Министерство науки и высшего образования Российской федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

по лабораторной работе №7

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Обход графа в глубину»

Выполнил ст. группы 22ВВВ1:

Колобов И.О.

Приняли:

К.э.н., доцент Акифьев И. В.

К.т.н., доцент Юрова О. В.

Пенза 2023

**Цель работы:**

Цель научиться использовать алгоритм обхода графа в глубину ы

**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 обхода в глубину, реализованную в соответствии с приведенным выше описанием.

**3.**\* Реализуйте процедуру обхода в глубину для графа, представленного списками смежности.

**Задание 2\***

1. Для матричной формы представления графов выполните преобразование рекурсивной реализации обхода графа к не рекурсивной.

**Ход работы:**

Данный код реализует обход в глубину для графа, заданного матрицей смежности и списком смежности. Создается матрица смежности и список смежности на основе нее. Затем происходит обход в глубину для матрицы смежности и для списка смежности. Код также содержит проверку корректности заполнения списка смежности.

Алгоритм обхода в глубину можно реализовать как с помощью рекурсии, так и с использованием стека

1. С помощью рекурсии:

- Пометить текущую вершину как посещенную.

- Вывести значение текущей вершины.

- Рекурсивно вызвать алгоритм DFS для каждой не посещенной смежной вершины.

2. С использованием стека:

- Создать пустой стек и поместить в него начальную вершину.

- Пока стек не пуст:

- Извлечь вершину из вершины стека.

- Если вершина не была посещена:

- Пометить вершину как посещенную.

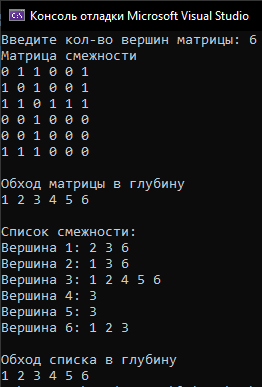
- Вывести значение вершины.

- Поместить в стек все не посещенные смежные вершины текущей вершины.

Оба подхода, рекурсивный и с использованием стека, дают одинаковый результат и реализуют обход графа в глубину.

**Результаты работы программы:**

Задание 1 и 2:



**Вывод:**

В ходе лабораторной работы научился работать с обходом графа в глубину для матрицы смежности и списка смежности, также научился преобразовывать рекурсивной реализацию обхода графа к не рекурсивной.

**Листинг:**

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <stack>

#include <vector>

#include <algorithm>

using namespace std;

// Обход в глубину для матрицы смежности

void DFSMatrix(vector<vector<int>>& matrix, int v, vector<bool>& visited) {

visited[v] = true;

cout << v + 1 << " ";

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

if (matrix[v][i] == 1 && !visited[i]) {

DFSMatrix(matrix, i, visited);

}

}

}

// Обход в глубину для списка смежности

void DFSList(int v, vector<vector<int>>& adj, vector<bool>& visited) {

visited[v] = true;

cout << v + 1 << " ";

for (int i = 0; i < adj[v].size(); ++i) {

int next = adj[v][i];

if (!visited[next]) {

DFSList(next, adj, visited);

}

}

}

int main() {

setlocale(LC\_ALL, "Russian");

srand(time(NULL));

int n;

cout << "Введите кол-во вершин матрицы: ";

cin >> n;

vector<vector<int>> matrix(n, vector<int>(n));

// Матрица 1

cout << "Матрица смежности " << endl;

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

for (int j = i; j < n; j++) {

if (i == j) {

matrix[i][j] = 0;

}

else {

matrix[i][j] = rand() % 2;

matrix[j][i] = matrix[i][j];

}

}

}

// Вывод матрицы 1

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

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

cout << matrix[i][j] << " ";

}

cout << endl;

}

cout << "\n";

vector<bool> visited(n, false);

// Обход матрицы в глубину

cout << "Обход матрицы в глубину\n";

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

int v = i;

if (!visited[v]) {

DFSMatrix(matrix, v, visited);

