Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Пенза 2021

**ОТЧЁТ**

по лабораторной работе №6

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний в 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 в графе и написать программу в соответствии с заданиями для решения задачи о поиске расстояний в графе.

**Задание 1**

**1.** Сгенерируйте (используя генератор случайных чисел) матрицу

смежности для неориентированного 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.**\* Реализуйте процедуру поиска расстояний для графа, представленного

списками смежности.

**Задание 2\***

**1.** Реализуйте процедуру поиска расстояний на основе обхода в глубину.

**2.** Реализуйте процедуру поиска расстояний на основе обхода в глубину

для графа, представленного списками смежности.

**3.** Оцените время работы реализаций алгоритмов поиска расстояний на

основе обхода в глубину и обхода в ширину для графов разных порядков.

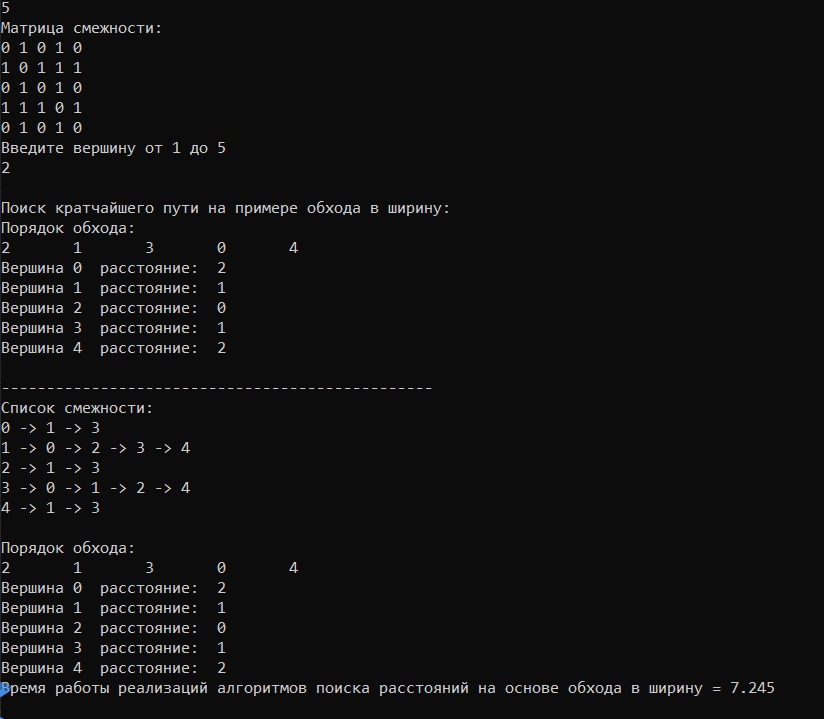


Рис 1 – результат работы задания 1 и 2.

(приложение 1)

**Вывод:** Изучили алгоритм поиска расстояний в графе и составили программу для решения для данного задания.

**Приложение:**

Приложение 1

#include <iostream>

#include <vector>

#include <queue>

#include <ctime>

using namespace std;

vector<vector<int>> convert(vector<vector<int>> a);

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

void BFSD\_for\_list(int ver, int n, vector<vector<int>> AdjList);

const int N = 1e5 + 2;

vector <int> adj[N];

int main()

{

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];

}

srand(time(NULL));

cout << "Матрица смежности:" << 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;

}

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\_matrix(v, matrix, n);

cout << endl;

vector<vector<int>>a;

a.resize(n);

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

{

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

a[i].push\_back(matrix[i][j]);

}

}

cout << "------------------------------------------------" << endl;

vector<vector<int>> AdjList = convert(a);

cout << "Список смежности:" << endl;

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

{

cout << i;

for (int j = 0; j < AdjList[i].size(); j++) {

cout << " -> " << AdjList[i][j];

}

cout << endl;

}

cout << endl;

BFSD\_for\_list(v, n, AdjList);

cout << "Время работы реализаций алгоритмов поиска расстояний на основе обхода в ширину = " << clock() / 1000.0 << endl; // время работы программы

system("pause>nul");

return 0;

}

void BFSD\_for\_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();

cout << vis << "\t";

q.pop();

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

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

q.push(i);

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

}

}

}

cout << endl;

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

{

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

}

}

vector<vector<int>> convert(vector<vector<int>> a)

{

vector<vector<int>> adjList(a.size());

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

{

for (int j = 0; j < a[i].size(); j++)

{

if (a[i][j] == 1)

{

adjList[i].push\_back(j);

}

}

}

return adjList;

}

void BFSD\_for\_list(int ver, int n, vector<vector<int>> AdjList) {

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

queue<int> q;

q.push(ver);

visited[ver] = 0;

cout << "Порядок обхода:" << endl;

int vis;

while (!q.empty()) {

vis = q.front();

cout << vis << "\t";

q.pop();

for (int i = 0; i < (int)AdjList[vis].size(); i++) {

int to = AdjList[vis][i];

if (visited[to] == -1) {

visited[to] = 1;

q.push(to);

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

}

}

}

cout << endl;

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

{

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

}

}