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 № 7**

**«Неориентированный невзвешенный граф. Циклы»**

Выполнил:

ст. гр. ПРИ-120

К. А. Борисова

Принял:

Преподаватель

О. Н. Шамышева

Владимир, 2020

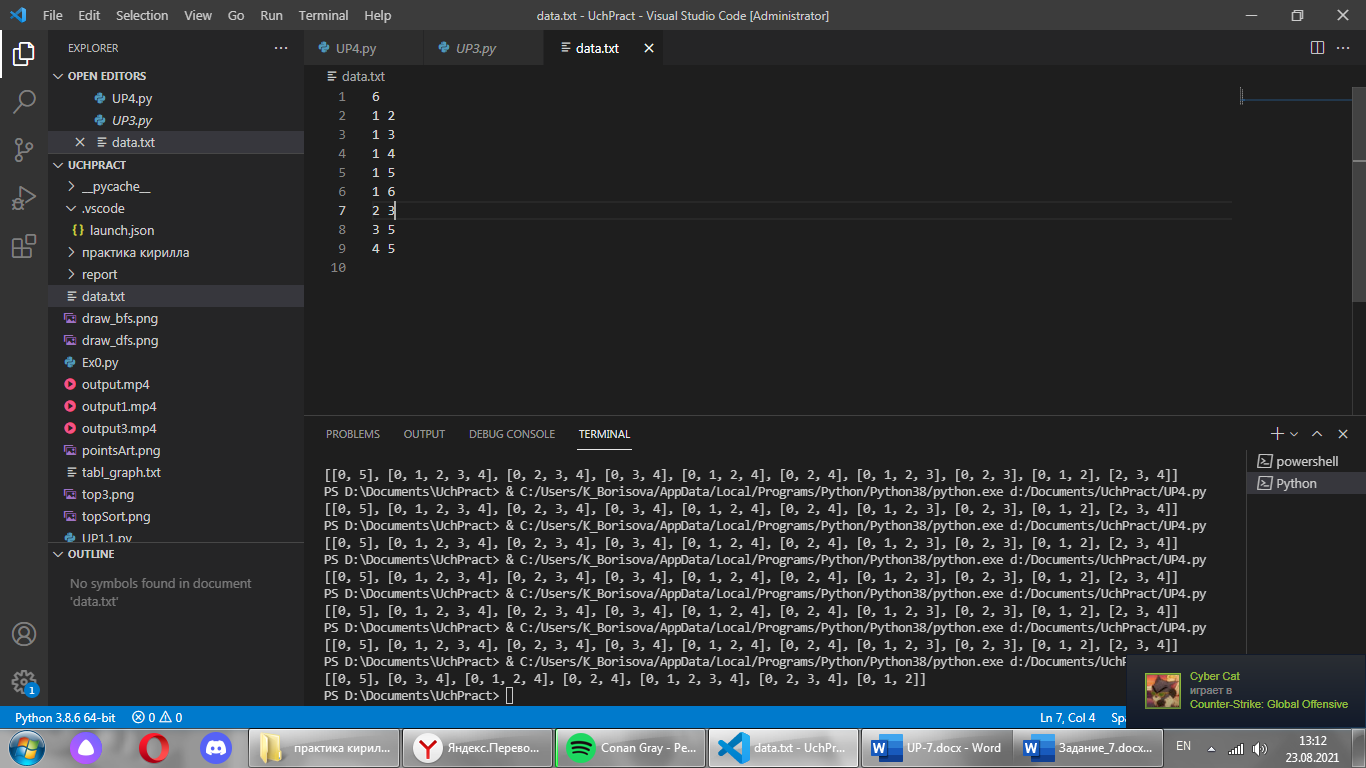
**Цель работы**

Изучить реализацию алгоритма поиска циклов в невзвешенном неориентированном графе.

**Ход работы**

Алгоритм получает на вход граф (информация считывается с файла). В результате работы программы из исходного графа выделяются циклы с наименьшей вершины к наибольшей. 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[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()

Результат работы программы:

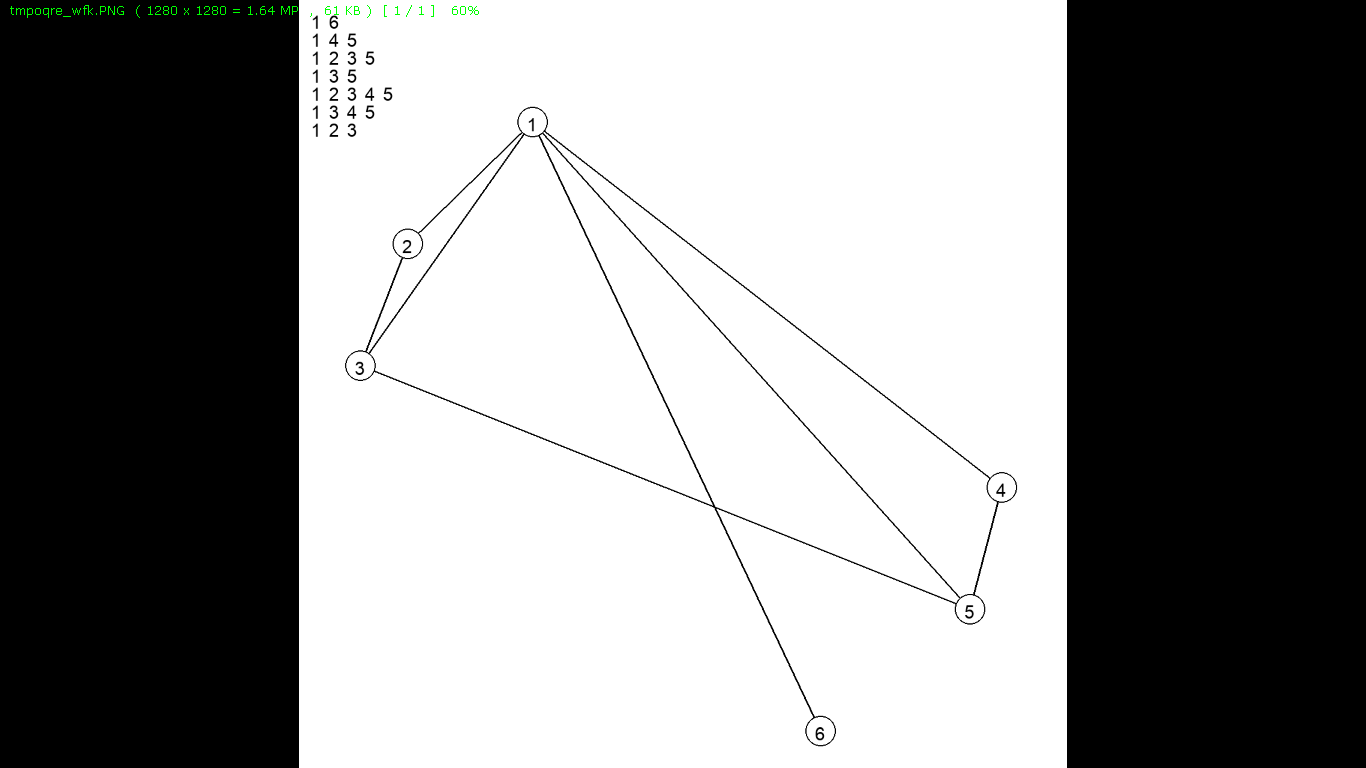


Рисунок 1. Скриншот работы программы

**Вывод**

В ходе работы был изучен и реализован поиск циклов в невзвешенном неориентированном графе.