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:**

Вартанов Алексей

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания:**

**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++.

**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<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<locale.h>

#include<queue>

#include"time.h"

usingnamespacestd;

int\*\* M;

int\* dist;

int BFSD(intsize,int v)

{

queue<int> Queue;

Queue.push(v);

dist[0]=0;

while (!Queue.empty())

{

int v = Queue.front();

Queue.pop();

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

{

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

{ // если вершина смежная и не обнаружена

Queue.push(i);

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

}

}

}

return 0;

}

int main(){

setlocale(LC\_ALL, "Russian");

intsize;

printf("Введите размер матрицы: ");

scanf("%d", &size);

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

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

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

}

srand(time(NULL));

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

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

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

}

}

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

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

if (i == j) {

M[i][j] = 0;

}

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

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

}

}

}

printf("Матрицасмежности:");

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

printf("\n");

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

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

}

}

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

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

dist[i] = -1;

printf("\nРасстояния в графе: ");

int start = clock();

BFSD(size, 0);

int end = clock();

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

printf("\nРасстояние до %d вершины -> %d",i, dist[i]);

}

printf("\n");

double time = (end-start) / 1000.0;

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

system("Pause");

return 0;

}

**Задание2:**

#include<iostream>

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<locale.h>

#include<queue>

#include"time.h"

#include<stack>

usingnamespacestd;

int\*\* M;

int\* dist;

int BFSD(intsize,int v)

{

stack<int>Stack;

Stack.push(v);

dist[0]=0;

while (!Stack.empty())

{

int v = Stack.top();

Stack.pop();

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

{

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

{ // если вершина смежная и не обнаружена

Stack.push(i);

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

}

}

}

return 0;

}

int main(){

setlocale(LC\_ALL, "Russian");

intsize;

printf("Введите размер матрицы: ");

scanf("%d", &size);

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

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

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

}

srand(time(NULL));

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

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

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

}

}

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

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

if (i == j) {

M[i][j] = 0;

}

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

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

}

}

}

printf("Матрицасмежности:");

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

printf("\n");

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

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

}

}

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

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

dist[i] = -1;

printf("\nРасстояния в графе: ");

int start = clock();

BFSD(size, 0);

int end = clock();

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

printf("\nРасстояние до %d вершины -> %d",i, dist[i]);

}

printf("\n");

double time = (end-start) / 1000.0;

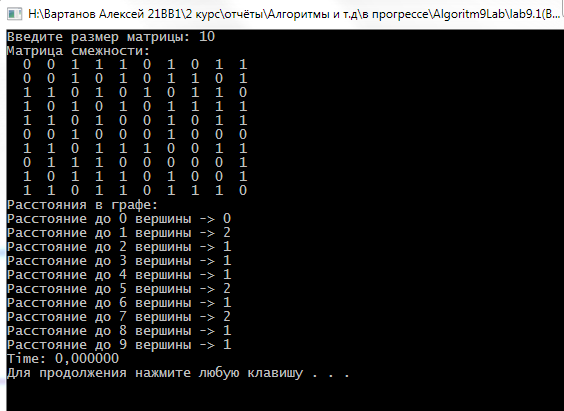
printf("Time: %lf\n", time);

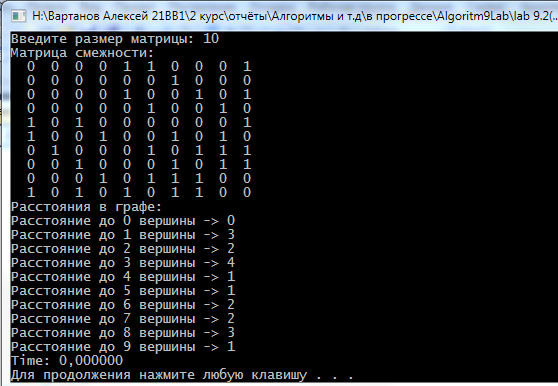
system("Pause");

return 0;

}

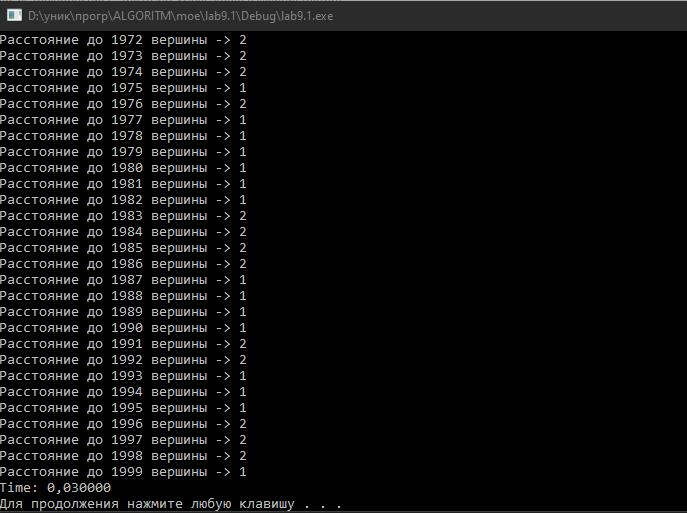
**Ходработы:**

****

****

**Сравнение времени работы:**

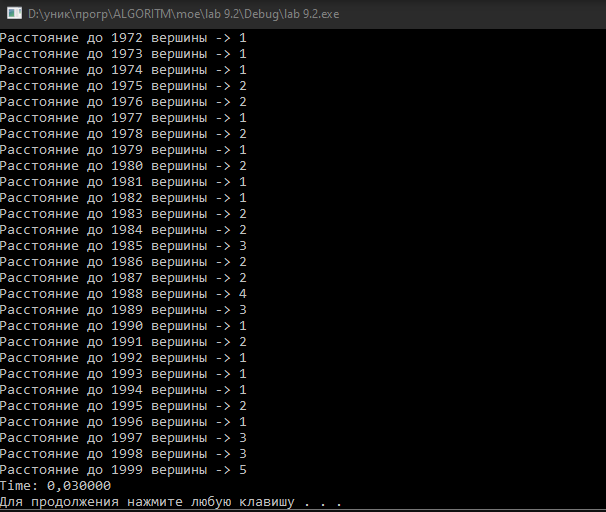
* **Алгоритм, реализованный на поиске в ширину:**

****

Число вершин:2000

Время: 0,03000 с.

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

****

Число вершин:2000

Время: 0,03000 с.

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

**Вывод:**

В ходе выполнения лабораторной работы были разработана программа, выполняющая поиск расстояний в графе.

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