Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

по дисциплине «Логика и основы алгоритмизации в инженерных задачах»

по лабораторной работе №9

## «Поиск расстояний в графе»

Выполнили:

студенты группы 22ВВС1

Казаров И.И.

Эрмедов А.Э.

Приняли:

к.т.н, доцент Юрова О.В.

к.э.н, доцент Акифьев И.В.

Пенза 2023

**Название**

Оценка времени выполнения программ

**Практическая часть**

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++.

**Листинг ConsoleApplication6.cpp**

#include <iostream>

#include <vector>

#include <queue>

#include <cstdlib>

#include <sstream>

using namespace std;

// Генерация матрицы смежности для неориентированного графа

vector<vector<int>> generateAdjacencyMatrix(int n) {

vector<vector<int>> matrix(n, vector<int>(n, 0));

cout << "Случайная матрица смежности графа (размер " << n << "x" << n << "):\n";

// Заполнение матрицы случайными значениями (0 или 1)

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

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

// Генерация случайного числа 0 или 1

int randomValue = rand() % 2;

// Запись случайного значения в матрицу

if (i != j) {

matrix[i][j] = randomValue;

}

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

}

cout << "\n";

}

return matrix;

}

// Поиск расстояний в графе с использованием очереди

void breadthFirstSearch(vector<vector<int>>& graph, int start) {

vector<string> distances(graph.size(), "Нет пути"); // Инициализация расстояний "Нет ребра"

queue<int> q;

distances[start] = "0"; // Расстояние до самого себя равно 0

q.push(start);

while (!q.empty()) {

int current = q.front();

q.pop();

for (int i = 0; i < graph[current].size(); ++i) {

if (graph[current][i] && distances[i] == "Нет пути") { // Если существует ребро и вершина не посещена

stringstream ss;

ss << stoi(distances[current]) + 1;

distances[i] = ss.str(); // Увеличиваем расстояние до вершины

q.push(i);

}

}

}

// Вывод результатов

cout << "Расстояния от вершины " << start << " до остальных вершин:\n";

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

cout << "До вершины " << i << ": ";

if (distances[i] == "Нет пути") {

cout << "Нет пути";

}

else {

cout << distances[i];

}

cout << "\n";

}

}

int main() {

setlocale(LC\_ALL, "ru\_RU");

int n;

cout << "Введите размер матрицы смежности графа: ";

cin >> n;

vector<vector<int>> adjacencyMatrix = generateAdjacencyMatrix(n);

int startVertex;

cout << "Введите номер начальной вершины: ";

cin >> startVertex;

if (startVertex < 0 || startVertex >= n) {

cout << "Некорректный номер начальной вершины!\n";

return 0;

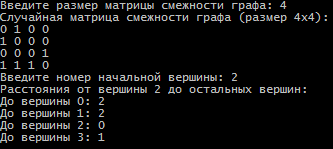
}

breadthFirstSearch(adjacencyMatrix, startVertex);

return 0;

}

**Результат**

****

**Вывод**

Сгенерировали (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Вывели матрицу на экран.

Для сгенерированного графа осуществили процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием. При реализации алгоритма в качестве очереди использовали класс **queue** из стандартной библиотеки С++.