**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра автоматизованих систем обробки інформації**

**і управління**

**Звіт**

з лабораторної роботи № 2 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

**«Прикладні задачі теорії графів ч.2»**

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-02 Гущін Д.О.*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.М.*

Київ 2021

Зміст

[1 Мета лабораторної роботи 3](#_Toc49622506)

[2 ЗаВдання 4](#_Toc49622507)

[3 Виконання 6](#_Toc49622508)

[3.1 Псевдокод алгоритму 6](#_Toc49622509)

[3.2 Програмна реалізація алгоритму 6](#_Toc49622510)

[3.2.1 Вихідний код 6](#_Toc49622511)

[Висновок 7](#_Toc49622512)

[Критерії оцінювання 8](#_Toc49622513)

# Мета лабораторної роботи

Мета роботи – вивчити додаткові прикладні алгоритми на графах та способи їх імплементації.

# ЗаВдання

Згідно варіанту (таблиця 2.1), розробити та записати алгоритм задачі на графах за допомогою псевдокоду (чи іншого способу за вибором).

Виконати програмну реалізацію алгоритму на будь-якій мові програмування для довільного графа, передбачити введення розмірності графа та введення даних графа вручну чи випадковим чином.

Для самостійно обраного графа (розмірності не менше 7 вершин) розв’язати задану за варіантом задачу вручну.

Зробити узагальнений висновок з лабораторної роботи, у якому порівняти програмне та ручне розв’язання задачі.

Таблиця 2.1 – Варіанти алгоритмів

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Задача** | **Алгоритм** | **Спосіб задання мережі** |
| 1 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 2 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 3 | Пошук усіх найкоротших шляхів | Данцига (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 4 | Пошук усіх найкоротших шляхів | Данцига (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 5 | Задача визначення збільшуючого ланцюга | За означенням | Ортграф, матриця вагів, типи дуг |
| 6 | Задача про максимальний потік | Форда - Фалкерсона | Ортграф, матриця вагів |
| 7 | Задача про максимальний потік | Едмондса - Карпа | Ортграф, матриця вагів |
| 8 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 9 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 10 | Пошук усіх найкоротших шляхів | Данцига (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 11 | Пошук усіх найкоротших шляхів | Данцига (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 12 | Задача визначення збільшуючого ланцюга | За означенням | Ортграф, матриця вагів, типи дуг |
| 13 | Задача про максимальний потік | Форда - Фалкерсона | Ортграф, матриця вагів |
| 14 | Задача про максимальний потік | Едмондса - Карпа | Ортграф, матриця вагів |
| 15 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 16 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 17 | Пошук усіх найкоротших шляхів | Данцига (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 18 | Пошук усіх найкоротших шляхів | Данцига (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 19 | Задача визначення збільшуючого ланцюга | За означенням | Ортграф, матриця вагів, типи дуг |
| 20 | Задача про максимальний потік | Форда - Фалкерсона | Ортграф, матриця вагів |
| 21 | Задача про максимальний потік | Едмондса - Карпа | Ортграф, матриця вагів |
| 22 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 23 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 24 | Пошук усіх найкоротших шляхів | Данцига (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 25 | Пошук усіх найкоротших шляхів | Данцига (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 26 | Задача визначення збільшуючого ланцюга | За означенням | Ортграф, матриця вагів, типи дуг |
| 27 | Задача про максимальний потік | Форда - Фалкерсона | Ортграф, матриця вагів |
| 28 | Задача про максимальний потік | Едмондса - Карпа | Ортграф, матриця вагів |
| 29 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 30 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 31 | Пошук усіх найкоротших шляхів | Данцига (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |
| 32 | Пошук усіх найкоротших шляхів | Данцига (зовнішній спосіб визначення шляхів) | Ортграф, матриця вагів |
| 33 | Задача визначення збільшуючого ланцюга | За означенням | Ортграф, матриця вагів, типи дуг |
| 34 | Задача про максимальний потік | Форда - Фалкерсона | Ортграф, матриця вагів |
| 35 | Задача про максимальний потік | Едмондса - Карпа | Ортграф, матриця вагів |
| 36 | Пошук усіх найкоротших шляхів | Флойда-Уоршелла (вбудований спосіб визначення шляхів) | Ортграф, матриця вагів |

