Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №7

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний во взвешенном графе»

Выполнил:

студент группы 20ВВ2

Барсуков Н.И.

Проверили:

Митрохин М.А.

Юрова О.В.

Пенза 2021

**Задание 1:**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного графа G. Выведите матрицу на экран.

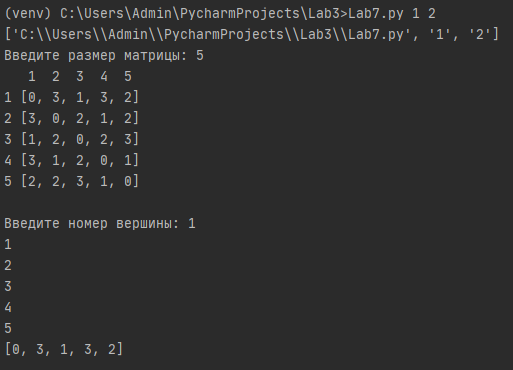
2. Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием.

3. \* Сгенерируйте (используя генератор случайных чисел) матрицу смежности для ориентированного взвешенного графа G. Выведите матрицу на экран и осуществите процедуру поиска расстояний.

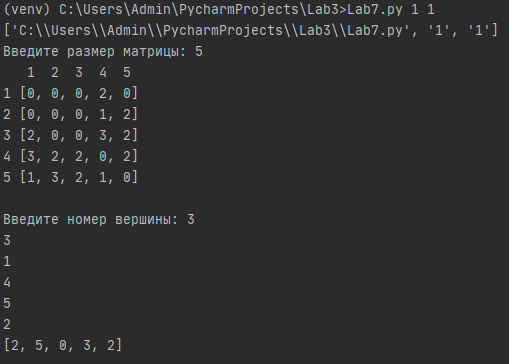
**Задание 2\*:**

1. Модернизируйте программу так, чтобы получить возможность запуска программы с параметрами командной строки. В качестве параметра должны указываться тип графа и наличие ориентации его ребер.

Взвешенный неориентированный граф.



Взвешенный ориентированный граф



**Листинг**

import Graph  
import Operations  
import sys  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print(sys.argv)  
 n = int(input("Введите размер матрицы: "))  
 if int(sys.argv[1]) == 1: # Взвешенный  
 if int(sys.argv[2]) == 1: # Ориентированный  
 graph = []  
 Graph.generator\_w\_orientation\_matrix(graph, n, 3)  
 Graph.print\_matrix(graph, 0, n)  
  
 Dist = [1000 for \_ in range(n)]  
 node = int(input("Введите номер вершины: "))  
 Operations.bfsd\_matrix\_real(graph, Dist, node - 1)  
 print()  
  
 elif int(sys.argv[2]) == 2: # Неориентированный  
 graph = []  
 Graph.generator\_w\_matrix(graph, n, 3)  
 Graph.print\_matrix(graph, 0, n)  
  
 Dist = [1000 for \_ in range(n)]  
 node = int(input("Введите номер вершины: "))  
 Operations.bfsd\_matrix\_real(graph, Dist, node - 1)  
 print()  
  
 elif int(sys.argv[1]) == 2: # Невзвешенный  
 if int(sys.argv[2]) == 1:  
 graph = []  
 Graph.generator\_w\_orientation\_matrix(graph, n, 1)  
 Graph.print\_matrix(graph, 0, n)  
  
 Dist = [1000 for \_ in range(n)]  
 node = int(input("Введите номер вершины: "))  
 Operations.bfsd\_matrix\_real(graph, Dist, node - 1)  
 print()  
  
 elif int(sys.argv[2]) == 2:  
 graph = []  
 Graph.generator\_w\_matrix(graph, n, 1)  
 Graph.print\_matrix(graph, 0, n)  
  
 Dist = [1000 for \_ in range(n)]  
 node = int(input("Введите номер вершины: "))  
 Operations.bfsd\_matrix\_real(graph, Dist, node - 1)  
 print()

**Функция generator\_w\_orientation\_matrix**

def generator\_w\_orientation\_matrix(graph, n, maxNumber):  
 for i in range(n):  
 a = [0 for i in range(n)]  
 graph.append(a)  
  
 for i in range(n):  
 for j in range(n):  
 if (graph[i][j] == 0) and (i != j):  
 graph[i][j] = random.randint(0, maxNumber)

**Функция generator\_w\_matrix**

def generator\_w\_matrix(graph, n, maxNumber):  
 for i in range(n):  
 a = [0 for i in range(n)]  
 graph.append(a)  
  
 for i in range(n):  
 for j in range(n):  
 if (graph[i][j] == 0) and (i != j):  
 graph[i][j] = random.randint(0, maxNumber)  
 graph[j][i] = graph[i][j]

**Функция bfsd\_matrix\_real**

def bfsd\_matrix\_real(graph, dist, node):  
 queue = Queue()  
 queue.add(node)  
 while queue.len != 0:  
 item = queue.pop()  
 dist[node] = 0  
 print(item+1)  
 for i in range(len(graph)):  
 if graph[item][i] != 0 and dist[i] > dist[item] + graph[item][i]:  
 queue.add(i)  
 dist[i] = dist[item] + graph[item][i]  
 print(dist)