Министерство науки и высшего 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 22ВВВ1:

Кирилин М. С.

Николаев А. А.

Приняли:

К.т.н, доцент Акифьев И. В.

К.т.н, доцент Юрова О. В.

Пенза 2023

**Цель работы**

Научится совершать обход графа в глубину. Реализовать функции обхода графа, представленного списками смежности, обход в глубину без рекурсии

**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**

**Теоретическая часть**

Обход графа – одна из наиболее распространенных операций с графами. Задачей обхода является прохождение всех вершин в графе. Обходы применяются для поиска информации, хранящейся в узлах графа, нахождения связей между вершинами или группами вершин и т.д.

Одним из способов обхода графов является поиск в глубину. Идея такого обхода состоит в том, чтобы начав обход из какой-либо вершины всегда переходить по первой встречающейся в процессе обхода связи в следующую вершину, пока существует такая возможность. Как только в процессе обхода исчерпаются возможности прохода, необходимо вернуться на один шаг назад и найти следующий вариант продвижения. Таким образом, итерационно выполняя описанные операции, будут пройдены все доступные для прохождения вершины. Чтобы не заходить повторно в уже пройденные вершины, необходимо их пометить как пройденные.

**Практическая часть**

Основное задание:

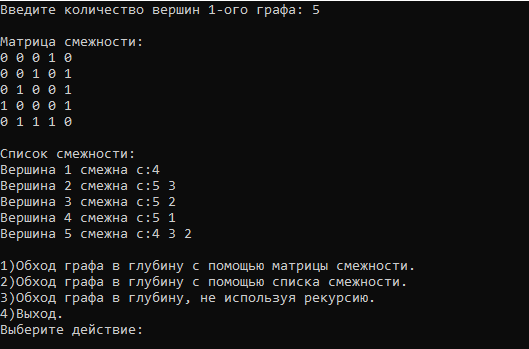


Рисунок 1- создали матрицу и списки смежности

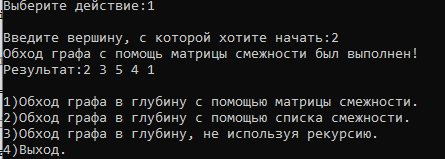


Рисунок 2- обход в глубину с помощью матрицы смежности

Дополнительное задание:

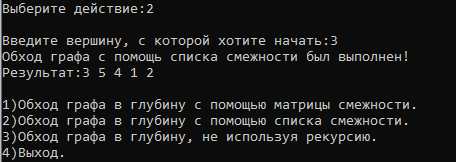


Рисунок 3- обход в глубину с помощью списка смежности

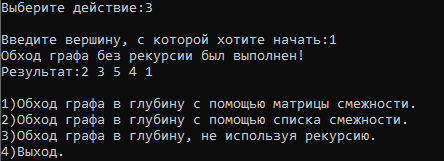


Рисунок 4- обход графа в глубину, не используя рекурсию

**Вывод**: В ходе данной лабораторной работы мы научились совершать обход графа в глубину. Создали матрицу и списки смежности. Реализовали функции обхода в глубину с помощью матрицы смежности; обхода в глубину с помощью списка смежности; обхода графа в глубину, не используя рекурсию

**Приложение А  
Листинг**

**Файл Source.cpp**

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <cstdio>

using namespace std;

typedef struct Spis {

int inf;

struct Spis\* next;

}SP;

int i, j, ver;

void BypassMatrix(int\*\* smej,int ver1, int\* visit, int ver )

{

visit[ver1] = 1; // Помечаем текущую вершину как посещенную

cout << ver1 + 1 << " ";

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

{

if (smej[ver1][i] == 1 && visit[i] !=1)

{

BypassMatrix(smej, i, visit, ver);

}

}

}

void BypassSpisok(SP\*\* MassSpis, int ver2, int\* visit)

{

visit[ver2] = 1; // Помечаем текущую вершину как посещенную

cout << ver2 + 1 << " ";

