МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
Національний технічний університет України
Факультет інформатики та обчислювальної техніки
«Київський політехнічний інститут імені Ігоря Сікорського»
Кафедра інформаційних систем та технологій

Звіт
з лабораторної роботи № 2
«Характеристики графів»
з дисципліни
«Дискретна математика»

Варіант № 25

Перевірила:
доц. Рибачук Людмила Віталіївна

Виконала: Павлова Софія
Студентка гр. ІС-12 , ФІОТ
1 курс,
залікова книжка № ІС-1224

Київ 2021

ЗМІСТ

ВСТУП

Тема: Характеристики графів**.**

Мета: Дослідити характеристики графів та навчитись визначати їх на конкретних прикладах.

Обладнання: Персональні комп’ютери.

ХІД РОБОТИ

Завдання:

Реалізувати програмне застосування (програму), яке виконує наступні функції. Причому на вхід програми подається вхідний файл з описом графу, зі структурою, яка вказана у практичному завданні №1 «Представлення графів».

1. Визначити степінь вершин графу. За запитом користувача програма на екран та/або у файл виводить степінь усіх вершин графу (напівстепені виходу та заходу). Визначити, чи граф є однорідним та якщо так, то вказати степінь однорідності графу.
2. Визначити всі висячі та ізольовані вершини. За запитом користувача програма на екран виводить перелік усіх висячих та ізольованих вершин графу.
3. 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

