|  |
| --- |
| Министерство образования Российской Федерации  Пензенский государственный университет  Кафедра «Вычислительная техника» |
| Отчет  по лабораторной работе №7  по курсу «Л и ОА в ИЗ»  на тему «Поиск расстояний во взвешенном графе» |
|  |
|  |
| Выполнил:  студент группы 20ВВ4  Кривцов Н.А.  Слабинский Н.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.

### **Задание 2**

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

**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 "conio.h"

#include <locale.h>

#include <stdlib.h>

#include <malloc.h>

#include <queue>

#include <time.h>

using namespace std;

queue <int> q;

void DFS1(int v, int\*\* matrix, int size, int iskl, int\* NUM, int chet) {

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

if (matrix[v][i] == 1 && NUM[i] == 0)

{

if (iskl != i) {

q.push(i);

}

NUM[i] = 1;

}

}

while (!q.empty()) {

printf("Расстояние от введённой вершины до вершины №%d =%d\n", q.front(), chet);

q.pop();

}

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

if (NUM[i] == 1)

{

chet = chet + 1;

NUM[i] = 2;

DFS1(i, matrix, size, iskl, NUM, chet);

}

}

}

void BFSD(int v, int\*\* G, int\* DIST, int size) {

q.push(v);

DIST[v] = 0;

while (!q.empty()) {

v = q.front();

q.pop();

//printf("В данный момент обрабатывается вершина №%d\n", v);

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

if (G[v][i] > 0 and DIST[i] == -1) {

q.push(i);

DIST[i] = DIST[v] + G[v][i];

}

}

}

}

int main()

{

srand(time(NULL));

setlocale(LC\_ALL, "RUS");

int v, n;

int k = 0;

printf("Введите размер матрицы: ");

scanf("%d", &n);

int\*\* M;

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

int\* A;

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

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

{

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

}

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

{

A[i] = -1;

}

/\*for (int i = 0; i < n; i++)

{

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

}\*/

printf("Матрицы:");

printf("\n");

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

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

if (j > k) {

int random = rand() % 2;

if (random == 0) {

M[i][j] = 0;

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

}

else {

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

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

}

}

else if (j < k) {

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

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

}

else {

M[i][j] = 0;

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

}

}

k++;

printf("\n");

}

printf("\n\n");

printf("Введите вершину, с которой будет начинаться обход: ");

scanf("%d", &v);

printf("\n");

int h = 0;

BFSD(v, M, A, n);

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

//if (i == v) {

//printf("Расстояние до вершины №%d = 0\n", i);

//}

//else {

printf("Расстояние до вершины №%d = %d\n", i, A[i]);

//}

}

/\*int chet = 1;

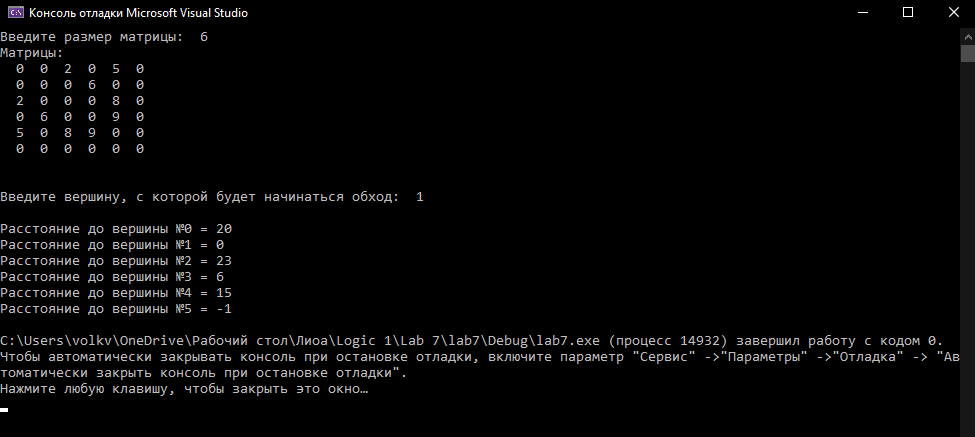
while (h != 1)

{

DFS1(v, M, n, v, A, chet);

}\*/

}



**Вывод:** Научились выполнять поиск расстояний во взвешенном графе и запуск программы с параметрами командной строки.