Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Пенза 2021

**ОТЧЁТ**

по лабораторной работе №7

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний во взвешенном графе»

Выполнили:

студентки группы 20ВВ4

Кожевникова А. В.

Мамелина Ю. В.

Приняла:

д.т.н Юрова О. В.

Пенза 2021

**Цель работы:** Изучить алгоритм поиска расстояний во взвешенном графе и написать программу в соответствии с заданиями для поиска расстояний.

**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).

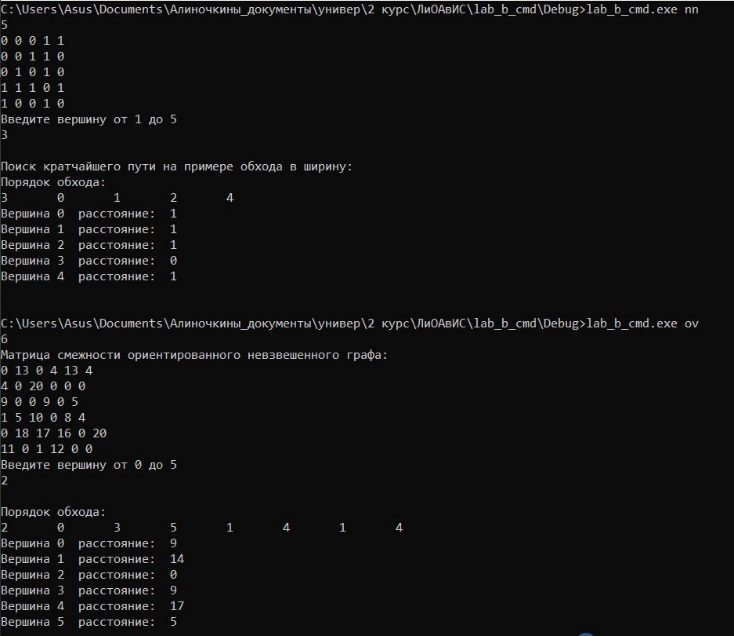


Рис 1 – результат работы задания 1.

(приложение 1)

**Вывод:** Изучили алгоритм поиска расстояний во взвешенном графе. Составили собственную программу, реализующую поиск расстояний во взвешенном графе.

**Приложение:**

Приложение 1

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

void BFSD\_for\_weighted\_matrix(int s, int\* m[], int n);

void BFSD\_for\_unweighted\_matrix(int s, int\* m[], int n);

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

{

setlocale(LC\_ALL, "rus");

int n;

cin >> n;

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

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

{

matrix[i] = new int[n];

}

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

{

if (strcmp(argv[i], "nv") == 0)

{

srand(time(NULL));

cout << "Матрица смежности неориентированного взвешенного графа:" << endl;

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

{

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

{

matrix[i][j] = rand() % 101;

if (i == j) {

matrix[i][j] = 0;

}

if (matrix[i][j] >= 0 && matrix[i][j] <= 30) {

matrix[i][j] = 0;

}

if (matrix[i][j] >= 31 && matrix[i][j] <= 100) {

matrix[i][j] = rand() % 20 + 1;

}

}

}

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

{

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

{

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

cout << matrix[i][j] << " ";

}

cout << endl;

}

int v;

cout << "Введите вершину от 0 до " << n - 1 << endl;

cin >> v;

cout << endl;

BFSD\_for\_weighted\_matrix(v, matrix, n);

cout << endl;

}

if (strcmp(argv[i], "nn") == 0)

{

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

{

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

{

matrix[i][j] = rand() % 100;

if (i == j) {

matrix[i][j] = 0;

}

if (matrix[i][j] >= 0 && matrix[i][j] <= 30) {

matrix[i][j] = 0;

}

else if (matrix[i][j] >= 31 && matrix[i][j] < 100)

{

matrix[i][j] = 1;

}

}

}

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

{

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

{

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

cout << matrix[i][j] << " ";

}

cout << endl;

}

int v;

cout << "Введите вершину от 1 до " << n << endl;

cin >> v;

cout << endl;

cout << "Поиск кратчайшего пути на примере обхода в ширину:" << endl;

BFSD\_for\_unweighted\_matrix(v, matrix, n);

cout << endl;

}

if (strcmp(argv[i], "ov") == 0)

{

srand(time(NULL));

cout << "Матрица смежности ориентированного невзвешенного графа:" << endl;

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

{

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

{

matrix[i][j] = rand() % 101;

if (i == j) {

matrix[i][j] = 0;

}

if (matrix[i][j] >= 0 && matrix[i][j] <= 30) {

matrix[i][j] = 0;

}

if (matrix[i][j] >= 31 && matrix[i][j] <= 100) {

matrix[i][j] = rand() % 20 + 1;

}

}

}

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

{

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

{

cout << matrix[i][j] << " ";

}

cout << endl;

}

int vd;

cout << "Введите вершину от 0 до " << n - 1 << endl;

cin >> vd;

cout << endl;

BFSD\_for\_weighted\_matrix(vd, matrix, n);

}

if (strcmp(argv[i], "on") == 0)

{

srand(time(NULL));

cout << "Матрица смежности ориентированного невзвешенного графа:" << endl;

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

{

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

{

matrix[i][j] = rand() % 100;

if (i == j) {

matrix[i][j] = 0;

}

if (matrix[i][j] >= 0 && matrix[i][j] <= 30) {

matrix[i][j] = 0;

}

else if (matrix[i][j] >= 31 && matrix[i][j] < 100)

{

matrix[i][j] = 1;

}

}

}

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

{

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

{

cout << matrix[i][j] << " ";

}

cout << endl;

}

int vd;

cout << "Введите вершину от 1 до " << n << endl;

cin >> vd;

cout << endl;

cout << "Поиск кратчайшего пути на примере обхода в ширину:" << endl;

BFSD\_for\_unweighted\_matrix(vd, matrix, n);

cout << endl;

}

}

system("pause>nul");

return 0;

}

void BFSD\_for\_weighted\_matrix(int s, int\* m[], int n) {

vector<int> dist(n, INT\_MAX);

queue<int> q;

q.push(s);

dist[s] = 0;

cout << "Порядок обхода:" << endl;

int vis;

while (!q.empty()) {

vis = q.front();

// Print the current node

cout << vis << "\t";

q.pop();

// For every adjacent vertex to the current vertex

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

if (m[vis][i] > 0 && dist[i] > dist[vis] + m[vis][i]) {

// Push the adjacent node to the queue

q.push(i);

dist[i] = dist[vis] + m[vis][i];

}

}

}

cout << endl;

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

{

cout << "Вершина " << i << " расстояние: " << dist[i] << " " << endl;

}

}

void BFSD\_for\_unweighted\_matrix(int s, int\* m[], int n) {

vector<int> visited(n, -1);

queue<int> q;

q.push(s);

visited[s] = 0;

cout << "Порядок обхода:" << endl;

int vis;

while (!q.empty()) {

vis = q.front();

// Print the current node

cout << vis << "\t";

q.pop();

// For every adjacent vertex to the current vertex

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

if (m[vis][i] == 1 && visited[i] == -1) {

// Push the adjacent node to the queue

q.push(i);

visited[i] = visited[vis] + 1;

}

}

}

cout << endl;

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

{

cout << "Вершина " << i << " расстояние: " << visited[i] << " " << endl;

}

}