Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота №1

з дисципліни «Дискретна математика»

на тему

«**Характеристики графів**»

Варіант: 26

Виконав:

студент гр. ІС - 02

Плостак Ілля

Викладач:

доц. Рибачук Л.В.

Київ – 2020

### Зміст

[Зміст 2](#_Toc190845267)

[1 Постановка задачі 3](#_Toc190845268)

[2 Результати виконання програми 4](#_Toc190845269)

[3 Лістинг програми](#_Toc190845271) 5

### 1 Постановка задачі

**Завдання**

Реалізувати програмне застосування (програму), яке виконує наступні функції: на вхід

програми подається вхідний файл з описом неорієнтованого графу, зі структурою, яка

вказана у лабораторній роботі No1 «Представлення графів» (списком ребер).

1. Визначити матриці відстаней та досяжності графу. Програма за запитом

користувача виводить на екран та/або у файл матрицю відстаней D та матрицю

досяжності R графу, за алгоритмом піднесення у степінь (конспект стор. 155-156).

2. Визначення метричних характеристик графу. Програма виводить наступні

характеристики:

a. Діаметр графу

b. Радіус графу

c. Центр графу

d. Яруси графу із переліком вершин, які входять до кожного ярусу

### 2 Результати виконання програми

### 3 Лістинг програми

[main.cpp](https://github.com/feedblackg44/kpilabs/blob/master/LabDiskretka1/LabDiskretka1/main.cpp):

#include <iostream>

#include <fstream>

#include <sstream>

#include <math.h>

#include <limits>

#include <locale.h>

#include <iomanip>

#include <string>

#include <vector>

#include "feed.h"

using namespace std;

int main()

{

SetDefaults();

ifstream inFile;

ofstream outFile;

inFile.open("./Graph.txt");

outFile.open("./Output.txt");

if (!inFile.is\_open())

{

cerr << "Error: Could not open the file!" << endl;

SystemPause();

exit(1);

}

string str\_temp;

int length = 0;

vector<int> GraphNumbers;

getline(inFile, str\_temp);

while (getline(inFile, str\_temp))

{

length++;

istringstream iss3;

int temp1, temp2;

iss3.str(str\_temp);

iss3 >> temp1;

iss3 >> temp2;

GraphNumbers.push\_back(temp1);

GraphNumbers.push\_back(temp2);

}

inFile.close();

inFile.open("./Graph.txt");

string str;

istringstream iss;

int apex, edge;

getline(inFile, str);

iss.str(str);

iss >> apex;

iss >> edge;

for (int i = 0; i < sizeof(GraphNumbers); i++)

{

if (GraphNumbers[i] > apex)

{

cerr << "Error: one of apexes is invalid!" << endl;

SystemPause();

exit(1);

}

}

if (length != edge)

{

cerr << "Error: amount of edges is invalid!" << endl;

SystemPause();

exit(1);

}

vector<vector<int>> GraphSumis(apex, vector<int>(edge));

vector<vector<int>> GraphSmezh(apex, vector<int>(apex));

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

{

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

{

GraphSumis[i][j] = 0;

}

}

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

{

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

{

GraphSmezh[i][j] = 0;

}

}

int k = 0;

while (getline(inFile, str))

{

istringstream iss2;

int temp1, temp2;

iss2.str(str);

iss2 >> temp1;

iss2 >> temp2;

GraphSmezh[temp1 - 1][temp2 - 1] = 1;

if (temp1 == temp2)

GraphSumis[temp1 - 1][k] = 2;

else

{

GraphSumis[temp1 - 1][k] = -1;

GraphSumis[temp2 - 1][k] = 1;

}

k++;

}

int mode;

while (true)

{

mode = GetInt("Input in file (1), in console (2) or both (3)?", "Invalid mode!");

if (mode < 1 || mode > 3)

cout << "Invalid mode!" << endl;

else

break;

}

if ((mode == 1) || (mode == 3))

{

outFile << "Incidence matrix:" << endl;

for (int i = -2; i < apex; i++)

{

if (i == -2)

{

outFile << " | ";

for (int k = 1; k <= edge; k++)

{

if (k < 10)

outFile << " e0" << k;

else

outFile << " e" << k;

}

}

else if (i == -1)

{

outFile << "-------";

}

else

{

if (i < 9)

outFile << " u0" << i + 1 << " |";

else

outFile << " u" << i + 1 << " |";

}

for (int j = -2; j <= edge + 1; j++)

