Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Обход графа в глубину»

Выполнили студенты группы 20ВВ3:

Мальков И.А.

Педай Н.Д.

Приняли:

Митрохин М.А.

Юрова О.В.

Пенза 2021

**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:**

1. Код программы

#include <stdio.h>

#include <iostream>

#include <locale.h>

#include <stdlib.h>

#include <time.h>

#include <stack> // стек

using namespace std;

typedef struct List {

int Data;

struct List\* next;

}List;

List\* init(int num)

{

List\* node = (List\*)malloc(sizeof(List));

node->Data = num;

node->next = NULL;

return node;

}

List\* add(List\* lst, int num)

{

struct List\* temp, \* p;

temp = (List\*)malloc(sizeof(List));

p = lst->next;

lst->next = temp;

temp->Data = num;

temp->next = p;

return temp;

}

void initialize(int n, List\*\* vrt)

{

int i;

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

{

vrt[i] = init(i);

}

}

void list(int n, int i, int\*\* A1, List\* lst)

{

int j;

for (j = n; j >= 0; j--)

{

if (A1[i][j] == 1)

add(lst, j);

}

}

void result(List\*\* vrt, int\*\* A1, int n)

{

int i;

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

{

list(n, i, A1, vrt[i]);

}

}

void print(List\*\* vrt, int n)

{

int i, j;

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

{

cout << "\n";

List\* x = vrt[i];

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

{

if (vrt[i] != NULL)

{

cout << vrt[i]->Data<< " ";

vrt[i] = vrt[i]->next;

}

}

vrt[i] = x;

}

cout << "\n";

}

void DFS\_list(bool\* A2, int n, int i, List\*\* vrt)

{

int j;

A2[i] = true;

cout << i << " ";

while (vrt[i]->next != NULL)

{

vrt[i] = vrt[i]->next;

j = vrt[i]->Data;

if (A2[j] == false)

DFS\_list(A2, n, j, vrt);

}

}

void steck(int\*\* A1, int n, int i)

{

stack<int> Stack;

int\* nodes; // вершины графа

nodes = (int\*)malloc(n \* sizeof(int\*));

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

nodes[i] = 0; // исходно все вершины равны 0

int a;

cout << "Enter vertex: ";

cin >> a;

Stack.push(a); // помещаем в очередь вершину

cout << "DFS stack result: ";

while (!Stack.empty()) // пока стек не пуст

{

int node = Stack.top(); // извлекаем вершину

Stack.pop();

if (nodes[node] == 2) continue;

nodes[node] = 2; // отмечаем ее как посещенную

for (int j = n - 1; j >= 0; j--) // проверяем для нее все смежные вершины

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

{

if (A1[node][j] == 1 && nodes[j] != 2) // если вершина смежная и не обнаружена

{

Stack.push(j); // добавляем ее в cтек

nodes[j] = 1; // отмечаем вершину как обнаруженную

}

}

cout << node << " "; // выводим номер вершины

}

cout << "\n";

cin.get();

}

int\*\* create(int n)

{

int\*\* A1, k = 1;

A1 = (int\*\*)malloc(n \* sizeof(int\*));

srand(time(NULL));

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

{

A1[i] = (int\*)malloc(n \* sizeof(int));

A1[i][i] = 0;

for (int j = k; j < n; j++)

{

if (rand() % 100 > 50)

{

A1[i][j] = 0;

}

else {

A1[i][j] = 1;

}

}

k++;

}

k = 1;

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

{

for (int j = k; j < n; j++)

{

A1[j][i] = A1[i][j];

}

k++;

}

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

cout << "V" << i << " ";

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

{

cout << "\n";

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

{

cout << A1[i][j] << " ";

}

}

return A1;

}

void DFS(bool\* A2, int n, int i, int\*\* A1)

{

A2[i] = true;

cout << i << " ";

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

{

if (A1[i][j] == 1 && A2[j] == false)

{

DFS(A2, n, j, A1);

}

}

}

int main()

{

int n, \*\* t, s, o;

cout << "Enter array size: ";

cin >> n;

List\*\* vrt = (List\*\*)malloc(n \* sizeof(List\*));

bool\* A2 = (bool\*)malloc(n \* sizeof(bool));

t = create(n);

cout << "\n";

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

A2[i] = false;

cout << "enter vertex: ";

cin >> s;

cout << "DFS result: ";

DFS(A2, n, s, t);

cout << "\n";

steck(t, n, 0);

initialize(n, vrt);

result(vrt, t, n);

print(vrt, n);

cout << "enter vertex: ";

cin >> o;

cout << "DFS list result: ";

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

A2[i] = false;

DFS\_list(A2, n, o, vrt);

cout << "\n";

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

free(t[i]);

free(t);

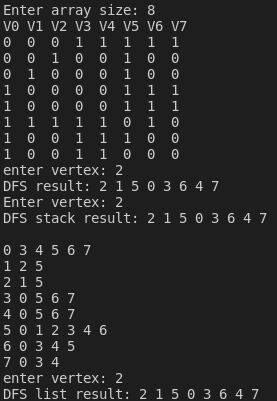
free(A2);

//system("pause");

return 0;

}

1. Результат работы программы.

Вывод: в ходе выполнения данной лабораторной работы разработали программу, согласно заданию.