Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №9

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний в графе»

**Выполнили студенты группы 21вв1:**

Кирьянов В.Е.

Аляев А.О.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**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.

**Листинг программы:**

**Задание 1:**

#include "StdAfx.h"

#include <iostream>

#include <queue>

#include "time.h"

using namespace std;

int\*\* G;

int\* dist;

int BFSD(int size,int v)

{

//const int size = 7;

queue<int> Queue;

Queue.push(v);

dist[0]=0;

while (!Queue.empty())

{

int v = Queue.front();

Queue.pop();

//cout << v << " ";

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

{

if (G[v][i] == 1 && dist[i] == -1)

{ // если вершина смежная и не обнаружена

Queue.push(i);

dist[i] = dist[v]+1;

}

}

// cout << node + 1 << " ";

}

return 0;

}

int main(){

setlocale(LC\_ALL, "Russian");

int size;

cout << "Введите размер матрицы: ";

cin >> size;

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

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

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

}

srand(time(NULL));

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

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

G[i][j] = rand() % 2;

}

}

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

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

if (i == j) {

G[i][j] = 0;

}

if (G[i][j] = G[j][i]) {

G[i][j] = G[j][i];

}

}

}

/\*for (size\_t i = 0; i < size; ++i) {

for (size\_t j = 0; j < size; ++j)

cout << G[i][j] << ' ';

cout << endl;

}\*/

dist = new int [size];

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

dist[i] = -1;

cout << "Расстояния в графе: ";

int start = clock();

BFSD(size, 0);

int end = clock();

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

cout <<"\nРасстояние до "<< i <<" вершины " << dist[i] << "";

}

double time = (end-start) / 1000.0;

printf("Time: %lf", time);

}

**Задание 2:**

#include "StdAfx.h"

#include <iostream>

#include <stack>

#include "time.h"

using namespace std;

int\*\* G;

int\* dist;

int BFSD(int size,int v)

{

//const int size = 7;

stack<int> Stack;

Stack.push(v);

dist[0]=0;

while (!Stack.empty())

{

int v = Stack.top();

Stack.pop();

//cout << v << " ";

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

{

if (G[v][i] == 1 && dist[i] == -1)

{ // если вершина смежная и не обнаружена

Stack.push(i);

dist[i] = dist[v]+1;

}

}

// cout << node + 1 << " ";

}

return 0;

}

int main(){

setlocale(LC\_ALL, "Russian");

int size;

cout << "Введите размер матрицы: ";

cin >> size;

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

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

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

}

srand(time(NULL));

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

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

G[i][j] = rand() % 2;

}

}

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

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

if (i == j) {

G[i][j] = 0;

}

if (G[i][j] = G[j][i]) {

G[i][j] = G[j][i];

}

}

}

/\*for (size\_t i = 0; i < size; ++i) {

for (size\_t j = 0; j < size; ++j)

cout << G[i][j] << ' ';

cout << endl;

}\*/

dist = new int [size];

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

dist[i] = -1;

cout << "Расстояния в графе: ";

int start = clock();

BFSD(size, 0);

int end = clock();

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

cout <<"\nРасстояние до "<< i <<" вершины " << dist[i] << "";

}

double time = (end-start) / 1000.0;

printf("Time: %lf", time);

}

**Ход работы:**

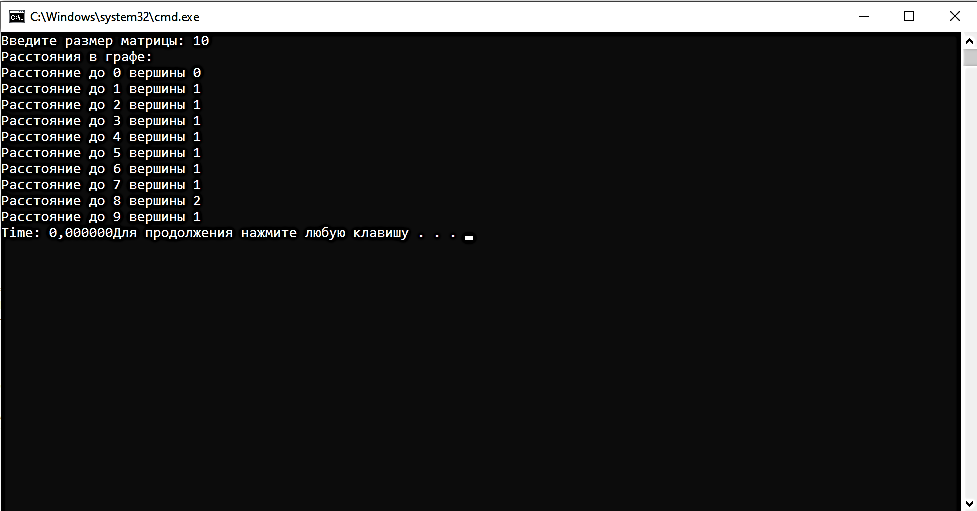
****

Рисунок 1 – результат работы программы с использование поиска в ширину.