# Виконання

## Псевдокод алгоритму

**for** k = 0 **to** size **do**

**for** i = 0 to size

**for** j = 0 **do** to size **do**

**if** (matrix[i][k] **and** matrix[k][j] **and** i! =j) **do**

matrix[i][j] = min(matrix[i][j], matrix[i][k] + matrix[k][j])

**end if**

**end for**

**end for**

**end for**

## Програмна реалізація алгоритму

### Вихідний код

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <iomanip>

using namespace std;

const int INF = 1000;

int\*\* ReadFile(int& size);

void FindSize(int& size);

int\*\* CreateMatrix(int&);

void FillMatrix(int\*\*, int&);

void DeleteMatrix(int\*\*, int&);

void PrintMatrix(int\*\* Matrix, int& size, string str);

void Floyd\_Warshall(int\*\* Matrix, int size, int\*\* p);

int MinValue(int a, int b);

int\*\* CreateP(int\*\* Matrix, int& size);

void IncrementP(int\*\* p, int& size);

vector<string> delim(string str, char delim);

int main()

{

int size;

int\*\* Matrix = ReadFile(size);

int\*\* p = CreateP(Matrix, size);

Floyd\_Warshall(Matrix, size, p);

IncrementP(p, size);

PrintMatrix(Matrix, size, "D(m): ");

PrintMatrix(p, size, "P(ij): ");

DeleteMatrix(Matrix, size);

DeleteMatrix(p, size);

}

int\*\* ReadFile(int& size) {

FindSize(size);

int\*\* Matrix = CreateMatrix(size);

cout << "Reading file..." << endl << endl;

FillMatrix(Matrix, size);

PrintMatrix(Matrix, size, "D(0): ");

return Matrix;

}

void FindSize(int& size) {

ifstream input("..\\iofiles\\input\_15.txt");

string currStr;

size = -1;

while (!input.eof()) {

getline(input, currStr);

size++;

cout << size << endl;

}

input.close();

input.clear();

}

int\*\* CreateMatrix(int& size) {

int\*\* Matrix = new int\* [size];

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

{

Matrix[i] = new int[size];

}

return Matrix;

}

void PrintMatrix(int\*\* Matrix, int& size, string str) {

cout << str << endl;

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

{

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

{

cout << setw(6) << Matrix[i][j] << ' ';

}

cout << endl;

}

cout << endl;

}

void FillMatrix(int\*\* Matrix, int& size) {

ifstream input("..\\iofiles\\input\_15.txt");

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

{

string currStr;

getline(input, currStr);

vector<string> tmp = delim(currStr, ' ');

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

{

if (tmp[j] == "INF") {

Matrix[i][j] = INF;

}

else {

Matrix[i][j] = stoi(tmp[j]);

}

}

}

input.close();

input.clear();

}

void DeleteMatrix(int\*\* Matrix, int& size) {

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

{

delete[] Matrix[i];

}

delete Matrix;

}

void Floyd\_Warshall(int\*\* Matrix, int size, int\*\* p) {

for (int k = 0; k < size; k++) {

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

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

if (Matrix[i][k] && Matrix[k][j] && i != j)

{

Matrix[i][j] = MinValue(Matrix[i][j], Matrix[i][k] + Matrix[k][j]);

if (Matrix[i][j] == Matrix[i][k] + Matrix[k][j]) p[i][j] = k;

}

}

}

}

}

int MinValue(int a, int b) {

int min;

(a <= b) ? min = a : min = b;

return min;

}

vector<string> delim(string str, char delim) {

int start, end = 0;

vector<string> vect;

while ((start = str.find\_first\_not\_of(delim, end)) != string::npos)

{

end = str.find(delim, start);

int length = end - start;

if (length == 0) continue;

string word = string(str, start, length);

vect.push\_back(word);

