. : ил. - ISBN 978-5-9963-2372-2. <http://www.studentlibrary.ru/book/ISBN9785392169016.html>
6. Информационные технологии [Электронный ресурс] / Е.В. Парфенова - М. : МИСиС, 2018. <http://www.studentlibrary.ru/book/misis_0020.html>
7. [http://www.edu.ru/](http://www.edu.ru/%20) – Федеральный портал «Российское образование»
8. [http://window.edu.ru/](http://window.edu.ru/%20) – Единое окно доступа к образовательным ресурсам
9. [http://library.vlsu.ru/](http://library.vlsu.ru/%20) – научная библиотека ВлГУ
10. <https://vlsu.bibliotech.ru> – электронно-библиотечная система ВлГУ
11. <http://www.studentlibrary.ru/> – электронно-библиотечная система «Консультант Студента»
12. [http://elibrary.ru/](http://elibrary.ru/%20) – научная электронная библиотека