Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительной техники»

**Отчет**

По лабораторной работе №7

По курсу «Логика и основы алгоритмизации в инженерных задачах»

На тему «Поиск расстояний в взвешенном графе»

Выполнили студенты гр.20ВВ4

Горбунов Н.А.

Погосян М.Д.

Проверили:

Акифьев И. В.

Юрова О. В.

Пенза, 2021

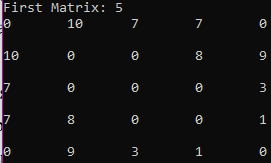
**Цель:** изучить поиск расстояний во взвешенном графе.

**Задание 1:**

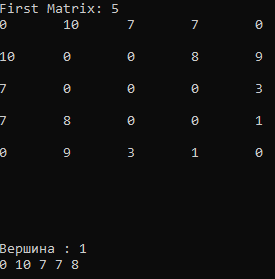
1. Сгенерировали (используя генератор случайных чисел) матрицу

смежности для неориентированного взвешенного графа G. Вывели матрицу

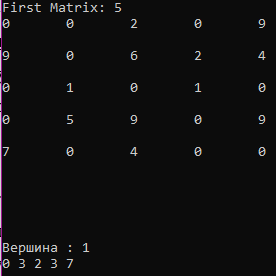
на экран:



1. Для сгенерированного графа осуществили процедуру поиска расстояний. При реализации алгоритма в качестве очереди использовали массив.



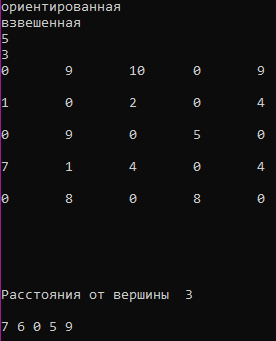
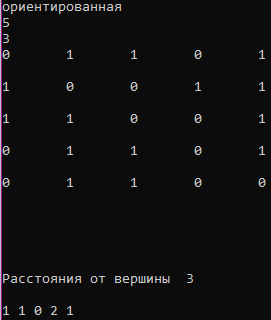
1. \*Сгенерировали (используя генератор случайных чисел) матрицу смежности для ориентированного взвешенного графа *G*. Вывели матрицу на экран и осуществили процедуру поиска расстояний:



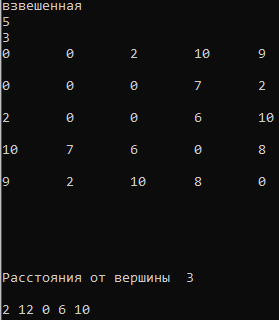
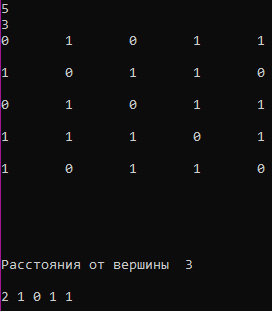
**Задание 2 \*:**

1. Модернизировали программу так, чтобы получить возможность запуска программы с параметрами командной строки. В качестве параметра указывается тип графа (взвешенный или нет) и наличие ориентации его ребер (есть ориентация или нет).

Ориентированная, взвешенная: Ориентированная, невзвешенная:

Неориентированная, взвешенная: Неориентированная, невзвешенная:

**Листинг задания 1:**

**1 и 2:**

import random

import sys

G = []

c = []

N1 = int(input("First Matrix: "))

chance = [0,0,0,1,1,1,1,1,1,1]

for i in range(N1):

G.append([])

for j in range(N1):

G[i].append(random.choice(chance))

if(G[i][j] == 1):

G[i][j] = random.randint(1,10)

if(i == j):

G[i][j] = 0

for i in range(N1):

for j in range(N1):

