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 № 6**

**«Остовное дерево. Алгоритм Крускала»**

Выполнил:

ст. гр. ПРИ-120

К. А. Борисова

Принял:

Преподаватель

О. Н. Шамышева

Владимир, 2020

**Цель работы**

Изучить алгоритм Крускала для создания остовного дерева из взвешенного неориентированного графа.

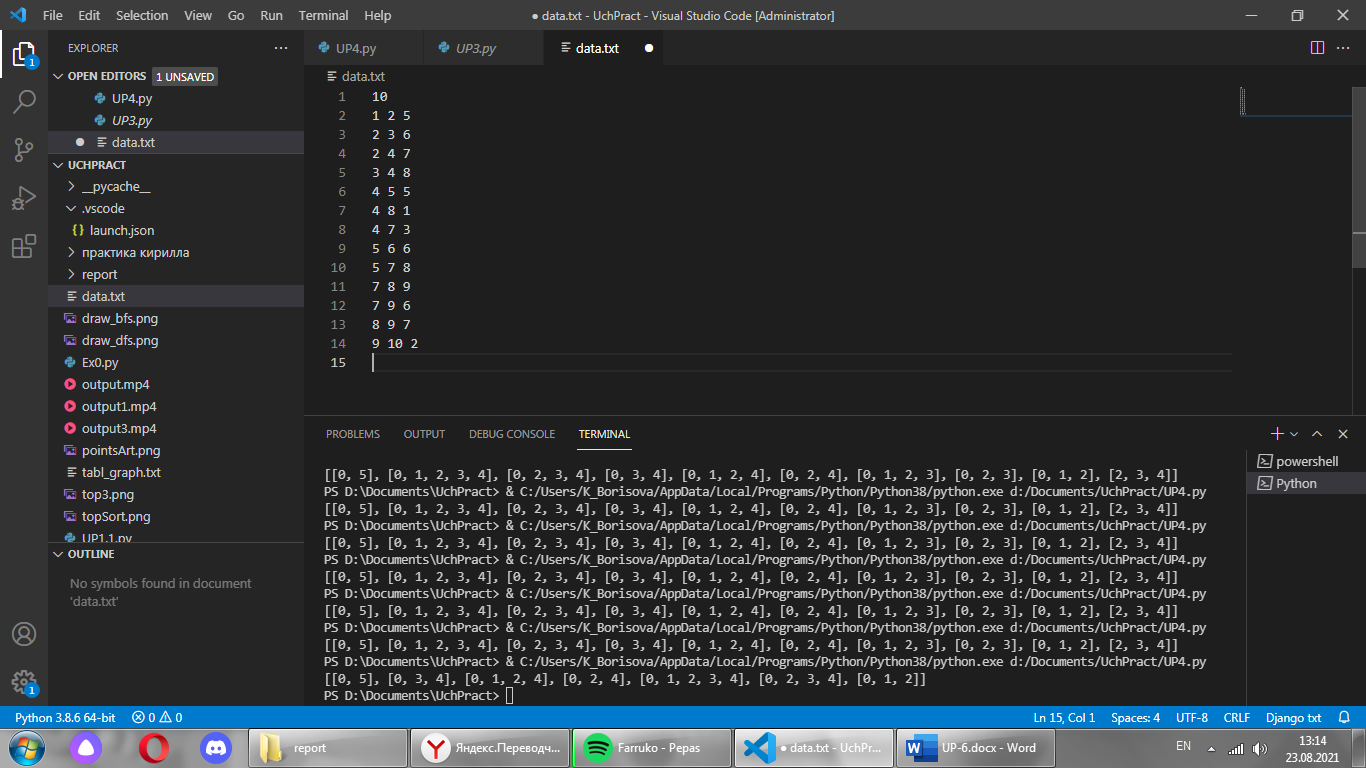
**Ход работы**

**Алгоритм Крускала** достаточно прост в своей идее и реализации. Он заключается в сортировке всех рёбер в порядке возрастания длины, и поочерёдному добавлению их в минимальный остов, если они соединяют различные компоненты связности.