}

}

fill(visited.begin(), visited.end(), false); // сброс посещенных

vector<vector<int>> adj(n);

// Заполнение списка смежности на основе матрицы

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

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

if (matrix[i][j] == 1) {

adj[i].push\_back(j);

//adj[j].push\_back(i); // добавляем обратное ребро

}

}

}

// Вывод списка смежности

cout << "\nСписок смежности:\n";

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

cout << "Вершина " << i + 1 << ": ";

for (int j = 0; j < adj[i].size(); ++j) {

cout << adj[i][j] + 1 << " ";

}

cout << "\n";

}

// Обход в глубину для списка

cout << "Обход списка в глубину\n";

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

int v = i;

if (!visited[v]) {

DFSList(v, adj, visited);

}

}

return 0;

}

//Преобразование рекурсивной реализации обхода графа к не рекурсивной.

// Обход в глубину для матрицы смежности

/\*void DFSMatrix(vector<vector<int>>& matrix, int v, vector<bool>& visited) {

stack<int> stk; // Стек для хранения вершин

stk.push(v); // добавляем начальную вершину в стек

while (!stk.empty()) {

int current = stk.top(); // берем вершину на вершине стека

stk.pop(); // удаляем вершину из стека

if (!visited[current]) {

visited[current] = true; // помечаем вершину как посещенную

cout << current + 1 << " ";

for (int i = matrix.size() - 1; i >= 0; --i) {

if (matrix[current][i] == 1 && !visited[i]) {

stk.push(i); // добавляем смежные вершины в стек

}

}

}

}

}

// Обход в глубину для списка смежности

void DFSList(int v, vector<vector<int>>& adj, vector<bool>& visited) {

stack<int> stk; // Стек для хранения вершин

stk.push(v); // добавляем начальную вершину в стек

while (!stk.empty()) {

int current = stk.top(); // берем вершину на вершине стека

stk.pop(); // удаляем вершину из стека

if (!visited[current]) {

visited[current] = true; // помечаем вершину как посещенную

cout << current + 1 << " ";

for (int i = adj[current].size() - 1; i >= 0; --i) {

int next = adj[current][i];

if (!visited[next]) {

stk.push(next); // добавляем смежные вершины в стек

}

}

}

}

}

int main() {

setlocale(LC\_ALL, "Russian");

srand(time(NULL));

int n;

cout << "Введите кол-во вершин матрицы: ";

cin >> n;

vector<vector<int>> matrix(n, vector<int>(n));

// Матрица 1

cout << "Матрица смежности " << endl;

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

for (int j = i; j < n; j++) {

if (i == j) {

matrix[i][j] = 0;

}

else {

matrix[i][j] = rand() % 2;

matrix[j][i] = matrix[i][j];

}

}

}

// Вывод матрицы 1

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

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

cout << matrix[i][j] << " ";

}

cout << endl;

}

cout << "\n";

vector<bool> visited(n, false);

// Обход матрицы в глубину

cout << "Обход матрицы в глубину\n";

for (int i = 0; i < n; ++i) {s

int v = i;

if (!visited[v]) {

DFSMatrix(matrix, v, visited);

}

}

fill(visited.begin(), visited.end(), false); // сброс посещенных

vector<vector<int>> adj(n);

// Заполнение списка смежности на основе матрицы

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

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

if (matrix[i][j] == 1) {

adj[i].push\_back(j);

//adj[j].push\_back(i); // добавляем обратное ребро

}

}

}

// Вывод списка смежности

cout << "\nСписок смежности:\n";

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

cout << "Вершина " << i + 1 << ": ";

for (int j = 0; j < adj[i].size(); ++j) {

cout << adj[i][j] + 1 << " ";

}

cout << "\n";

}

// Обход в глубину для списка

cout << "Обход списка в глубину\n";

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

int v = i;

if (!visited[v]) {

DFSList(v, adj, visited);

}

}

return 0;

}\*/