Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе № 10  
по дисциплине: «Поиск расстояний во взвешенном графе»

Выполнили студенты группы 22ВВВ2:

Сергунов М.

Ипполитов И.

Приняли:  
Акифьев И. В.

Митрохин М. А.

Пенза 2023

**Название**

Поиск расстояний во взвешенном графе

**Цель работы**

Разработать и реализовать алгоритм нахождения 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. Для каждого из вариантов сгенерированных графов (ориентированного и не ориентированного) определите радиус и диаметр.
2. Определите подмножества периферийных и центральных вершин.

Задание 3\*

1. Модернизируйте программу так, чтобы получить возможность запуска программы с параметрами командной строки (см. описание ниже).  В качестве параметра должны указываться тип графа (взвешенный или нет) и наличие ориентации его ребер (есть ориентация или нет).

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

#include <queue>

#include <clocale>

#define MAX\_NODES 400

// Функция для генерации случайной матрицы смежности

void generateRandomGraph(int graph[MAX\_NODES][MAX\_NODES], int n) {

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

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

if (i == j) {

graph[i][j] = 0; // Нет петель

}

else {

int random\_value = rand() % n; // Генерируем 0 или 1

graph[i][j] = random\_value;

graph[j][i] = random\_value; // Граф неориентированный

}

}

}

}

// Функция для поиска расстояний в графе с использованием BFS

int bfs(int graph[MAX\_NODES][MAX\_NODES], int n, int startNode) {

bool visited[MAX\_NODES] = { false };

int distances[MAX\_NODES] = { 0 };

std::queue<int> q;

q.push(startNode);

visited[startNode] = true;

while (!q.empty()) {

int currentNode = q.front();

q.pop();

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

if (graph[currentNode][i] && !visited[i]) {

q.push(i);

visited[i] = true;

distances[i] = distances[currentNode] + graph[startNode][i];

}

}

}

// Выводим расстояния от startNode до всех остальных узлов

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

//printf("\nРасстояние от ребра %d до ребра %d: %d\n", startNode+1, i+1, distances[i]);

}

int max = 0;

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

if (distances[i] > max) {

max = distances[i];

}

}

return max;

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

if (distances[i] == min) {

min = distances[i];

}

}

printf("\nРадиус равен: %d\nДиаметр равен: %d\n", min, max);

int midlle[MAX\_NODES] = { 0 };

int perf[MAX\_NODES] = { 0 };

printf("\nПодмножество центральных вершин: ");

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

if (distances[i] == min) {

printf("%d ", i+1);

}

}

printf("\nПодмножество перефирийнных вершин: ");

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

if (distances[i] == max) {

printf("%d ", i+1);

}

}\*/

}

int main() {

setlocale(LC\_ALL, "Russian");

srand(time(NULL)); // Инициализируем генератор случайных чисел

int n; // Количество узлов в графе

printf("Введите количество ребер в графе: ");

scanf("%d", &n);

int graph[MAX\_NODES][MAX\_NODES] = { 0 };

generateRandomGraph(graph, n);

// Выводим матрицу смежности

printf("Матрица смежности графа:\n");

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

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

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

}

printf("\n");

}

int startNode;

printf("Введите начальный узел для поиска расстояний: ");

scanf("%d", &startNode);

startNode = startNode - 1;

//int x = bfs(graph, n, 0);

int min = 100;

int max = 0;

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

int x = bfs(graph, n, i);

printf("Эксцентриситет %d вершины: %d\n", i + 1, x);

if (x < min) {

min = x;

}

if (x > max) {

max = x;

}

}

printf("\nРадиус: %d\nДиаметр графа: %d\n", max, min);

printf("\nПодмножество центральных вершин: ");

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

int x = bfs(graph, n, i);

if (x == min) {

printf("%d ", i + 1);

}

}

printf("\nПодмножество перефирийнных вершин: ");

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

int x = bfs(graph, n, i);

if (x == max) {

printf("%d ", i + 1);

}

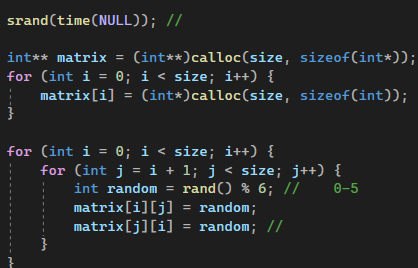
}

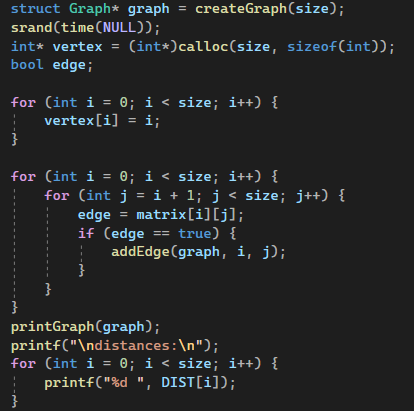
return 0;

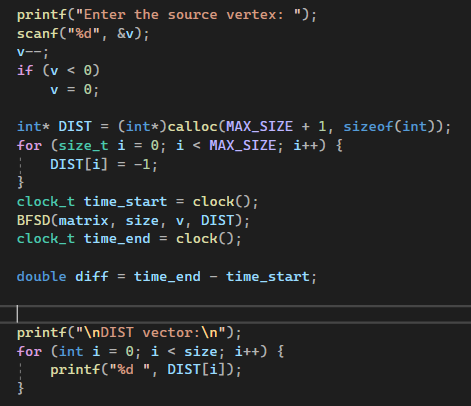
}

**Задания**

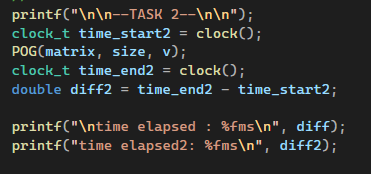
Задание 1



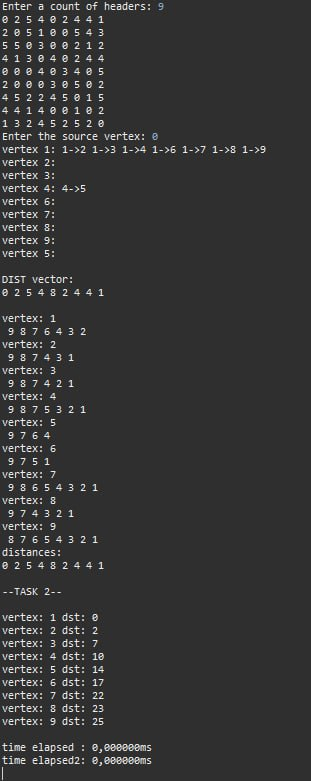




Задание 2



**Результат работы программы**



**Вывод**

Разработали и реализовали алгоритм нахождений расстояний во взвешенном графе.