Более формально: пусть мы уже нашли некоторые рёбра, входящие в минимальный остов. Утверждается, что среди всех рёбер, соединяющих различные компоненты связности, в минимальный остов будет входить ребро с минимальной длиной.

Алгоритм получает на вход граф в виде матрицы весов (информация считывается с файла). В результате работы программы из исходного графа, не являющегося остовным деревом, строится остовное дерево с минимальным весом ребер. 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 = []

    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()

Результат работы программы:

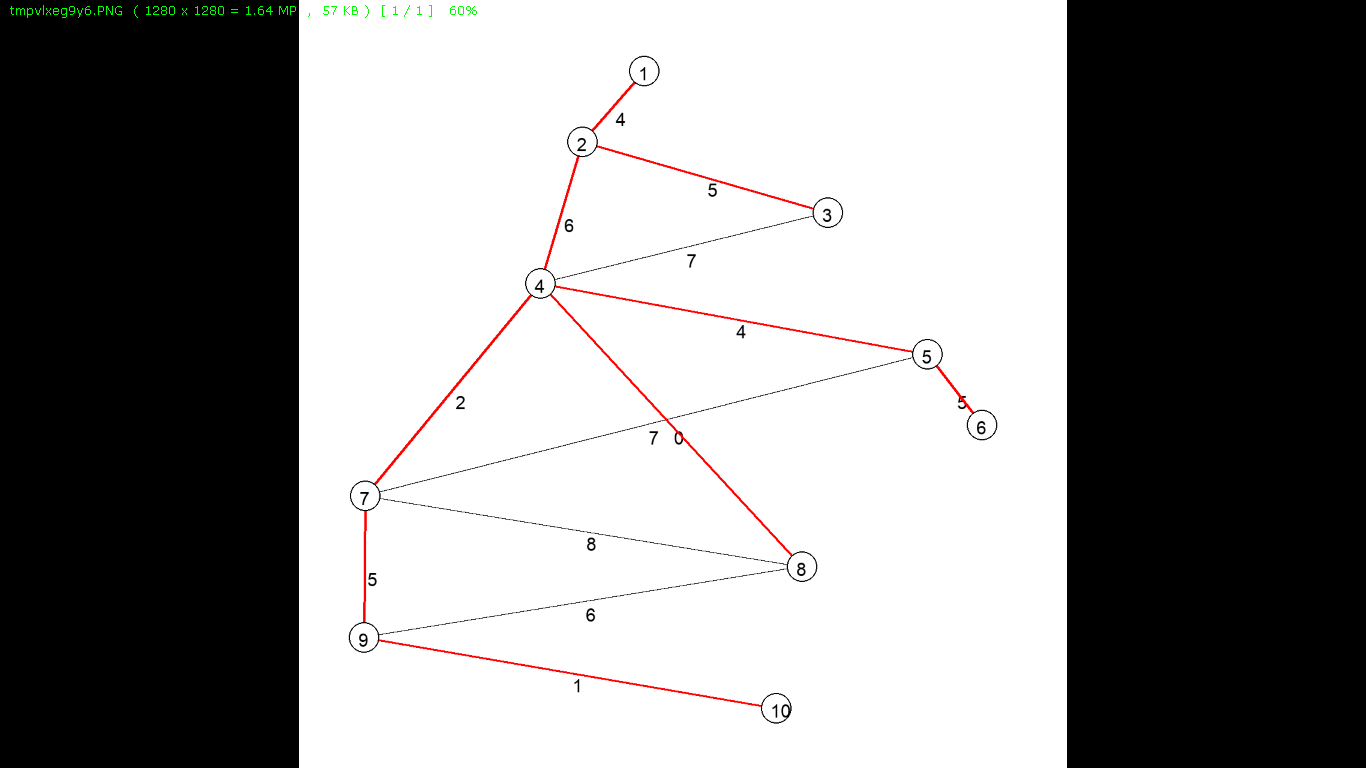


Рисунок . Скриншот работы программы

**Вывод**

В ходе работы был изучен и применен алгоритм Крускала для построения остовного дерева с минимальным весом ребер.