}

return vect;

}

int\*\* CreateP(int\*\* graph, int& size) {

int\*\* Matrix = new int\* [size];

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

Matrix[i] = new int[size];

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

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

if (graph[i][j] != 0 && graph[i][j] != INF) Matrix[i][j] = i;

else Matrix[i][j] = -1;

}

}

return Matrix;

}

void IncrementP(int\*\* p, int& size) {

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

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

if (p[i][j] != -1) ++p[i][j];

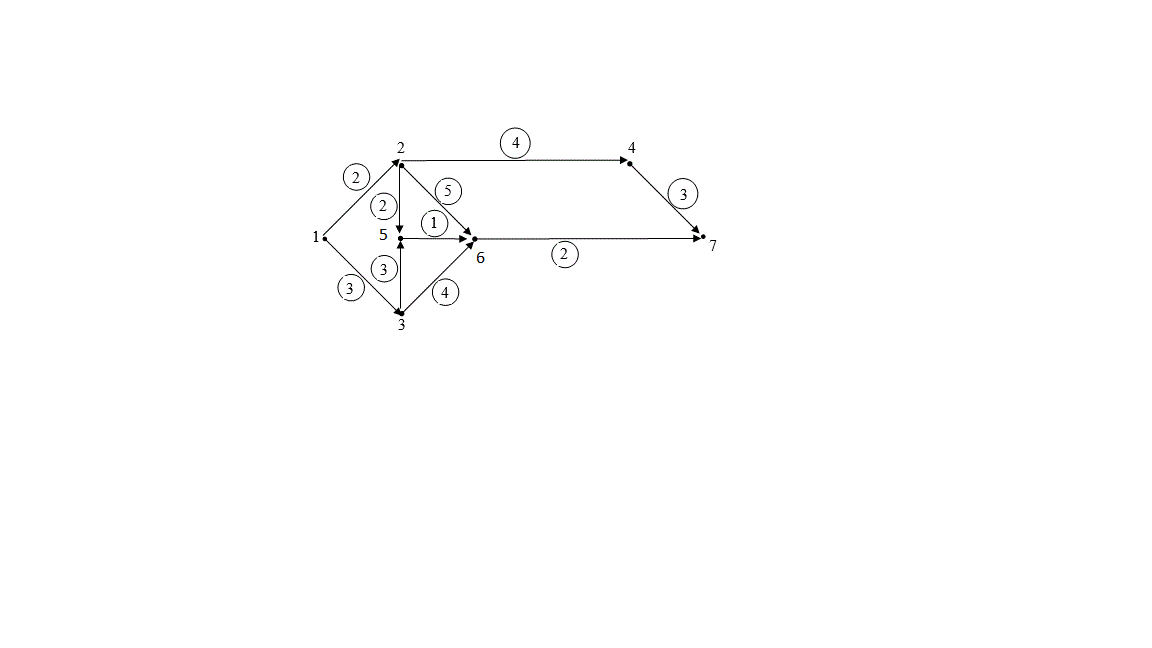
}

}

}

### Приклад роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми для графів на 7 і 15 вершин відповідно.



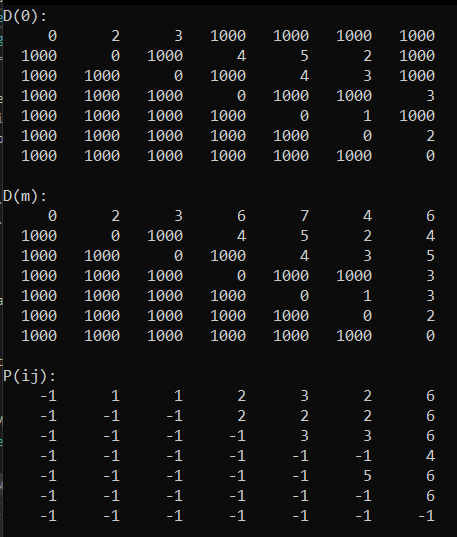
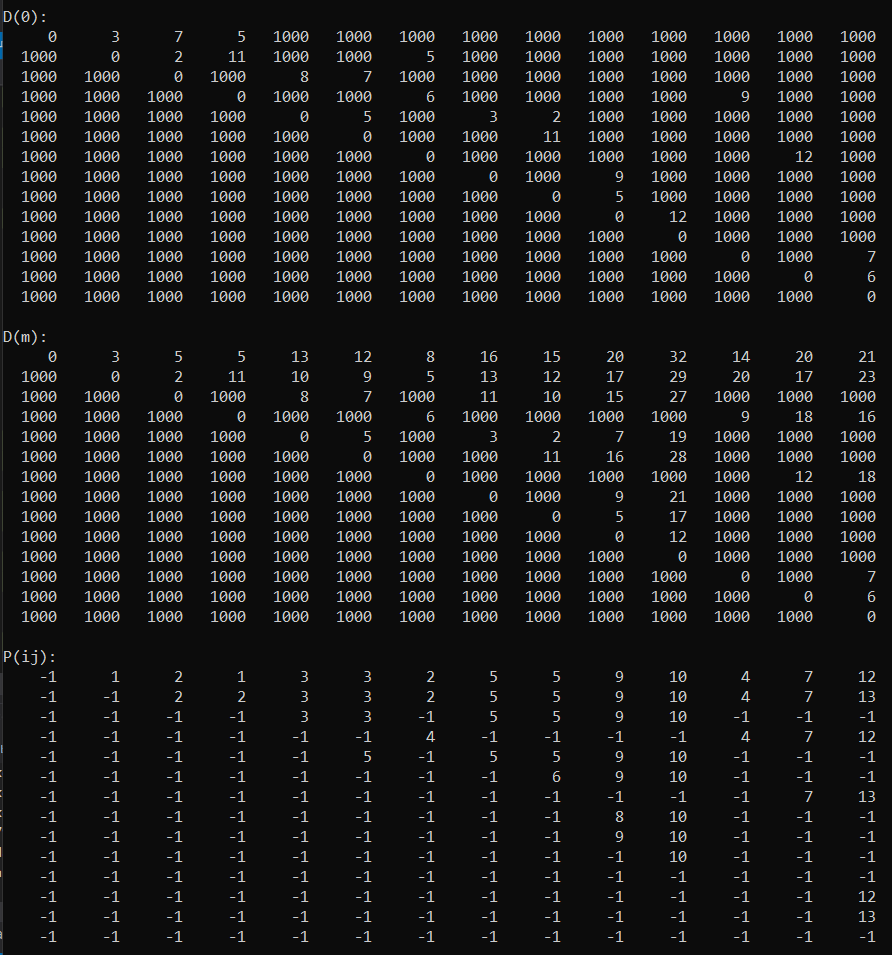


Рисунок 3.1

Рисунок 3.2

## Розв’язання задачі вручну

На рисунку 3.3 наведено розв’язання задачі вручну.

