Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительной техники»

**ОТЧЕТ**

по лабораторной работе №10

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

на тему: «Поиск расстояний во взвешенном графе.»

Выполнили**:**

студенты группы 21ВВ4

Федоренко Вероника

Роганов Данила

Проверили:

Юрова О.В,

Акифьев И.В.

Пенза 2022

**Цель работы**

Научиться реализовывать алгоритм поиска расстояний во взвешенном графе.

**Лабораторное задание**

**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

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <queue>

#include <vector>

using namespace std;

struct Eccentricity {

int vertexNumber;

int max;

};

bool checkIsVertertexAdjacent(int\*\* sourceMatrix, int firstVertexToDelite, int secondVertexToDelite) {

if (sourceMatrix[firstVertexToDelite][secondVertexToDelite] == 1) {

return true;

}

return false;

}

void allocateMatrix(int\*\*\* sourceMatrix, int columnCount, int rowsCount) {

\*sourceMatrix = (int\*\*)malloc(columnCount \* sizeof(int\*));

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

{

(\*sourceMatrix)[i] = (int\*)calloc(rowsCount, sizeof(int));

}

}

void fillMatrixRandomElements(int\*\* sourseMatrix, int columnCount, int rowsCount) {

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

{

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

{

if (i == j) {

sourseMatrix[i][j] = 0;

}

else

{

sourseMatrix[i][j] = rand() % 10;

sourseMatrix[j][i] = sourseMatrix[i][j];

}

//sourseMatrix[i][j] = rand() % 100 + 1; // заполняем массив случайными числами

}

}

}

void printMatrix(int\*\* sourseMatrix, int columnCount, int rowsCount) {

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

{

printf("\n");

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

{

printf("\t%d", sourseMatrix[i][j]);

}

}

}

Eccentricity\* breadthFirstSearch(int\*\* sourceMatrix, int matrixSize, int inputVertix) {

int\* visitedVertexs = (int\*)calloc(matrixSize, sizeof(int));

queue<int> vertexQueue;

vertexQueue.push(inputVertix);

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

{

visitedVertexs[i] = -1;

}

visitedVertexs[inputVertix] = 0;

printf("Маршрут: ");

while (!vertexQueue.empty())

{

int vertex = vertexQueue.front();

vertexQueue.pop();

printf("%d", vertex);

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

{

if (sourceMatrix[vertex][i] > 0 && visitedVertexs[i] == -1) {

vertexQueue.push(i);

visitedVertexs[i] = visitedVertexs[vertex] + sourceMatrix[vertex][i];

}

}

}

printf("\nДистанция: ");

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

{

printf("%d", visitedVertexs[i]);

}

int maxDistance = INT16\_MIN;

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

{

if (visitedVertexs[i] > maxDistance)

{

maxDistance = visitedVertexs[i];

}

}

Eccentricity\* minmax = new Eccentricity();//(Eccentricity\*)malloc(sizeof(Eccentricity));

minmax->max = maxDistance;

minmax->vertexNumber = inputVertix;

return minmax;

}

void main()

{

srand(time(NULL));

setlocale(LC\_ALL, "Rus");

int\*\* sourceMatrix = NULL;

int matrixSize = 0;

printf("Введите размер матрицы смежности = ");

scanf\_s("%d", &matrixSize);

allocateMatrix(&sourceMatrix, matrixSize, matrixSize);

fillMatrixRandomElements(sourceMatrix, matrixSize, matrixSize);

printf("\nИсходная матрица смежности\n");

printMatrix(sourceMatrix, matrixSize, matrixSize);

printf("\n\n\n");

vector<Eccentricity> allEccentricity;

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

{

Eccentricity\* minMax = breadthFirstSearch(sourceMatrix, matrixSize, i);

printf("\nЭксцентриситет вершины:%d = %d \n",i, minMax->max);

printf("\n\n");

allEccentricity.push\_back(\*minMax);

}

Eccentricity diameter = allEccentricity.at(0);

Eccentricity radius = allEccentricity.at(0);

for (int i = 1; i < matrixSize; i++)

{

Eccentricity element = allEccentricity.at(i);

if (diameter.max < element.max)

{

diameter = element;

}

if (radius.max > element.max)

{

radius = element;

}

}

printf("Диаметр графа: %d", diameter.max);

printf("\nРадиус графа: %d", radius.max);

printf("\n\n");

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

{

Eccentricity element = allEccentricity.at(i);

if (element.max == diameter.max)

{

printf("\nПериферийная вершина %d", element.vertexNumber);

}

if (element.max == radius.max)

{

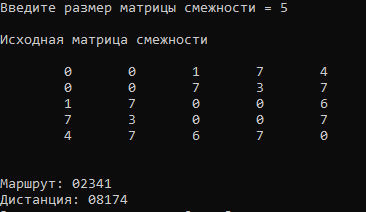
printf("\nЦентральная вершина %d", element.vertexNumber);

}

}

}

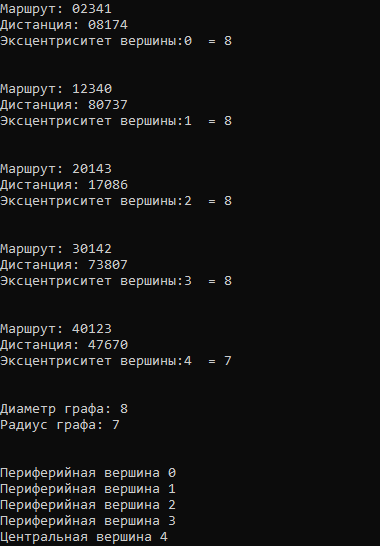
Результат



**Задание 2**

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

Результат



Вывод: в ходе выполнения лабораторной работы были разработана программа, реализующая алгоритм поиска расстояний во взвешенном графе.