Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительной техники»

**Отчет**

По лабораторной работе №6

По курсу «Логика и основы алгоритмизации в инженерных задачах»

На тему «Поиск расстояний в графе»

Выполнили студенты гр.20ВВ4

Горбунов Н.А.

Погосян М.Д.

Проверили:

Акифьев И. В.

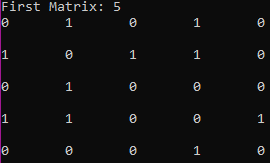
Юрова О. В.

Пенза, 2021

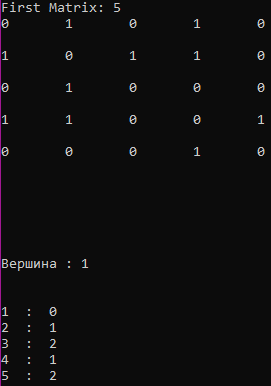
**Цель:** изучить поиск расстояний в графе.

**Задание 1:**

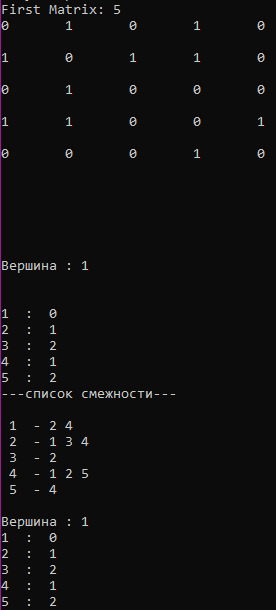
1. Сгенерировали (используя генератор случайных чисел) матрицу смежности для неориентированного графа. Вывели матрицу на экран:



1. Для сгенерированного графа осуществили процедуру поиска расстояний:

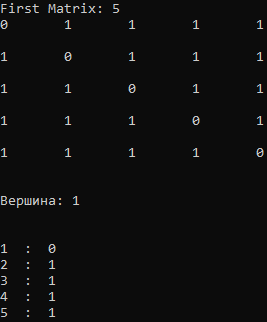
****

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

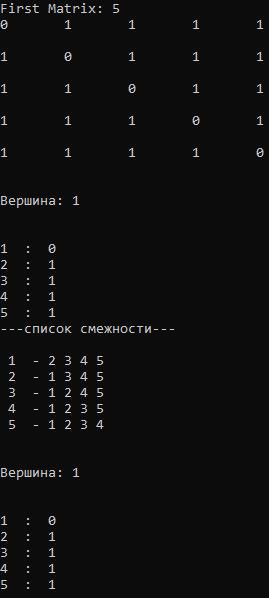
****

**Задание 2\***

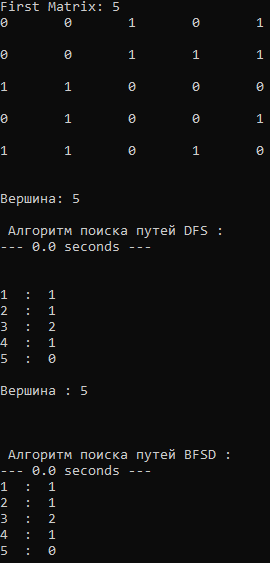
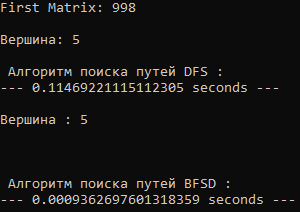
1. Реализовали процедуру поиска расстояний на основе обхода в глубину:



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



1. Оценили время работы реализаций алгоритмов поиска расстояний на основе обхода в глубину и обхода в ширину для графов разных порядков:

**Листинг задания 1:**

import random

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(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')

def BFSD(v):

Vis[v] = 1

queue.append(v)

while(bool(queue)):

s = queue[0]

queue.pop(0)

for i in range(len(G)):

if(G[s][i] == 1 and Vis[i] == -1):

queue.append(i)

Vis[i] = 1 + Vis[s]

queue = []

Vis = []

for i in range(len(G)):

Vis.append(-1)

v = int(input('\nВершина : ')) - 1

print("\n")

