Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Л и ОА в ИЗ»

на тему «Обход графа в глубину»

**Выполнили:**

студент группы 19ВВ3

Яцков В.В

**Приняли:**

Митрохин М. А.

Пенза 2020

**Название:**

Обход графа в глубину

**Цель работы:**

Научиться применять алгоритм рекурсивного обхода графа в глубину.

**Лабораторные задания:**

*Задание 1*

1. Сгенерируйте (используя генератор случайных чисел) две матрицу

смежности для неориентированного графа G. Выведите сгенерированные

матрицы 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.

**Описание метода решения задач:**

Для начала нужно сгенерировать матрицу смежности графа. Она выглядит, как симметричная относительно главной диагонали.

Реализация состоит из подготовительной части, в которой все вершины помечаются как не помеченные и осуществляется запуск процедуры обхода для вершин графа. И непосредственно процедуры обхода, которая помечает текущую (т.е. ту, в которой на текущей итерации находится алгоритм) вершину как посещенную.

Затем выводит номер текущей вершины на экран и в цикле просматривает v-ю строку матрицы смежности графа G(v,i). Как только алгоритм встречает смежную с v не посещенную вершину, то для этой вершины вызывается процедура обхода.

**Листинг:**

#include <iostream>

#include <conio.h>

#include <time.h>

#include <stack>

using namespace std;

int i, j, m, n;

bool\* visited = new bool[m];

int\*\* graph;

stack <int> s;

void DFS(int st)

{

int r;

printf("%d ", st + 1);

visited[st] = true;

for (r = 0; r <= m; r++)

if ((graph[st][r] == 1) && (visited[r] == false))

DFS(r);

}

void DFS\_1(int st) {

visited[st] = true;

s.push(st);

int v;

while (!s.empty()) {

st = s.top();

printf("%d ", st + 1);

s.pop();

for (i = m-1; i >= 0; i--) {

if ((graph[st][i] == 1) && (visited[i] == false)) {

s.push(i);

visited[i] = true;

}

}

}

}

void DFS\_main()

{

setlocale(LC\_ALL, "Rus");

printf("Введите количество вершин графа: ");

scanf\_s("%d", &m);

n = m;

graph = new int\* [m];

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

graph[i] = new int[m];

}

int start;

printf("Матрица смежности графа:\n");

srand(time(NULL));

for (i = 0; i < m; ++i)

for (j = i; j < m; ++j)

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

printf(" "); for (i = 0; i < m; ++i)

for (j = i; j < m; ++j) {

if (i == j) {

graph[i][j] = 0;

}

}

for (j = 0; j < m; j++)

{

printf("v%d ", j + 1);

}

printf("\n");

for (i = 0; i < m; ++i)

{

printf("v%d ", i + 1);

visited[i] = false;

for (j = 0; j < m; ++j)

printf("%d ", graph[i][j]);

printf("\n\n");

}

for (i = 0; i < m; i++)

{

printf("\n%d - ", i + 1);

for (j = 0; j < m; j++) {

if (graph[i][j] == 1) printf("%d ", j + 1);

}

}

printf("\nВведи вершину, с которой начать обход в глубину: ");

scanf\_s("%d", &start);

while (start > m) {

printf("\nВершина не существует!\n");

printf("\nВведи вершину, с которой начать обход в глубину: ");

scanf\_s("%d", &start);

}

printf("Порядок обхода (не рекурсивный): ");

DFS\_1(start - 1);

for (i = 0; i < m; i++) {

visited[i] = 0;

}

printf("\nПорядок обхода (рекурсивный): ");

DFS(start - 1);

\_getch();

}

int Menu(void)

{

system("CLS");

int c = 0;

setlocale(LC\_ALL, "Rus");

while (c < '0' || c > '1')

{

printf(

"|| 1 : Поиск в глубину ||\n"

"|| 0 : Выход из программы ||\n"

"> ");

c = \_getch();

system("cls");

}

return c;

}

void main(void)

{

int Select;

while ((Select = Menu()) != '0' && Select != 27)

switch (Select)

{

case '1':

DFS\_main();

break;

}

}

**Вывод:**

Научились применять алгоритм рекурсивного и не рекурсивного обхода графа в глубину на примере матричного представления графа, а также визуализировать его в виде обыкновенного графа.