Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №7

# по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «поиск расстояния во взвешенном графе»

**Выполнил:**

студент группы 20ВВ3

Шадрин Данила

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пенза 2021

# Название

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++.

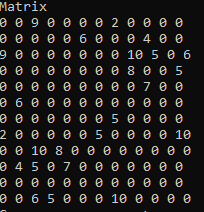
**3.**\* Сгенерируйте (используя генератор случайных чисел) матрицу смежности для ориентированного взвешенного графа *G*. Выведите матрицу на экран и осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием.

**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).

# Результат работы программы

**Задание 1.1.** Результаты работы программы показаны на рисунке 1.



**Рисунок 1 – Результат работы программы**

**Задание 1.2.** Результат поиска расстояния.



**Рисунок 2 – Результат работы программы**

**Вывод:** яизучил алгоритм поиска расстояний во взвешенном графе.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <conio.h>

#include <time.h>

#include <stack>

#include <vector>

#include <queue>

using namespace std;

void BFS(int n, int\* Arr2, int i, int\*\* Arr)

{

queue<int> que;

que.push(i);

Arr2[i] = 0;

while (!que.empty())

{

i = que.front();

que.pop();

cout << i + 1 << " ";

for (int j = 1; j < n; j++)

{

if ((Arr[i][j] > 0) && (Arr2[j] > Arr2[i] + Arr[i][j]))

{

que.push(j);

Arr2[j] = Arr2[i] + Arr[i][j];

}

}

}

}

void main() {

system("chcp 1251"); system("cls");

int start, n;

cout << "Задание 1" << endl;

cout << "Количество вершин > "; cin >> n;

int\* visited = new int[n];

int\*\* matrix;

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

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

//matrix[i] = new int[n];

visited[i] = 1000;

}

int s = 1;

srand(time(NULL));

for (int i = 0; i < n; i++) // создание матрицы

{

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

matrix[i][i] = 0;

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

{

if (rand() % 100 > 30)

{

matrix[i][j] = 0;

}

else {

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

}

}

s++;

}

s = 1;

for (int i = 0; i < n; i++) // неориентированный граф

{

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

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

s++;

}

cout << "Matrix"; // вывод матрицы на экран

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

{

cout << endl;

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

{

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

}

}

cout << endl;

cout << "Стартовая вершина > "; cin >> start;

cout << "Порядок обхода: ";

float t = clock();

BFS(n, visited, start - 1, matrix);

float Time = t / CLOCKS\_PER\_SEC;

cout << endl;

cout << "Ves = ";

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

if (visited[i] == 1000) {

visited[i] = -1;

cout << visited[i] << " ";

}

cout << visited[i] << " ";

}

delete[]visited;

cout << fixed;

cout.precision(5);

cout << "\nТакты:" << Time << endl;

system("pause");

}