{

if (i == -1 && j >= 0 && j < edge)

{

outFile << "----";

}

else if (i >= 0 && j >= 0 && i < apex && j < edge)

{

if (GraphSumis[i][j] >= 0)

outFile << " ";

else

outFile << " ";

outFile << GraphSumis[i][j];

}

}

if (i >= 0 && i < apex - 1)

{

outFile << endl;

outFile << " |";

}

outFile << endl;

}

outFile << endl;

outFile << "Adjacency matrix:" << endl;

for (int i = -2; i < apex; i++)

{

if (i == -2)

{

outFile << " | ";

for (int k = 1; k <= apex; k++)

{

if (k < 10)

outFile << " u0" << k;

else

outFile << " u" << k;

}

}

else if (i == -1)

{

outFile << "-------";

}

else

{

if (i < 9)

outFile << " u0" << i + 1 << " | ";

else

outFile << " u" << i + 1 << " | ";

}

for (int j = -2; j <= apex + 1; j++)

{

if (i == -1 && j >= 0 && j < apex)

{

outFile << "----";

}

else if (i >= 0 && j >= 0 && i < apex && j < apex)

{

outFile << GraphSmezh[i][j];

if (j + 1 != apex)

outFile << " ";

}

}

if (i >= 0 && i < apex - 1)

{

outFile << endl;

outFile << " |";

}

outFile << endl;

}

outFile << endl;

}

if ((mode == 2) || (mode == 3))

{

cout << "Incidence matrix:" << endl;

for (int i = -2; i < apex; i++)

{

if (i == -2)

{

cout << " | ";

for (int k = 1; k <= edge; k++)

{

if (k < 10)

cout << " e0" << k;

else

cout << " e" << k;

}

}

else if (i == -1)

{

cout << "-------";

}

else

{

if (i < 9)

cout << " u0" << i + 1 << " |";

else

cout << " u" << i + 1 << " |";

}

for (int j = -2; j <= edge + 1; j++)

{

if (i == -1 && j >= 0 && j < edge)

{

cout << "----";

}

else if (i >= 0 && j >= 0 && i < apex && j < edge)

{

if (GraphSumis[i][j] >= 0)

cout << " ";

else

cout << " ";

cout << GraphSumis[i][j];

}

}

if (i >= 0 && i < apex - 1)

{

cout << endl;

cout << " |";

}

cout << endl;

}

cout << endl;

cout << "Adjacency matrix:" << endl;

for (int i = -2; i < apex; i++)

{

if (i == -2)

{

cout << " | ";

for (int k = 1; k <= apex; k++)

{

if (k < 10)

cout << " u0" << k;

else

cout << " u" << k;

}

}

else if (i == -1)

{

cout << "-------";

}

else

{

if (i < 9)

cout << " u0" << i + 1 << " | ";

else

cout << " u" << i + 1 << " | ";

}

for (int j = -2; j <= apex + 1; j++)

{

if (i == -1 && j >= 0 && j < apex)

{

cout << "----";

}

else if (i >= 0 && j >= 0 && i < apex && j < apex)

{

cout << GraphSmezh[i][j];

if (j + 1 != apex)

cout << " ";

}

}

if (i >= 0 && i < apex - 1)

{

cout << endl;

cout << " |";

}

cout << endl;

}

cout << endl;

}

inFile.close();

outFile.close();

SystemPause();

return 0;

}

[feed.h](https://github.com/feedblackg44/kpilabs/blob/master/include/feed.h):

#pragma once

#define DEFAULT\_DELAY 20 // Default delay for PrintFormatted func

#ifndef M\_PI

#define M\_PI acos(-1.0)

#endif

double RoundTo(double number, int decimalPlace);

/\*

\* Round the number to specified precision (number of digits after the decimal point)

\*/

double GetDouble(const char promptMessage[], const char failMessage[]);

/\*

\* Get double number from user in console using promptMessage and failMessage

\*/

int GetInt(const char promptMessage[], const char failMessage[]);

/\*

\* Get integer number from user in console using promptMessage and failMessage

\*/

double Minimum(double number1, double number2);

/\*

\* Return minimum number of two numbers

\*/

void SetDefaults();

/\*

\* Everything that need to be done at the start of the program

\*/

void PrintFormatted(const char name[], double number, int width, int precision, bool align);

/\*

\* Formatted printing.

