Пензенский государственный университет

Кафедра "Вычислительной техники"

**Отчёт**

По лабораторной №7

по дисциплине "Л и ОА в ИТ"

на тему "Поиск расстояний во взвешенном графе"

***Выполнил студент группы 20ВВ1:***

Тюгаев К.А.

***Приняли:***

Юрова О.В.

Акифьев И.В.

*.*

Пенза 2021

# Цель работы: выполнить поиск расстояний во взвешенном графе.

**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.

**Теоретическая часть:**

Поиск расстояний – довольно распространенная задача анализа графов.

Для поиска расстояний можно использовать процедуры обхода графа. Для этого при каждом переходе в новую вершину необходимо запоминать, сколько шагов до нее мы сделали. При этом вектор, который хранил информацию о посещении вершин становится вектором расстояний. Довольно просто модернизировать для поиска расстояний в графе алгоритм обхода в ширину, т.к. этот алгоритм проходит вершины по уровням удаленности, то для не ориентированного графа для вершин каждого следующего уровня глубины расстояние от исходной вершины увеличивается на 1.

Удалённость в данном случае понимается как количество ребер, по которым необходимо прейти до достижения вершины.

**Практическая часть:**

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <locale.h>

#include <queue>

**using** **namespace** std;

**int**\* NUM;

**int** num;

**int**\*\* matrix;

**void** BFSD(**int** num);

**void** BFSD(**int** v) {

// int s;

queue<**int**> q;

NUM[v] = 0;

q.push(v);

**while** (!q.empty()) {

v = q.front();

q.pop();

printf("%4i", v);

**for** (**int** i = 0; i < num; i++) {

**if** (matrix[v][i] > 0 && NUM[i] > (matrix[v][i] + NUM[v])) {

q.push(i);

NUM[i] = matrix[v][i] + NUM[v];

}

}

}

printf("\nРасстояние до вершин: ");

**for** (**int** i = 0; i < num; i++) {

printf("%4i", NUM[i]);

}

}

**int** main()

{

setlocale(LC\_ALL, "Rus");

**int** generate;

**int** c;

printf("Введите размер массива: ");

scanf("%i", &num);

matrix = **new** **int**\* [num];

**for** (**int** i = 0; i < num; i++) {

matrix[i] = **new** **int**[num];

}

srand(time(**NULL**));

**for** (**int** i = 0; i < num; i++) {

**for** (**int** j = 0; j < num; j++) {

**if** (j == i) {

matrix[i][j] = 0;

}

**else** **if** (j > i) {

generate = rand() % 110;

**if** (generate >= 50) {

matrix[i][j] = rand() % 10;

}

**else** {

matrix[i][j] = 0;

}

}

**else** {

matrix[i][j] = matrix[j][i];

}

}

}

//вывод массива

printf("Массив:");

**for** (**int** i = 0; i < num; i++) {

printf("\n");

**for** (**int** j = 0; j < num; j++) {

printf("%4i", matrix[i][j]);

}

}

NUM = **new** **int**[num];

**for** (**int** i = 0; i < num; i++) {

NUM[i] = 10000;

}

printf("\nВведите вершину начала обхода: ");

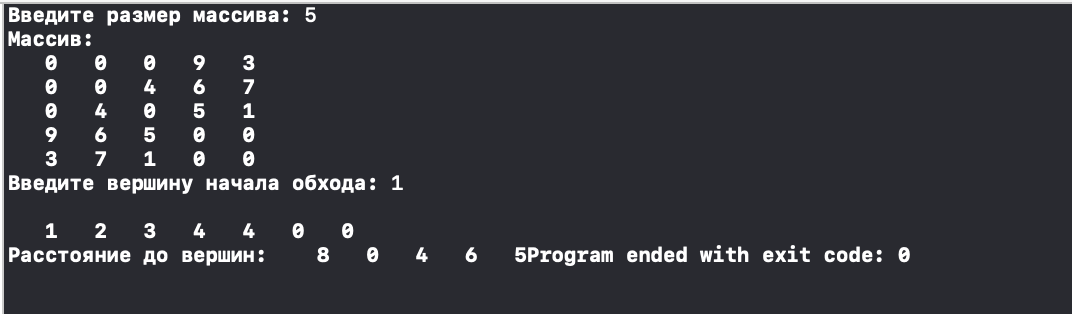
scanf("%i", &c);

printf("\n");

BFSD(c);

}

**Результат работы программы:**



**Вывод:** научились выполнять поиск расстояний во взвешенном графе.