Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний в графе»

Выполнил:

студент группы 20ВВ2

Барсуков Н.И.

Проверили:

Митрохин М.А.

Юрова О.В.

Пенза 2021

**Задание 1:**

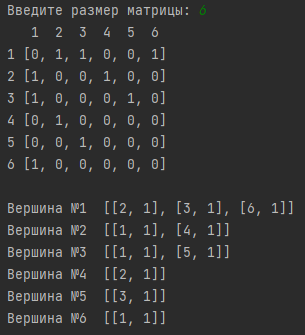
1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа G. Выведите матрицу на экран.

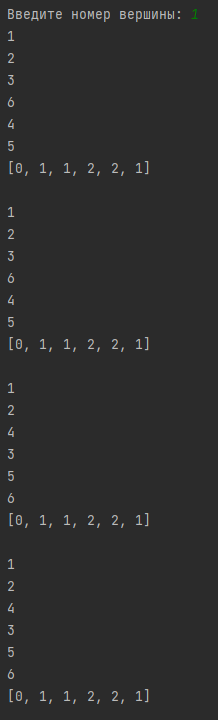
2. Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием.

3. \*Реализуйте процедуру поиска расстояний для графа, представленного списками смежности.

**Задание 2\*:**

1. Реализуйте процедуру поиска расстояний на основе обхода в глубину.
2. Реализуйте процедуру поиска расстояний на основе обхода в глубину для графа, представленного списками смежности.





**Листинг**

import Graph  
import Operations  
  
n = int(input("Введите размер матрицы: "))  
  
# graph = []  
graph = [[0, 1, 1, 0, 0, 1],  
 [1, 0, 0, 1, 0, 0],  
 [1, 0, 0, 0, 1, 0],  
 [0, 1, 0, 0, 0, 0],  
 [0, 0, 1, 0, 0, 0],  
 [1, 0, 0, 0, 0, 0]]  
line = []  
# Graph.generator\_matrix(graph, n)  
Graph.print\_matrix(graph, 0, n)  
Graph.generator\_line(line, graph, n)  
  
Dist = [-1 for \_ in range(n)]  
  
node = int(input("Введите номер вершины: "))  
Operations.bfsd\_matrix(graph, Dist, node-1)  
print()  
  
Dist = [-1 for \_ in range(n)]  
Operations.bfsd\_line(line, Dist, node-1)  
print()  
  
Dist = [-1 for \_ in range(n)]  
Operations.dfsd\_matrix(graph, Dist, node-1)  
print(Dist)  
print()  
  
Dist = [-1 for \_ in range(n)]  
Operations.dfsd\_line(line, Dist, node-1)  
print(Dist)  
print()

**Функция bfsd\_matrix**

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

**Функция bfsd\_line**

def bfsd\_line(graph, dist, node):  
 queue = [node]  
 while len(queue) > 0:  
 item = queue.pop(0)  
 dist[node] = 0  
 print(item+1)  
 for i in graph[item]:  
 if dist[i[0]-1] == -1:  
 queue.append(i[0]-1)  
 dist[i[0]-1] = dist[item] + 1  
 print(dist)

**Функция dfsd\_matrix**

def dfsd\_matrix(graph, dist, node):  
 stack = [node]  
 dist[node] = 0  
 while len(stack) > 0:  
 item = stack.pop()  
 print(item + 1)  
 for i in range(len(graph[item]) - 1, -1, -1):  
 if graph[item][i] == 1 and dist[i] == -1:  
 dist[i] = dist[item] + 1  
 stack.append(i)

**Функция dfsd\_line**

def dfsd\_line(graph, dist, node):  
 stack = [node]  
 while len(stack) > 0:  
 item = stack.pop()  
 dist[node] = 0  
 print(item + 1)  
 for i in range(len(graph[item]) - 1, -1, -1):  
 if graph[item][i][1] == 1 and dist[graph[item][i][0]-1] == -1:  
 dist[graph[item][i][0]-1] = dist[item] + 1  
 stack.append(graph[item][i][0]-1)