Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе № 7

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Обход графа в глубину»

Выполнили:

студенты группы 22ВВВ2:

Зубриянова А.А.

Кондратьева В.И.

Приняли:

Акифьев И.В.

Митрохин М.А.

Пенза 2023

**Название**

Обход графа в глубину.

**Цель работы**

Научиться выполнять обход графа в глубину.

**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.

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 задачи**

Задание 1:

1. Сгенерировали (используя генератор случайных чисел) матрицу смежности для неориентированного графа G.

2. Используя рекурсивную функцию void DFS1(int s), осуществили процедуру обхода в глубину: если в матрице есть ребро и мы в нем ещё не были – заходим в вершину, помечаем её и продолжаем обход.

3. По аналогичному алгоритму осуществили процедуру обхода в глубину через списки смежности, используя рекурсию.

Задание 2:

1. Для матричной формы представления графов выполнили преобразование рекурсивной реализации обхода графа к не рекурсивной с использованием стека. Для этого использовали функции: S.push(s), S.top(), S.pop() и S.empty().

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <windows.h>

#include <time.h>

#include <stdlib.h>

#include <locale.h>

#include <conio.h>

#include <iostream>

#include <stack>

using namespace std;

stack <int> S;

int k;

int size1, h;

int\*\* arr1;

int\* vis;

void print\_vis();

struct Node

{

int n; //Номер вершины (0..N-1)

Node\* next;

};

Node\* LA; //Массив структур - списков смежности [N]

Node\* make\_LA()

{

Node\* la, \* p;

la = (Node\*)malloc(sizeof(Node) \* size1);

for (int i = 0; i < size1; i++)

{

la[i].n = i;

la[i].next = NULL;

p = &la[i];

for (int j = 0; j < size1; j++)

{

if (arr1[i][j] == 1)

{

p->next = (Node\*)malloc(sizeof(Node));

p = p->next;

p->n = j;

p->next = NULL;

}

}

}

return la;

}

void print\_LA()

{

Node\* p;

for (int i = 0; i < size1; i++)

{

p = &LA[i];

while (p != NULL)

{

printf("%d > ", p->n);

p = p->next;

}

printf("\n");

}

}

void first\_list(int l)

{

Node\* p;

p = &LA[l];

vis[l] = 1;

printf("%5d", l);

while (p->next != NULL)

{

if (vis[p->next->n] == 0)

{

vis[p->next->n] = 1;

l = p->next->n;

first\_list(l);

}

p = p->next;

}

}

void create\_matrix() {

for (int i = 0; i < size1; i++) {

for (int j = i; j < size1; j++) {

if (i == j) { //главная диагональ == 0

arr1[i][j] = 0;

}

else { //ниже главной диагонали отразить то, что выше

arr1[i][j] = rand() % 2;

arr1[j][i] = arr1[i][j];

}

}

}

}

void print\_matrix() {

printf(" ");

for (int i = 0; i < size1; i++) {

printf("%4d ", i);

}

printf("\n");

printf("-----------------------------------------------------------\n");

for (int i = 0; i < size1; i++) {

printf("%4d|", i);

for (int j = 0; j < size1; j++) {

printf("%4d ", arr1[i][j]);

}

printf("\n");

}

printf("\n");

}

void print\_vis() {

printf("\nМассив vis:\n");

for (int i = 0; i < size1; i++) {

printf("%7d", vis[i]);

}

printf("\n\n");

}

void DFS1(int s) {

vis[s] = 1;

printf("%5d", s);

for (int i = 0; i < size1; i++) {

if ((arr1[s][i] == 1) && (vis[i] == 0)) {

DFS1(i);

}

}

}

void DES2(int s) {

vis[s] = 1;

printf("%5d", s);

S.push(s);

while (!S.empty()) {

s = S.top();

S.pop();

for (int i = size1 - 1; i >= 0; i--) {

if ((arr1[s][i] == 1) && (vis[i] == 0)) {

printf("%5d", i);

vis[i] = 1;

S.push(i);

}

}

}

}

void clean\_vis() {

for (int i = 0; i < size1; i++) {

vis[i] = 0;

}

}

int main() {

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

printf("Задание 1\n");

printf("Введите размер матрицы смежности M1: ");

scanf("%d", &size1);

printf("\n");

vis = (int\*)malloc(sizeof(int) \* size1);

arr1 = (int\*\*)malloc(sizeof(int\*) \* size1);

for (int i = 0; i < size1; i++) {

arr1[i] = (int\*)malloc(sizeof(int) \* size1);

}

printf("Матрица M1:\n");

create\_matrix();

print\_matrix();

clean\_vis();

printf("=======================================================================\n\n");

printf("Задание 2\n");

printf("Введите вершину (от 0 до %d) для начала обхода: ", size1 - 1);

scanf("%d", &h);

int l = h;

k = h;

printf("\n");

printf("я был в вершинах:");

DFS1(h);

printf("\n");

clean\_vis();

printf("\n");

printf("=======================================================================\n\n");

printf("Задание 3\n");

printf("Список смежности S1:\n");

LA = make\_LA();

print\_LA();

printf("\n");

printf("я был в вершинах:");

first\_list(l);

printf("\n");

printf("=======================================================================\n\n");

printf("Задание 4\n");

clean\_vis();

printf("я был в вершинах:");

DES2(k);

for (int i = 0; i < size1; ++i) free(arr1[i]);

free(arr1);

free(vis);

\_getch();

return 0;

}

**Результаты работы программы**



Рисунок 1 – Обход графа в глубину: рекурсивная реализация, с помощью списков смежности и через стек

**Вывод**

В ходе выполнения лабораторной работы была разработана программа, выполняющая обход графа в глубину. Результаты работы программы совпали с ожидаемыми результатами, следовательно, программа работает без ошибок.

Получили опыт в создании проектов в среде Microsoft Visual Studio, приобрели навыки программирования алгоритмов.