МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»

(БГТУ им. В.Г.Шухова)

**Лабораторная работа № 3**

Дисциплина: Теория надежности

Выполнил: студент группы ВТ-31

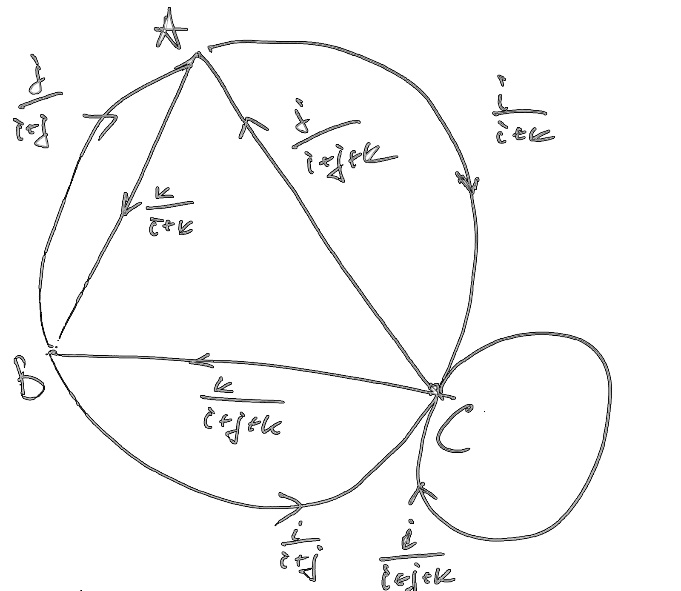
Ковалёв И. Д.

Проверил:

Кабалянц П. С

Белгород 2020

**Задание к работе:** смоделировать эргодическую марковскую цепь, описанную следующим графом



Составим матрицу вероятностей переходов Р по имеющемуся графу переходов и заданным вероятностям:

Решим уравнение , преобразовав его к виду . В результате получим:

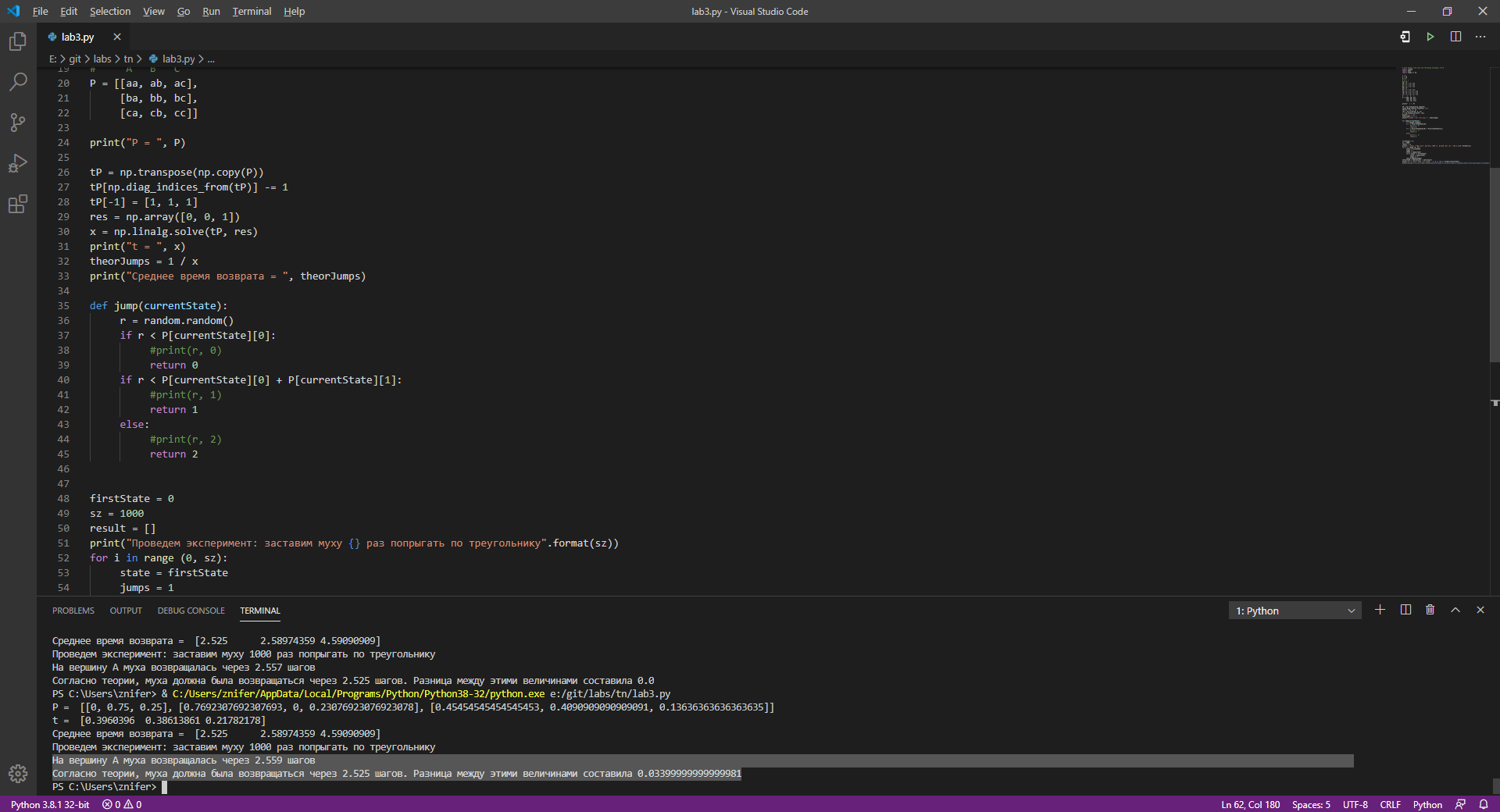
t = [0.396 0.386 0.218]