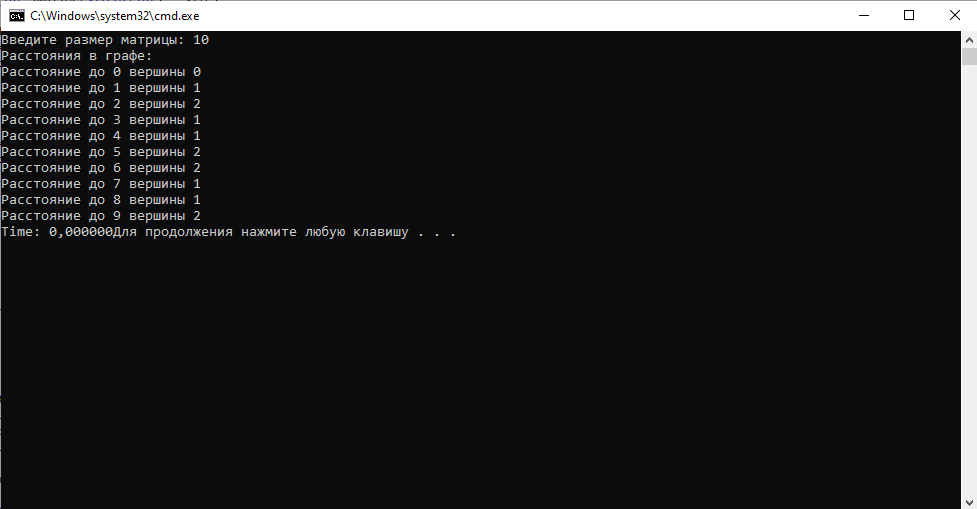
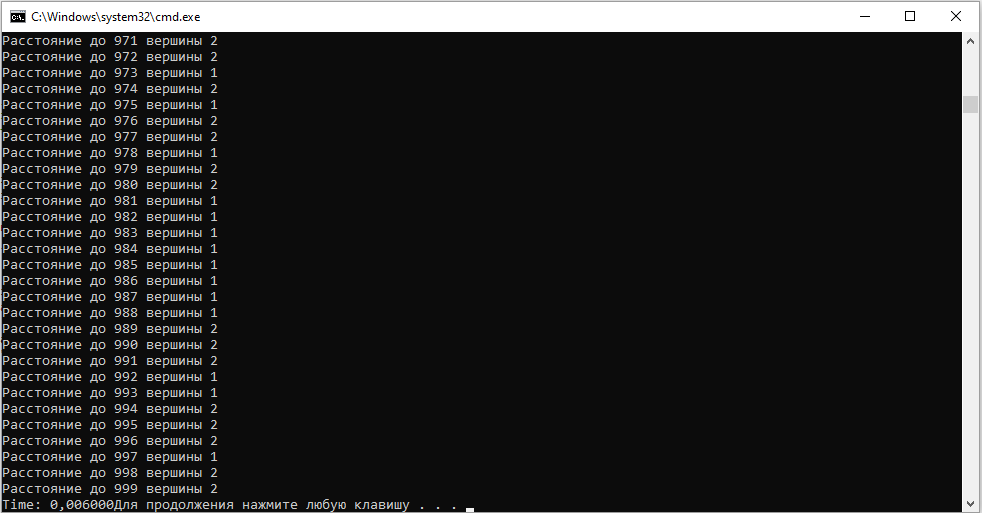


Рисунок 2 – результат работы программы с использование поиска в глубину.

**Сравнение времени работы:**

* **Алгоритм, реализованный на поиске в ширину:**

****

Число вершин:1000

Время: 0,006 с.

* **Алгоритм, реализованный на поиске в глубину:**

****

Число вершин:1000

Время: 0,002 с.

В результате сравнения быстродействия работы алгоритмов поиска расстояний в графе, реализованных на поиске в ширину и поиске в глубину, выяснилось, что их быстродействие примерно одинаково на разных диапазонах входных данных.

**Вывод:** в ходе выполнения лабораторной работы были разработана программа, выполняющая поиск расстояний в графе.

Получили опыт в создании проектов в среде Microsoft Visual Studio, научились писать и отлаживать программы с применением поиска в глубину на языке Си.