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

**Практическая работа № 4**

**«Графы. Мосты»**

Выполнил:

ст. гр. ПРИ-120

К. А. Борисова

Принял:

Преподаватель

О. Н. Шамышева

Владимир, 2020

**Цель работы**

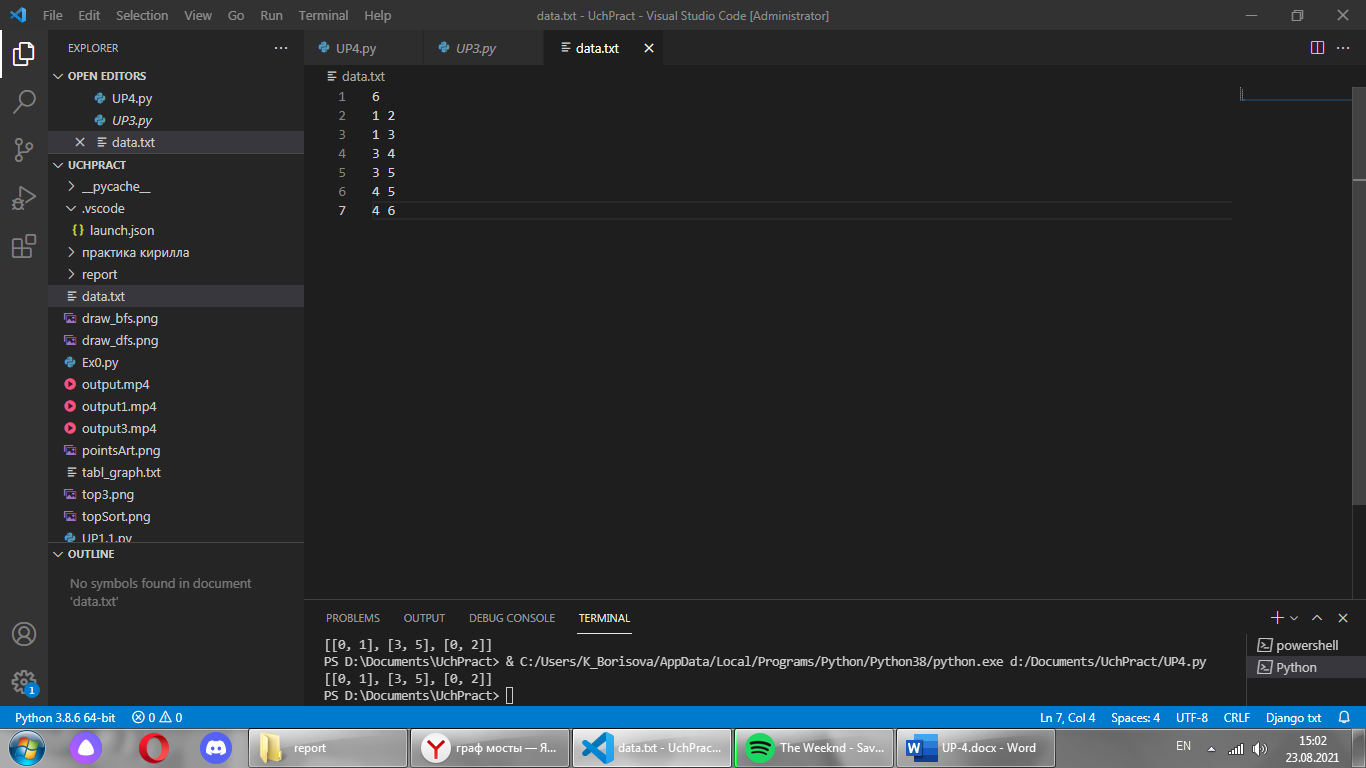
Научиться использовать алгоритм DFS для поиска мостов в графе.

**Ход работы**

**Мост** — ребро в теории графов, удаление которого увеличивает число компонент связности.

Алгоритм получает на вход граф (информация считывается с файла). В результате работы программы будут найдены и выделены все мосты в графе. Результат визуализирован на графе.