Для того, чтобы получить среднее время возврата в начальную позицию , необходимо найти обратную величину от . В нашем случае: = [2.525 2.589 4.591], т. е. среднее время возвращения в точку А, если начали в ней же равняется 2.525 ед. времени.

Смоделируем полет мухи со стартовой позицией в точке А, и повторим его 1000 раз.

В результате моделирования было установлено, что муха возвращалась через 2.559 шагов, что на 0.034 больше, чем значение, вычисленное теоретически.

**Скриншот разработанной программы:**



**Листинг программы:**

# This Python file uses the following encoding: utf-8

import random

import math

import numpy as np

i = 3

j = 10

k = 9

# из / в

aa = 0

ab = k / (i + k)

ac = i / (i + k)

ba = j / (i + j)

bb = 0

bc = i / (i + j)

ca = j / (i + j + k)

cb = k / (i + j + k)

cc = i / (i + j + k)

#     A   B   C

P = [[aa, ab, ac],

     [ba, bb, bc],

     [ca, cb, cc]]

print("Р = ", P)

tP = np.transpose(np.copy(P))

tP[np.diag\_indices\_from(tP)] -= 1

tP[-1] = [1, 1, 1]

res = np.array([0, 0, 1])

x = np.linalg.solve(tP, res)

print("t = ", x)

theorJumps = 1 / x

print("Среднее время возврата = ", theorJumps)

def jump(currentState):

     r = random.random()

     if r < P[currentState][0]:

          #print(r, 0)

          return 0

     if r < P[currentState][0] + P[currentState][1]:

          #print(r, 1)

          return 1

     else:

          #print(r, 2)

          return 2

firstState = 0

sz = 1000

result = []

print("Проведем эксперимент: заставим муху {} раз попрыгать по треугольнику".format(sz))

for i in range (0, sz):

     state = firstState

     jumps = 1

     state = jump(state)

     while state != firstState:

          state = jump(state)

          jumps += 1

     result.append(jumps)

returnTime = sum(result) / len(result)

print("На вершину А муха возвращалась через {} шагов".format(returnTime))

print("Согласно теории, муха должна была возвращаться через {} шагов. Разница между этими величинами составила {}".format(round(theorJumps[0], 4), abs(returnTime - theorJumps[0])))