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 20ВВ3

Новиков Иван

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пеза 2021

# **Название**

Поиск расстояний во взвешенном графе

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

# **Методические указания**

Во взвешенном графе в отличие от не взвешенного каждое ребро имеет вес, отличный от нуля. Поэтому в матрице смежности взвешенного графа содержится информация не только о наличии ребра, но и о его весе.

Поиск расстояний между вершинами в таком графе также возможно построить используя процедуры обхода графа. 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.

Таким образом, можно предложить следующую реализацию алгоритма обхода в ширину.

**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) > 0 **И** DIST [ i ] > 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);

**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.

**Ход Работы**

**Листинг**

#define \_CRT\_NONSTDC\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#include <string.h>

#include <windows.h>

#include <queue>

using namespace std;

void matrix(int\*\* g, int size)

{

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

{

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

{

g[i][j] = rand() % 11;

}

}

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

{

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

{

if (i == j)

g[i][j] = 0;

else

g[i][j] = g[j][i];

}

}

printf("Матрица: \n");

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

printf("%3d", g[i][j]);

}

printf("\n");

}

}

void F(int\*\* g, int\* dist, int v, int size)

{

queue <int> q;

q.push(v);

dist[v] = 0;

while (!q.empty())

{

v = q.front();

q.pop();

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

{

if ((g[v][i] > 0) && (dist[i]) > dist[v]+g[v][i])

{

q.push(i);

dist[i] = dist[v]+g[v][i];

}

}

}

printf("\n\n");

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

{

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

}

}

void main()

{

setlocale(LC\_ALL, "RUS");

int i = 0;

int j = 0;

int size = 0;

int\*\* g = 0;

int v = 0;

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

printf("Размер матрицы: ");

scanf("%d", &size);

g = (int\*\*)malloc(size \* sizeof(int\*));

for (i = 0; i < size; i++)

{

g[i] = (int\*)malloc(size \* sizeof(int));

}

matrix(g, size);

printf("Точка входа: ");

scanf("%d", &v);

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

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

{

dist[i] = INT16\_MAX;

}

F(g, dist, v, size);

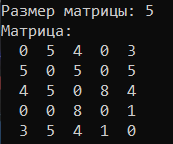
printf("\n");

system("pause");

}

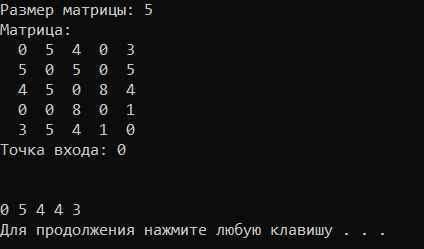
**Результат работы программы:**

1. Сгенерировал (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного графа G и вывели матрицу на экран.



*Рисунок 1. Генерация массива*

2. Для сгенерированного графа осуществили процедуру поиска расстояний.



*Рисунок 2. Ввод точки входа и результат выполнения программы.*

**Вывод:** изучил поиск расстояний в неориентированном взвешенном графе. Написал и протестировал программу, которая создает граф определенного размера и ищет расстояния.