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 № 4
«Обхід 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: Обхід графів**.**

Мета: Дослідити роботу алгоритмів обходу в графах: пошуку вшир та пошуку вглиб.

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». *При реалізації алгоритмів вважати, що заданий граф є зв'язним.*

1. Обійти граф пошуком вшир. Користувач вводить початкову вершину графу. Програма виконує обхід графу, починаючи з вказаної початкової вершини. На екран виводиться протокол обходу – таблиця, яка містить наступні дані по кожній ітерації алгоритму обходу: поточна вершина, її BFS-номер, вміст черги (див. тему 28 електронного конспекту).
2. Обійти граф пошуком углиб. Аналогічно за пунктом 1 завдання, але програма виконує обхід графу пошуком углиб. На екран виводиться протокол обходу: поточна вершина, її DFS-номер, вміст стеку.

Код:

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

float n, m;
int a[20][20], sum[20][20];
int nodes[20]; // tops
int v;
string str;
char t;
queue<int> q;
bool* visited = new bool[20]; // for v_gub poisk

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';
        cout << "\n";
        break;
    case '1':
        cout << "\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);
            }
        }
    }
    cout << "Вершина, с которой начать обход: ";
    cin >> v;
}
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 << "\n\nСМЕЖНАЯ МАТРИЦА:\n";
    for (int i = 0; i < m; i++) {
        sum[a[i][0]][a[i][1]] = 1;
    }
    // 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 << "         |";
                }
                // 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) {
                // 10-99 numbs
                if (s + 1 > 9) {
                    cout << "   v" << s + 1 << "   |";
                }
                // 1-9 numbs
                else {
                    cout << "    v" << s + 1 << "   |";
                }
            }
            cout << "    " << sum[s + 1][k + 1] << "    |";
        }
        cout << "\n";
    }
}
void v_shir() {
    // firstly all tops = 0
    for (int i = 0; i < n; i++) {
        nodes[i] = 0;
    }
    cout << "\n\nОБХОД ВШИРИНУ: ";
    // put the first top to the q
    q.push(v);
    while (!q.empty()) {
        // while the q isn't empy, v-- in q
        int node = q.front();
        q.pop();
        // mark the top as visited
        nodes[node] = 2;
        // for checking all sum tops
        for (int j = 0; j < 7; j++) {
            // if top is sum and not visited
            if (sum[node][j + 1] == 1 && nodes[j + 1] == 0) {
                // v++ in q
                q.push(j + 1);
                // mark the top as visited
                nodes[j + 1] = 1;
            }
        }
        // print
        cout << node << " ";
    }
```

```
    cout << "\n";
}
// poisk v glub
void DFS(int v) {
    cout << v << " ";
    visited[v] = true;
    for (int r = 1; r <= n; r++) {
        if ((sum[v][r] != 0) && (!visited[r])) {
            DFS(r);
        }
    }
}
void v_glub() {
    for (int i = 1; i <= n; i++) {
        visited[i] = false;
    }
    // mass of visited tops
    bool* vis = new bool[20];
    cout << "\n\nПОИСК ВГЛУБИНУ: ";
    DFS(v);
    delete[]visited;
    cout << "\n\n";
}

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    generate();
    output();
    sumizhna_matr();

    v_shir();
    v_glub();
}
```

Результат:

```
Microsoft Visual Studio Debug Console
Вершин: 7
Ребер: 8

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

СПИСОК РЕБЕР:
------------
e1  = 2 1
e2  = 2 3
e3  = 2 4
e4  = 1 3
e5  = 1 7
e6  = 7 6
e7  = 6 5
e8  = 4 5


Вершина, с которой начать обход: 2


СМЕЖНАЯ МАТРИЦА:
```

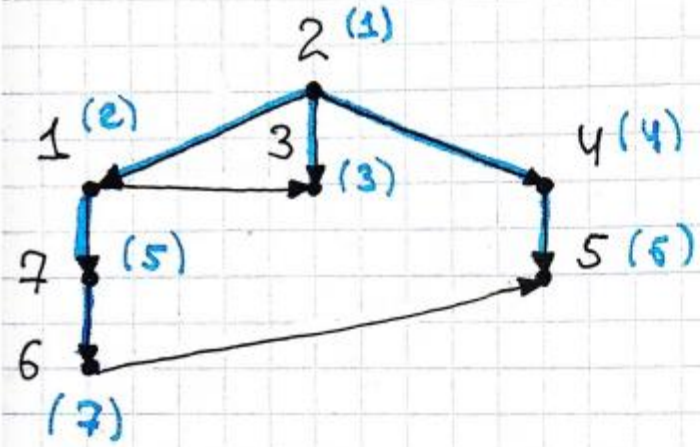
	v1	v2	v3	v4	v5	v6	v7
v1	0	0	1	0	0	0	1
v2	1	0	1	1	0	0	0
v3	0	0	0	0	0	0	0
v4	0	0	0	0	1	0	0
v5	0	0	0	0	0	0	0
v6	0	0	0	0	1	0	0
v7	0	0	0	0	0	1	0

```
ОБХОД ВШИРИНУ: 2 1 3 4 7 5 6


ПОИСК ВГЛУБИНУ: 2 1 3 7 6 5 4
```

Рис. 1. Результат виконання програми

Побудову таблиць протоколів обходу було виконано вручну. Для цього було взято за основу граф, характеристики якого зазначено на рис. 1.



Протокол обходу вшир:

V	BFS	Очеке
2	1	2
1	2	21
3	3	213
4	4	2134
–	–	134
7	5	1347
–	–	347
–	–	47
5	6	475
–	–	75
6	7	756

–	–	56
–	–	6
–	–	∅

Протокол обходу вглиб:

V	DFS	Stack
2	1	2
1	2	21
3	3	213
–	–	21
7	4	217
6	5	2176
5	6	21765
–	–	2176
–	–	217
–	–	21
–	–	2
4	7	24
–	–	2
–	–	∅

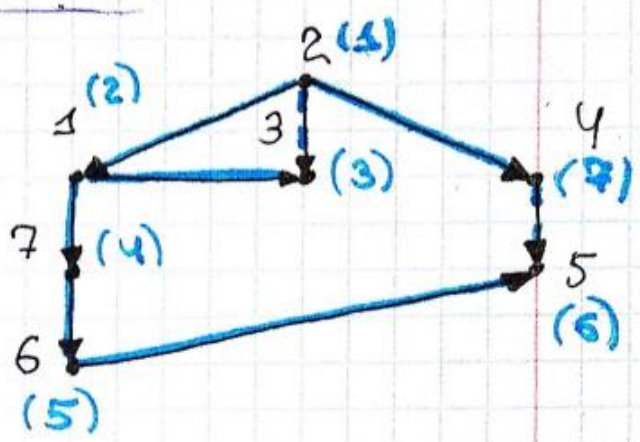


Рис. 2. Результат побудови протоколів обходу графа

ВИСНОВКИ

У ході виконання лабораторної роботи було розглянуто та вивчено різні обходу графів та методи їх представлення в пам’яті обчислювальних пристроїв (комп’ютерів) за допомогою черги та стеку.

Створено програмне забезпечення на мові програмування C++ для роботи з графами та знаходження порядку обходу графа вглиб та в ширину. Реалізована програма приймає від користувача граф, заданий списком ребер вручну: користувачем з клавіатури, або за допомогою рандомайзеру і будує для заданого графа матрицю суміжності та визначає порядок обходу графа два способами. Побудова таблиць протоколів обходу графів була виконана вручну, без застосування програмного забезпечення.