BFSD(v)

for i in range(len(Vis)):

print(i+1 ,' : ' , Vis[i] - 1)

print("---список смежности---")

for i in range(len(G)):

c.append([])

for j in range(len(G[i])):

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

c[i].append(j)

print("\n",i+1," - ",end='')

for j in range(len(c[i])):

print(c[i][j]+1, end=' ')

def BFSD1(v1):

Vis1[v1] = 1

queue1.append(v1)

while(bool(queue1)):

s = queue1[0]

queue1.pop(0)

for i in range(len(c[s])):

if(Vis1[c[s][i]] == -1):

queue1.append(c[s][i])

Vis1[c[s][i]] = 1 + Vis1[s]

print('\n')

v1 = int(input('Вершина : ')) - 1

Vis1 = []

for i in range(len(c)):

Vis1.append(-1)

queue1 = []

BFSD1(v1)

for i in range(len(Vis1)):

print(i+1 ,' : ' , Vis1[i] - 1)

**Листинг задания 2:**

**1 и 2:**

import random

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(i == j):

G[i][j] = 0

for i in range(len(G)):

for j in range(len(G[i])):

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

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

print('\n')

def DFS(v, count):

vis[v] = count

for i in range(len(G)):

if(G[v][i] == 1 and vis[i] == -1):

DFS(i,count + 1)

if(G[v][i] == 1 and vis[i] > count):

DFS(i,count + 1)

v = int(input("\nВершина: ")) - 1

vis = []

for i in range(len(G)):

vis.append(-1)

count = 0

DFS(v,count)

print("\n")

for i in range(len(vis)):

print(i+1,' : ' , vis[i])

print("---список смежности---")

for i in range(len(G)):

c.append([])

for j in range(len(G[i])):

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

c[i].append(j)

print("\n",i+1," - ",end='')

for j in range(len(c[i])):

print(c[i][j]+1, end=' ')

print("\n")

def DFS1(v,count):

vis[v] = count

for i in range(len(c[v])):

if(vis[c[v][i]] == -1):

DFS(c[v][i],count + 1)

if(vis[c[v][i]] > count):

DFS(c[v][i],count + 1)

v = int(input("\nВершина: ")) - 1

for i in range(len(vis)):

vis[i] = -1

count = 0

DFS1(v,count)

print("\n")

for i in range(len(vis)):

print(i+1,' : ' , vis[i])

**3:**

import random

import time

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(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')

def DFS(v):

global count

count += 1

vis[v] = 1

for i in range(len(G)):

if(G[v][i] == 1 and vis[i] == 0):

DFS(i)

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

res[i] = count

count -= 1

if(count > res[i]):

return

res[v] = count

v = int(input("\nВершина: ")) - 1

vis = []

for i in range(len(G)):

vis.append(0)

res = []

for i in range(len(G)):

res.append(0)

count = 0

print("\n Алгоритм поиска путей DFS :")

start\_time = time.time()

DFS(v)

print("--- %s seconds ---" % (time.time() - start\_time))

print("\n")

for i in range(len(res)):

print(i+1,' : ' , res[i])

def BFSD(v):

Vis[v] = 1

queue.append(v)

while(bool(queue)):

s = queue[0]

queue.pop(0)

for i in range(len(G)):

if(G[s][i] == 1 and Vis[i] == -1):

queue.append(i)

Vis[i] = 1 + Vis[s]

queue = []

Vis = []

for i in range(len(G)):

Vis.append(-1)

v = int(input('\nВершина : ')) - 1

print("\n")

print("\n Алгоритм поиска путей BFSD :")

start\_time = time.time()

DFS(v)

print("--- %s seconds ---" % (time.time() - start\_time))

BFSD(v)

for i in range(len(Vis)):

print(i+1 ,' : ' , Vis[i] - 1)

**Вывод:** изучили процедуру поиска расстояний в графе на основе поиска в глубину и также применили это на списки смежности, оценили время работы реализаций алгоритмов поиска расстояний на основе обхода в глубину и обхода в ширину для графов разных порядков.