Код:

```
#include <iostream>
#include <windows.h>
#include <random>
#include <string>
using namespace std;

float n, m;
int a[20][20], sum[20][20], d[20], d_plus[20], d_minus[20], izo[20], vus[20]; // massiv for sum in row and colmn, izolated and hangigng peaks
int sumr, sumc, od, c_izo, c_vus; // sum of ell in row and colmn, definder of odnor, couter of izolated and hangigng peaks
string str;
char t;

void generate() {
    random_device random_device; // entropy sourse
    mt19937 generator(random_device()); // initialise randimizer
    cout << "Вершин: ";
    cin >> n;
    if (n <= 0 || (n - int(n)) != 0) {
```

```
        cout << "\n-------------------------\nN - не удовлетворяет условие\n-------------------
------\n";
        exit(0);
    }
    uniform_int_distribution<> distribution(1, n);
    cout << "Ребер: ";
    cin >> m;
    if (m <= 0 || (m - int(m)) != 0) {
        cout << "\n-------------------------\nM - не удовлетворяет условие\n-------------------
-------\n";
        exit(0);
    }
    cout << "\n";
    // define work type
    cout << "Сгенерирровать граф или задать вручную (0/1): ";
    cin >> t;
    switch (t) {
    case '0':
        t = '0';
        break;
    case '1':
        cout << "\n\nСПИСОК РЕБЕР:";
        cout << "\n-----------\n";
        for (int i = 0; i < m; i++) {
            if (i + 1 > 9) {
                cout << "e" << i + 1 << " = ";
            }
            else {
                cout << "e" << i + 1 << "  = ";
            }
            for (int j = 0; j < 2; j++) {
                cin >> a[i][j];
                if (a[i][j] >= 1 && a[i][j] <= n) {
                    continue;
                }
                else {
                    cout << "\n-------------------------\nV" << a[i][j] << " - НЕ является точкой
графа\n-------------------------\n";
                    exit(0);
                }
            }
        }
        cout << "\n\n";
        break;
    default:
        cerr << "\n-----------------------------------------\n";
```

```
        cerr << "Неправильный тип работы программы, введите:\n0 - чтоб сгенерировать граф\n1 - чтоб задать его вручную";
        cerr << "\n-----------------------------------------\n";
        exit(0);
    }
    if (t == '0') {
        // generate
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < 2; j++) {
                a[i][j] = distribution(generator);
            }
        }
    }
}
void output() {
    if (t == '0') {
        cout << "\nСПИСОК РЕБЕР:";
        cout << "\n----------------\n";
        for (int i = 0; i < m; i++) {
            str = "   ";
            // 10-99 numbs
            if (i >= 9) {
                cout << "e" << i + 1 << " = (";
            }
            // 1-9 numbs
            else {
                cout << "e" << i + 1 << "  = (";
            }
            for (int j = 0; j < 2; j++) {
                if (j < 1) {
                    // 10-99 numbs
                    if (((a[i][j] / 10) != 0) && (a[i][j] / 10) <= 9) {
                        str = "  ";
                    }
                    // 100-999 numbs
                    else if (((a[i][j] / 10) != 0) && (a[i][j] / 10) > 9 && (a[i][j] / 10) <= 99) {
                        str = " ";
                    }
                    cout << "v" << a[i][j] << "," << str;
                }
                else {
                    cout << "v" << a[i][j] << ")\n";
                }
            }
        }
        cout << "\n\n";
```

```
    }
}
void sumizhna_matr() {
    cout << "СМЕЖНАЯ МАТРИЦА \nИ ПОЛУСТЕПЕНИ ВЕРШИН (входа+ и выхода-
):\n";
    for (int i = 0; i < m; i++) {
        sum[a[i][0]][a[i][1]] = 1;
    }
    // to define -degree of peaks
    for (int i = 0; i <= m; i++) {
        sumr = 0;
        for (int j = 0; j <= m; j++) {
            sumr += sum[i][j];
        }
        d_minus[i] = sumr;
    }
    // to define +degree of peaks
    for (int i = 0; i <= m; i++) {
        sumc = 0;
        for (int j = 0; j <= m; j++) {
            sumc += sum[j][i];
        }
        d_plus[i] = sumc;
    }
    // to define full degree of peaks
    for (int i = 0; i <= n; i++) {
        d[i] = d_plus[i] + d_minus[i];
    }
    // print
    for (int b = 0; b <= n; b++) {
        if (b == 0) {
            cout << "----------";
            cout << "----------";
        }
        else {
            cout << "----------";
        }
    }
    cout << "\n";
    for (int s = 0; s <= n; s++) {
        // first interpr
        if (s == 0) {
            // names of tables
            for (int f = 0; f <= n; f++) {
                // first interpr
                if (f == 0) {
```

```
            cout << "         |";
        }
        // last interpr
        if (f == n) {
            cout << "  d-(vi) |";
            continue;
        }
        // 10-99 numbs
        if (f >= 9) {
            cout << "    v" << f + 1 << "  |";
        }
        // 1-9 numbs
        else {
            cout << "    v" << f + 1 << "   |";
        }
    }
    cout << "\n";
}
// skelet of the tabl
for (int l = 0; l <= n; l++) {
    if (l == 0) {
        cout << "----------";
        cout << "----------";
    }
    else {
        cout << "----------";
    }
}
cout << "\n";
for (int k = 0; k <= n; k++) {
    // first interpr
    if (k == 0) {
        if (s == n) {
            cout << "  d+(vi) |";
            continue;
        }
        // 10-99 numbs
        if (s + 1 > 9) {
            cout << "   v" << s + 1 << "   |";
        }
        // 1-9 numbs
        else {
            cout << "    v" << s + 1 << "   |";
        }
    }
    // print degree- of peaks
```

```
            if (k == n && s!=n) {
                cout << "    " << d_minus[s + 1] << "    |";
            }
            // print degree+ of peaks
            else if (s == n) {
                cout << "    " << d_plus[k] << "    |";
            }
            // print matrix el
            else {
                cout << "    " << sum[s + 1][k + 1] << "    |";
            }
        }
        cout << "\n";
    }
}
void pecksFullDegree() {
    cout << "\n\nСТЕПЕНИ ВЕРШИН:\n---------------\n";
    for (int i = 0; i < n; i++) {
        // define odnor
        // prev deg = curr deg
        if (i > 0 && d_minus[i] == d_plus[i] && od != -1) {
            od = 1;
        }
        // first interpr
        else if (i == 0) {
            od = 0;
        }
        // prev deg != curr deg
        else {
            od = -1;
        }
        // define izol
        if (d[i + 1] == 0) {
            izo[c_izo] = i + 1;
            c_izo++;
        }
        // define vusyacha
        else if (d[i + 1] == 1) {
            vus[c_vus] = i + 1;
            c_vus++;
        }
        // print full degrees
        // 10-99 numbs
        if (i + 1 > 9) {
            cout << "d(v" << i + 1 << ") = " << d[i + 1] << "\n";
        }
```

```
        // 1-9 numbs
        else {
            cout << "d(v" << i + 1 << ")  = " << d[i + 1] << "\n";
        }
    }
    cout << "\n\n";
}
void peaksHalfDegree() {
    cout << "\n\nПОЛУСТЕПЕНИ ВХОДА:\n------------------\n";
    for (int i = 0; i < n; i++) {
        // 10-99 numbs
        if (i + 1 > 9) {
            cout << "d+(v" << i + 1 << ") = " << d_plus[i + 1] << "\n";
        }
        // 1-9 numbs
        else {
            cout << "d+(v" << i + 1 << ")  = " << d_plus[i + 1] << "\n";
        }
    }
    cout << "\n\nПОЛУСТЕПЕНИ ВЫХОДА:\n-------------------\n";
    for (int i = 0; i < n; i++) {
        // 10-99 numbs
        if (i + 1 > 9) {
            cout << "d-(v" << i + 1 << ") = " << d_minus[i + 1] << "\n";
        }
        // 1-9 numbs
        else {
            cout << "d-(v" << i + 1 << ")  = " << d_minus[i + 1] << "\n";
        }
    }
}
void odnor() {
    // odnorodniy
    if (od == 1) {
        cout << "ГРАФ - ОДНОРОДНЫЙ СО СТЕПЕНЬЮ " << d_minus[1] << "\n\n\n";
    }
    // nope
    else {
        cout << "ГРАФ - НЕ ОДНОРОДНЫЙ\n\n\n";
    }
}
void izol_vus() {
    // exist izol
    if (c_izo != 0) {
        cout << "ИЗОЛИРОВАНЫЕ ВЕРШИНЫ:\n--------------------\n";
        for (int i = 0; i < c_izo; i++) {
```

```
            cout << "v" << izo[i] << "\n";
        }
        cout << "\n\n";
    }
    // nope
    else {
        cout << "ИЗОЛИРОВАНЫХ ВЕРШИН - НЕТ\n\n\n";
    }
    // exist vusyach
    if (c_vus != 0) {
        cout << "ВИСЯЧИЕ ВЕРШИНЫ:\n----------\n";
        for (int i = 0; i < c_vus; i++) {
            cout << "v" << vus[i] << "\n";
        }
        cout << "\n\n";
    }
    // nope
    else {
        cout << "ВИСЯЧИХ ВЕРШИН - НЕТ\n\n\n";
    }
}
int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    generate();
    output();
    sumizhna_matr();
    peaksHalfDegree();
    pecksFullDegree();
    odnor();
    izol_vus();
}
```