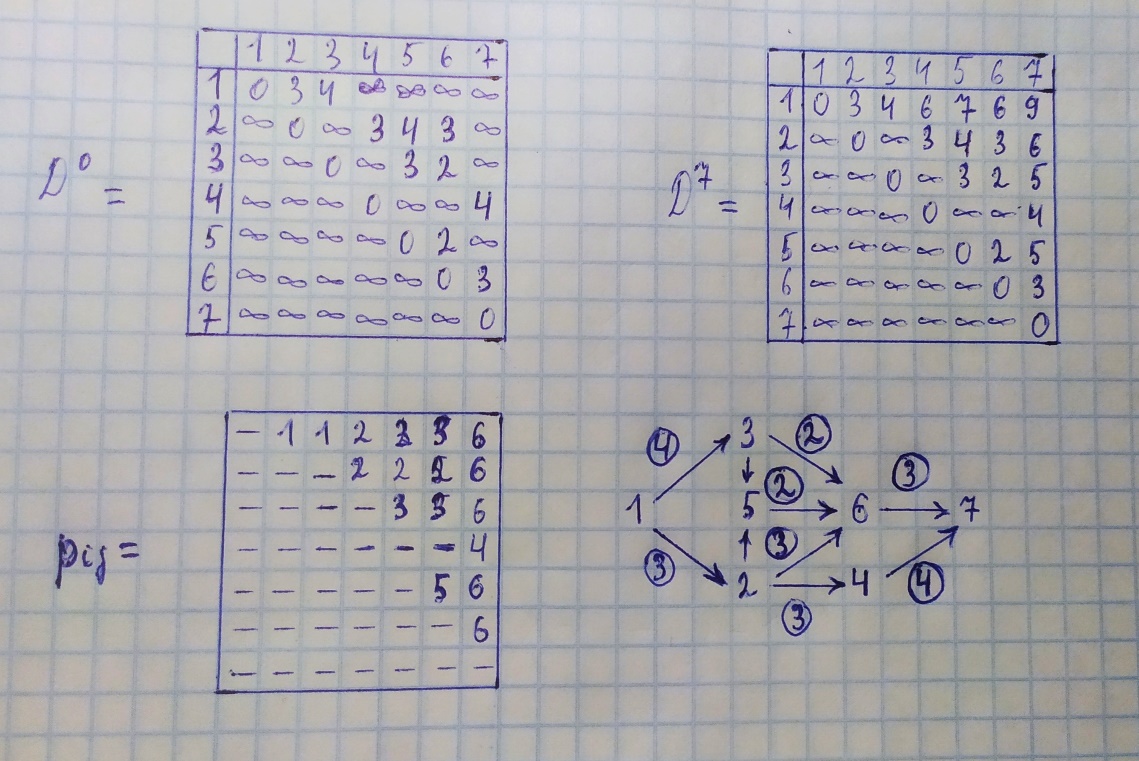


Рисунок 3.3 – Розв’язання задачі вручну

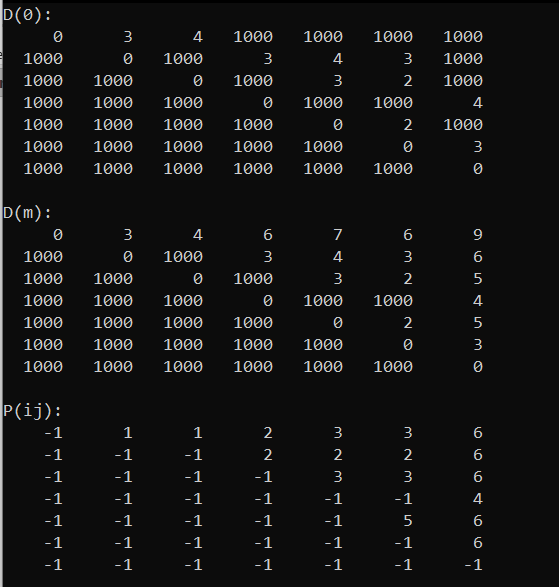


Рисунок 3.4 – Перевірка результатів

Висновок

При виконанні даної лабораторної роботи ми розглянули додаткові прикладні алгоритми на графах та способи їх імплементації, розробили та записали алгоритм Флойда-Уоршелла пошуку всіх найкоротших шляхів у довільному графі за допомогою псевдокоду, виконали програмну реалізацію алгоритму на мові С++, а також розв’язали цю задачу на прикладі конкретного графа вручну. Можна зробити висновок, що програмна реалізація алгоритму дозволяє зекономити час і сили необхідні для вирішення цієї задачі вручну.

Критерії оцінювання

У випадку здачі лабораторної роботи до 30.03.2021 включно максимальний бал дорівнює – 5. Після 30.03.2021 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

* псевдокод алгоритму – 10%;
* програмна реалізація алгоритму – 50%;
* розв’язання задачі вручну – 20%;
* відповідь на 3 теоретичні питання по темі роботи 15%
* висновок – 5%.