G[i][j] = G[j][i]

print(G[i][j],end='\t')

print('\n')

print('\n'\*3)

def BFS(v):

vis[v] = 0

queue.append(v)

while(bool(queue)):

s = queue[0]

queue.pop(0)

for i in range(len(G)):

if(G[s][i] > 0 and vis[i] > vis[s] + G[s][i]):

queue.append(i)

vis[i] = vis[s] + G[s][i]

v = int(input('Вершина : ')) - 1

vis = []

for i in range(len(G)):

vis.append(sys.maxsize)

queue = []

BFS(v)

for i in range(len(vis)):

print(vis[i], end=' ')

input()

**3:**

import random

import sys

G = []

c = []

N1 = int(input("First Matrix: "))

chance = [0,0,0,1,1,1,1,1,1,1]

for i in range(N1):

G.append([])

for j in range(N1):

G[i].append(random.choice(chance))

if(G[i][j] == 1):

G[i][j] = random.randint(1,10)

if(i == j):

G[i][j] = 0

for i in range(N1):

for j in range(N1):

print(G[i][j],end='\t')

print('\n')

print('\n'\*3)

def BFS(v):

vis[v] = 0

queue.append(v)

while(bool(queue)):

s = queue[0]

queue.pop(0)

for i in range(len(G)):

if(G[s][i] > 0 and vis[i] > vis[s] + G[s][i]):

queue.append(i)

vis[i] = vis[s] + G[s][i]

v = int(input('Вершина : ')) - 1

vis = []

for i in range(len(G)):

vis.append(sys.maxsize)

queue = []

BFS(v)

for i in range(len(vis)):

print(vis[i], end=' ')

input()

**Листинг задания 2:**

import random

import sys

import os

INPUT = sys.argv

orintation = False

weighted = False

N1 = 0

v = 0

def InputParams():

global N1

global orintation

global weighted

global v

for i in range(len(INPUT)):

try:

print(INPUT[i])

INPUT[i] = INPUT[i].upper()

if(INPUT[i] == "ОРИЕНТИРОВАННАЯ" or INPUT[i] == "ОРИНТИРОВАННАЯ" or INPUT[i] == "ОРИЕНТИРОВАНАЯ" or INPUT[i] == "ОРЕНТИРОВАННАЯ" or INPUT[i] == "ЕСТЬ ОРИЕНТАЦИЯ"):

orintation = True

if(INPUT[i] == "ВЗВЕШЕННАЯ" or INPUT[i] == "ВЗВЕШЕНАЯ" or INPUT[i] == "ВЗВЕЩЕННАЯ" or INPUT[i] == "ВЗВЕШИННАЯ" or INPUT[i] == "ВЗВЕШИНАЯ"):

weighted = True

if(N1 == 0):

N1 = int(INPUT[i])

else:

v = int(INPUT[i]) - 1

except ValueError:

continue

InputParams()

G = []

c = []

chance = [1,1,1,1,1,1,1,0,0,0]

for i in range(N1):

G.append([])

for j in range(N1):

G[i].append(random.choice(chance))

if(weighted):

if(G[i][j] == 1):

G[i][j] = random.randint(1,10)

if(i == j):

G[i][j] = 0

for i in range(N1):

for j in range(N1):

if(orintation == False):

G[i][j] = G[j][i]

print(G[i][j],end='\t')

print('\n')

print('\n'\*3)

def BFS(v):

vis[v] = 0

queue.append(v)

while(bool(queue)):

s = queue[0]

#print(s+1)

queue.pop(0)

for i in range(len(G)):

if(G[s][i] > 0 and vis[i] > vis[s] + G[s][i]):

queue.append(i)

vis[i] = vis[s] + G[s][i]

vis = []

for i in range(len(G)):

vis.append(sys.maxsize)

queue = []

BFS(v)

for i in range(len(vis)):

if(vis[i] == sys.maxsize):

print("Вершина ", i + 1, "изолированная")

vis[i] = 0

print("Расстояния от вершины ", v + 1 , "\n")

for i in range(len(vis)):

print(vis[i], end=' ')

input()

**Вывод:** изучили алгоритм поиска расстояний во взвешенном графе, также применили его на различных типах графов.