Листинг программы и информация из файла представлены ниже:

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)

with open('data.txt', 'r') as file:

    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 + 1), (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] + 1), (0, 0, 0), ImageFont.truetype('C:\Windows\Fonts\Arial.ttf', 30))

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

imgDrawer.text((100, 50), str(bridges), (0,0,0), font)

img.show()

Результат работы программы:

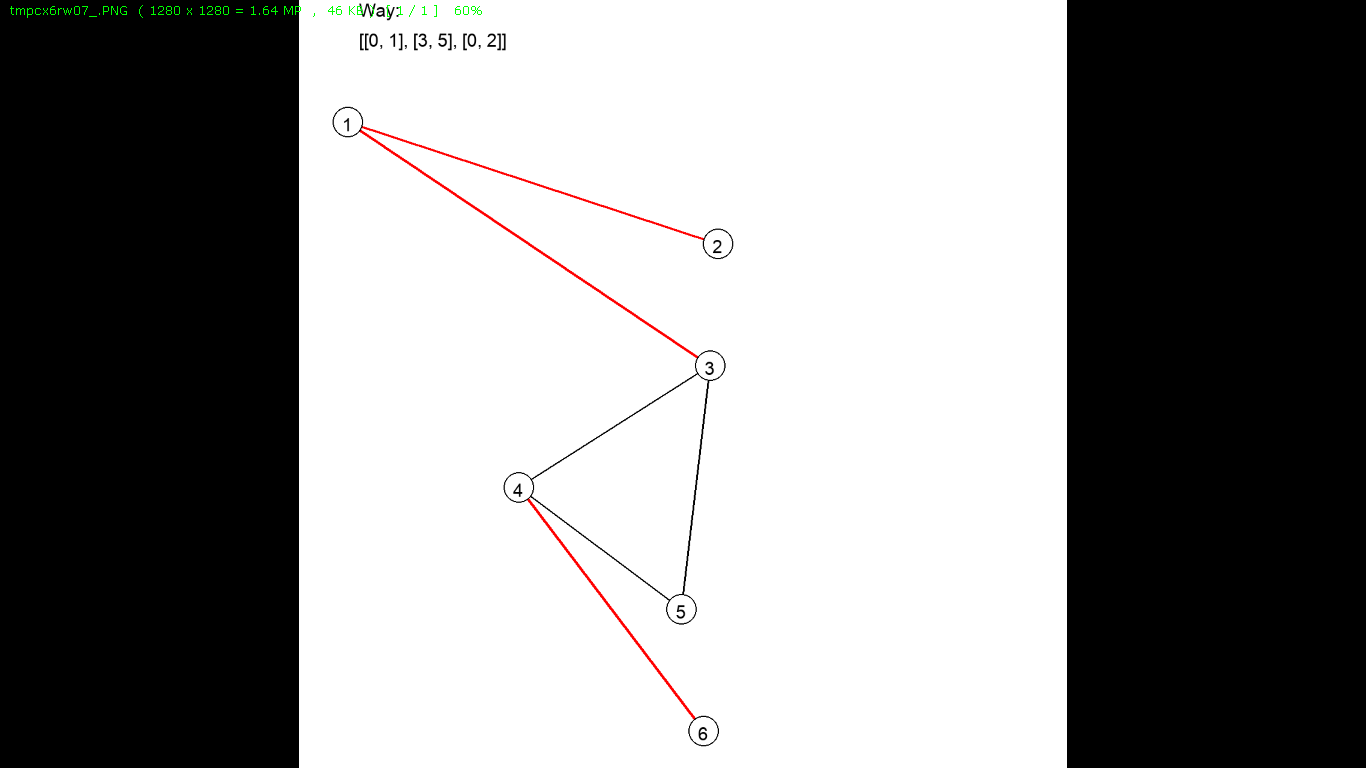


Рисунок . Скриншот результата работы программы

**Вывод**

В ходе данной работы была изучена тема «Мосты» в графах, реализован алгоритм поиска мостов в графе.