Результат:

Microsoft Visual Studio Debug Console

```
Вершин: 4
Ребер: 3

Сгенеирровать граф или задать вручную (0/1): 0

СПИСОК РЕБЕР:
---------------
e1  = (v3,   v1)
e2  = (v1,   v3)
e3  = (v1,   v2)


СМЕЖНАЯ МАТРИЦА
И ПОЛУСТЕПЕНИ ВЕРШИН (входа+ и выхода-):
-------------------------------------------------------------
        |    v1   |    v2   |    v3   |    v4   | d-(vi) |
-------------------------------------------------------------
    v1  |    0    |    1    |    1    |    0    |    2   |
-------------------------------------------------------------
    v2  |    0    |    0    |    0    |    0    |    0   |
-------------------------------------------------------------
    v3  |    1    |    0    |    0    |    0    |    1   |
-------------------------------------------------------------
    v4  |    0    |    0    |    0    |    0    |    0   |
-------------------------------------------------------------
  d+(vi) |    1    |    1    |    1    |    0    |

ПОЛУСТЕПЕНИ ВХОДА:
------------------
d+(v1)  = 1
d+(v2)  = 1
d+(v3)  = 1
d+(v4)  = 0


ПОЛУСТЕПЕНИ ВЫХОДА:
-------------------
d-(v1)  = 2
d-(v2)  = 0
d-(v3)  = 1
d-(v4)  = 0


СТЕПЕНИ ВЕРШИН:
---------------
d(v1)  = 3
d(v2)  = 1
d(v3)  = 2
d(v4)  = 0


ГРАФ - НЕ ОДНОРОДНЫЙ


ИЗОЛИРОВАНЫЕ ВЕРШИНЫ:
---------------------
v4


ВИСЯЧИЕ ВЕРШИНЫ:
----------------
v2
```