\*

\* name - number's name to show

\* number - number itself

\* width - number of digits to show (0 to igonre)

\* precision - number of digits after decimal point (0 to igonre)

\* align - whether to align numbers by a decimal point

\*/

void PrintSlow(const char str[], int delay, bool endline);

/\*

\* Slow animated-like printing.

\*

\* str - string to print

\* delay - delay between printing symbols (milliseconds)

\* endline - whether to print an endline symbol

\*/

void CoutReset();

/\*

\* Reset current output stream flags

\*/

void SleepFor(int milliseconds);

/\*

\* Program sleep time in milliseconds (same for Linux and Win)

\*/

void ClearScr();

/\*

\* Clear screen in milliseconds (same for Linux and Win)

\*/

void SystemPause();

/\*

\* Cross-platform system pause

\*/

void BirthdayParty();

/\*

\* :)

\*/

void LabHeader(int index);

/\*

\* Show index number of lab work, name and group

\*/

const char\* Declination(const char pluralWord234[], const char singleWord[], const char pluralWord[], int amount);

/\*

\* Get correct word form in dependance of its amount

\*/

[feed.cpp](https://github.com/feedblackg44/kpilabs/blob/master/include/feed.cpp):

#include <iostream>

#include <cmath>

#include <limits>

#include <locale>

#include <iomanip>

#include <string>

#include <cstring>

#include <ctime>

#include "feed.h"

#pragma warning(disable : 4996)

#ifdef \_WIN32

#include <Windows.h>

#define ClearScreen "cls"

#else

#include <unistd.h>

#define ClearScreen "clear"

#endif

#undef max

using namespace std;

void SetDefaults()

{

setlocale(LC\_ALL, "ru-RU.UTF-8");

BirthdayParty();

}

double RoundTo(double number, int decimalPlace)

{

double D = round(number \* pow(10, decimalPlace)) / pow(10, decimalPlace);

return D;

}

double GetDouble(const char promptMessage[], const char failMessage[])

{

double d\_Number;

while (true)

{

cout << promptMessage << "\n";

cin >> d\_Number;

if (cin.fail())

{

cout << failMessage << "\n";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

else

{

return d\_Number;

}

}

}

int GetInt(const char promptMessage[], const char failMessage[])

{

float f\_Number;

while (true)

{

cout << promptMessage << "\n";

cin >> f\_Number;

if (cin.fail())

{

cout << failMessage << "\n";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

else

{

if (round(f\_Number) == f\_Number)

return round(f\_Number);

else

cout << failMessage << "\n";

}

}

}

double Minimum(double number1, double number2)

{

if (number1 < number2)

return number1;

else

return number2;

}

void PrintFormatted(const char name[], double number, int width, int precision, bool align)

{

cout << name << " = ";

if (align)

cout << fixed;

else

cout << defaultfloat;

if (width > 0)

cout << setw(width);

if (precision > 0)

cout << setprecision(precision);

cout << number << endl;

}

void PrintSlow(const char str[], int delay, bool endline)

{

int count = strlen(str);

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

{

cout << str[i];

SleepFor(delay);

}

if (endline)

cout << endl;

}

void CoutReset()

{

cout << defaultfloat;

}

void SleepFor(int milliseconds)

{

#ifdef \_WIN32

Sleep(milliseconds);

#else

milliseconds \*= 1000;

usleep(milliseconds);

#endif

}

void ClearScr()

{

system(ClearScreen);

}

void SystemPause()

{

#ifdef \_WIN32

system("pause");

#else

cout << "Press any key to continue . . .";

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin.get();

cout << endl;

#endif

}

void BirthdayParty()

{

time\_t t = time(NULL);

tm\* timePtr = localtime(&t);

if ((timePtr->tm\_mday == 3) && (timePtr->tm\_mon + 1 == 12))

{

cout << "Давайте поздравим с Днём Рождения разработчика - Плостака Илью :)\n" << endl;

SystemPause();

ClearScr();

}

}

void LabHeader(int index)

{

cout << "Лабораторная работа №" << index << endl;

cout << "Выполнил: Плостак Илья" << endl;

cout << "Группа: ИС-02\n" << endl;

}

const char\* Declination(const char pluralWord234[], const char singleWord[], const char pluralWord[], int amount)

{

const char\* resultWord;

if (amount % 10 <= 4 && amount % 10 >= 2 && (amount % 100 < 12 || amount % 100 > 14))

resultWord = pluralWord234;

else if (amount % 10 == 1 && amount % 100 != 11)

resultWord = singleWord;

else

resultWord = pluralWord;

return resultWord;

}