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»

**Выполнили:**

студент группы 20ВВ3

Шмелёв Д.В.

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пенза 2021

# Название

Поиск расстояний во взвешенном графе

**Цель работы** – изучение алгоритма поиска расстояний во взвешенном графе.

# Методические указания

Отличие от поиска расстояний в не взвешенном графе будет состоять в том, что при обновлении расстояния до вершины при ее посещении оно будет увеличиваться не на 1, а на величину веса ребра.

# Лабораторное задание

**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.

При реализации алгоритма в качестве очереди используйте класс queue из стандартной библиотеки С++.

3.\* Сгенерируйте (используя генератор случайных чисел) матрицу смежности для ориентированного взвешенного графа G. Выведите матрицу на экран и осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием.

**Задание 2\***

1. Модернизируйте программу так, чтобы получить возможность запуска программы с параметрами командной строки (см. описание ниже). В качестве параметра должны указываться тип графа (взвешенный или нет) и наличие ориентации его ребер (есть ориентация или нет).

**Псевдокод**

Вход: G – матрица смежности графа, v – исходная вершина.

Выход: DIST – вектор расстояний до всех вершин от исходной.

Алгоритм ПОШ

1.1. для всех i положим DIST [i] = 1000 пометим как «не посещенную»;

1.2. ВЫПОЛНЯТЬ BFSD (v).

1.3 для всех i вывести DIST [i] на экран;

Алгоритм BFSD(v):

2.1. Создать пустую очередь Q = {};

2.2. Поместить v в очередь Q.push(v);

2.3. Обновить вектор расстояний DIST [ x ] = 0;

2.4. ПОКА Q != ∅ очередь не пуста ВЫПОЛНЯТЬ

2.5. v = Q.front() установить текущую вершину;

2.6. Удалить первый элемент из очереди Q.pop();

2.7. вывести на экран v;

2.8. ДЛЯ i = 1 ДО size\_G ВЫПОЛНЯТЬ

2.9. ЕСЛИ G(v,i) &gt; 0 И DIST [ i ] &gt; DIST [ v ] + G(v,i)

2.10. ТО

2.11. Поместить i в очередь Q.push(i);

2.12. Обновить вектор расстояний DIST [ i ] = DIST [ v ] +

G(v,i);

# Листинг

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_CRT\_NONSTDC\_NO\_WARNINGS

#define HEADER ("Лабораторная работа №7\nВыполнили: Шмелёв Д. и Пантюшов Е.\n\n")

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

#include <queue>

using namespace std;

void BFSD(int\*\* M, int n, int\* dist, int x)

{

queue <int> Q;

Q.push(x);

dist[x] = 0;

while (!Q.empty())

{

x = Q.front();

Q.pop();

printf(" %d ", x);

for (int i = 0; i < n; i++)

{

if (M[x][i] > 0 && dist[i] > dist[x] + M[x][i])

{

Q.push(i);

dist[i] = dist[x] + M[x][i];

}

}

}

}

int\*\* createMatrix(char\* argv[], int n)

{

int\*\* M = (int\*\*)(malloc(n \* sizeof(int\*)));

for (int i = 0; i < n; i++)

{

M[i] = (int\*)(malloc(n \* sizeof(int)));

for (int j = 0; j < n; j++)

{

if (rand() % 100 < 40 || i == j)

{

M[i][j] = 0;

}

else

{

if (!strcmp(argv[1], "-0"))

{

M[i][j] = 1;

}

else

{

M[i][j] = rand() % n;

}

}

}

}

if (!strcmp(argv[2], "-0"))

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (M[i][j] > 0)

{

M[j][i] = M[i][j];

}

}

}

}

return M;

}

void printMatrix(int\*\* M, int n)

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

printf(" %d ", M[i][j]);

}

printf("\n");

}

}

void func(int\*\* M, int n)

{

printf("\nМатрица смежности:\n");

printMatrix(M, n);

int\* dist = (int\*)(malloc(n \* sizeof(int)));

int x;

printf("\nВведите номер вершины: ");

scanf("%d", &x);

for (int i = 0; i < n; i++)

{

dist[i] = INT\_MAX;

}

BFSD(M, n, dist, x);

printf("\nРезультат работы алгоритма поиска расстояний: \n");

for (int i = 0; i < n; i++)

{

if (dist[i] == INT\_MAX)

{

dist[i] = 0;

}

printf("%d: %d\n", i, dist[i]);

}

}

int main(int argc, char\* argv[])

{

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

printf(HEADER);

if (argc > 1)

{

int n;

sscanf(strtok(argv[3], "-"), "%d", &n);

int\*\* M = createMatrix(argv, n);

func(M, n);

}

else

{

printf("Введите аргументы!\n");

}

system("PAUSE");

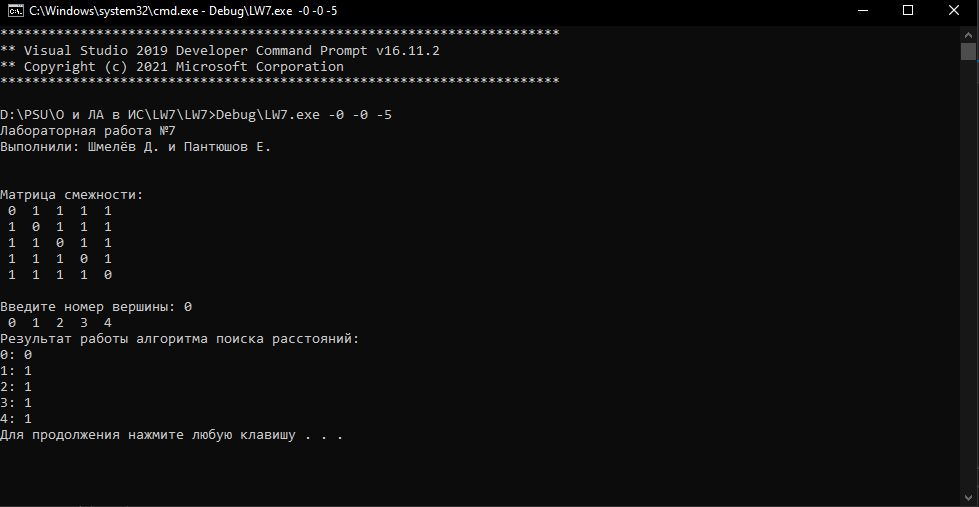
return 0;

}

# Результат работы программы

**Пункт 1.1-2 и 2.1**

Результаты работы программы показаны на рисунке 1.



**Рисунок 1 – Результат работы программы**

# Вывод

# В ходе выполнения лабораторной работы была разработана программа, в который был реализован алгоритм поиска расстояний в графа на основе алгоритма поиска в ширину. Также были изучены основные параметры функции main() и были приобретены навыки по запуску программы с выбранными параметрами командной строки.