Рис. 1. Результат виконання програми при завданні графа рандомом

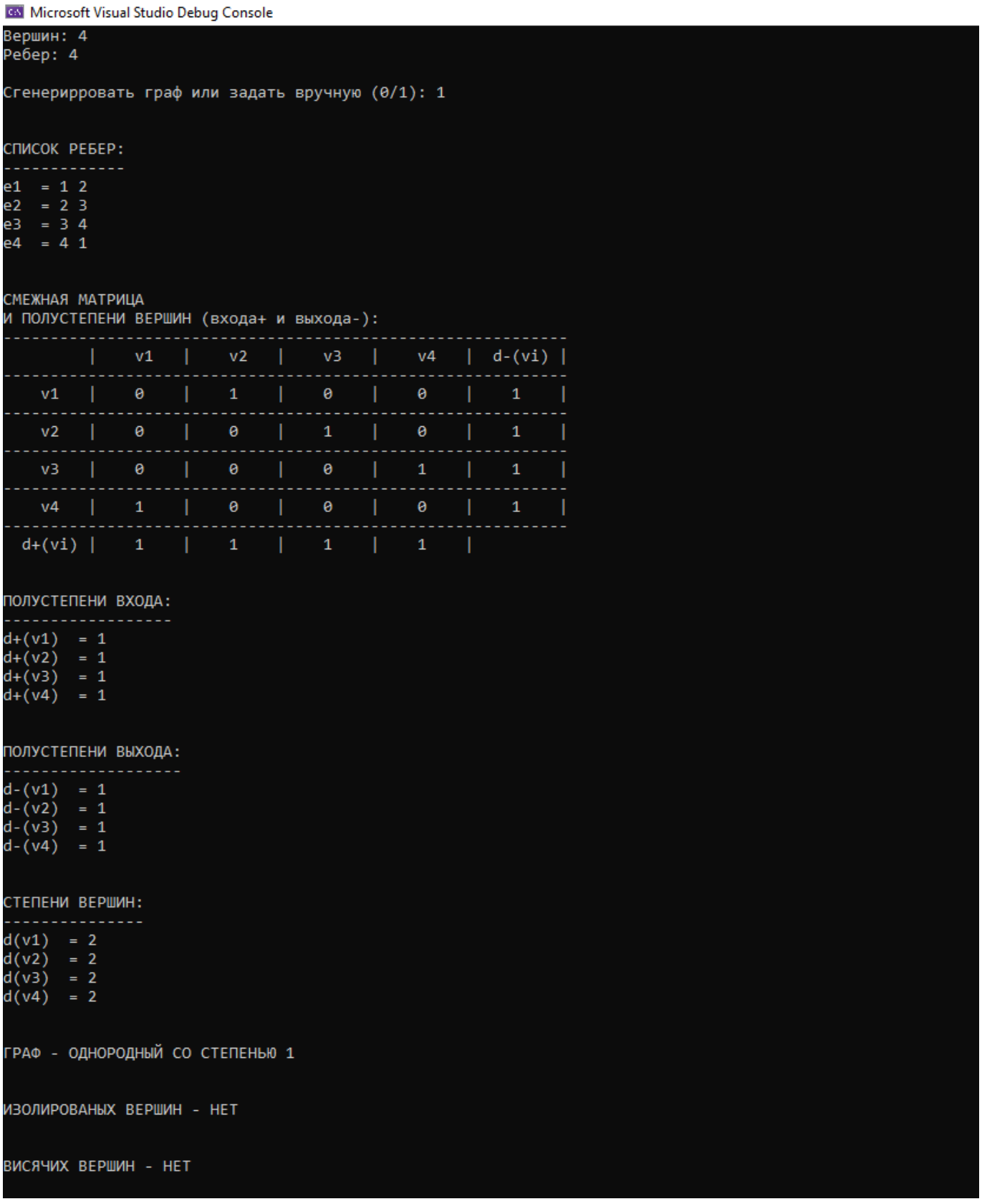

Рис. 2. Результат виконання програми при завданні графа вручну

ВИСНОВКИ

У ході виконання лабораторної роботи було розглянуто та вивчено різні типи представлення графів в пам’яті обчислювальних пристроїв (комп’ютерів).

Створено програмне забезпечення на мові програмування C++ для роботи з графами та визначення їх характеристик. Реалізована програма приймає від користувача граф, заданий списком ребер вручну: користувачем з клавіатури, або за допомогою рандомайзеру і будує для заданого графа матриці інцидентності та суміжності, визначає степінь, напівстепінь входу та виходу вершин, ізольовані та висячі вершини, однорідність графа та степінь його однорідності.