SP\* current = MassSpis[ver2];

while (current != NULL)

{

int temp = current->inf;

if (!visit[--temp])

{

BypassSpisok(MassSpis, temp, visit);

}

current = current->next;

}

}

void BypassNoRek(int\*\* smej, int\* visit, int ver3, int ver)

{

int\* stack = new int[ver]; // Стек

int top = -1; // Вершина стека

stack[++top] = ver3; // Помещаем начальную вершину в стек

while (top != -1) {

int temp = stack[top--]; // Извлекаем вершину из стека

if (visit[temp] != 1)

{

visit[temp] = 1; // Помечаем вершину как посещенную

printf("%d ", temp + 1); // Выводим вершину

for (int i = ver - 1; i >= 0; i--)

{

if (smej[temp][i] == 1 && visit[i] != 1)

{

stack[++top] = i; // Помещаем смежную непосещенную вершину в стек

}

}

}

}

delete[] stack;

}

void main()

{

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int\*\* smej = 0; //Матрица смежности

int choice, ver1, ver2, ver3;

SP\*\* MassSpis = NULL;//Массив списков смежности

int\* visit = 0;

bool prov1 = 1, prov2=1, prov3=1;

cout << "Введите количество вершин 1-ого графа: ";

cin >> ver;

cout << endl;

//Выделение памяти под массив смежности

smej = new int\* [ver];

if (smej == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

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

{

smej[i] = new int[ver];

}

//Выделение памяти под массив списков

MassSpis = new SP\* [ver];

if (MassSpis == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

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

{

MassSpis[i] = NULL;

}

//Генерация массива смежности

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

{

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

{

if (i == j)

{

smej[i][j] = 0; // на главной диагонали нули

}

else

{

smej[i][j] = rand() % 2; // случайные значения 0 или 1

smej[j][i] = smej[i][j]; // симметрично заполнять значения для неориентированного графа

}

}

}

//Генерация списка смежности

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

{

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

{

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

{

SP\* newSP = new SP;

newSP->inf = j + 1;

newSP->next = MassSpis[i];

MassSpis[i] = newSP;

}

}

}

//Вывод матрицы смежности

cout << "Матрица смежности:" << endl;

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

{

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

{

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

}

cout << endl;

}

cout << endl;

//Вывод списка смежности

cout << "Список смежности:" << endl;

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

{

if (MassSpis[i] == NULL)

{

cout << "Вершина " << i + 1 << " не имеет смежных";

}

else

{

cout << "Вершина " << i + 1 << " смежна с:";

SP\* current = MassSpis[i];

while (current != NULL)

{

cout << current->inf << " ";

current = current->next;

}

}

cout << endl;

}

do

{

cout << endl;

cout << "1)Обход графа в глубину с помощью матрицы смежности." << endl;

cout << "2)Обход графа в глубину с помощью списка смежности." << endl;

cout << "3)Обход графа в глубину, не используя рекурсию." << endl;

cout << "4)Выход." << endl;

cout << "Выберите действие:";

cin >> choice;

switch (choice)

{

case 1:

visit = new int[ver];

if (visit == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

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

{

visit[i] = 0;

}

do

{

cout << endl;

cout << "Введите вершину, с которой хотите начать:";

cin >> ver1;

if (ver1 < 1 || ver1 > ver)

printf("Некорректный номер вершины! Повторите попытку.\n");

else

prov1 = 0;

} while (prov1);

cout << "Обход графа с помощь матрицы смежности был выполнен!" << endl;

cout << "Результат:";

BypassMatrix(smej, ver1 - 1, visit, ver);

cout << endl;

delete[] visit;

break;

case 2:

visit = new int[ver];

if (visit == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

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

{

visit[i] = 0;

}

do

{

cout << endl;

cout << "Введите вершину, с которой хотите начать:";

cin >> ver2;

if (ver2 < 1 || ver2 > ver)

printf("Некорректный номер вершины! Повторите попытку.\n");

else

prov2 = 0;

} while (prov2);

cout << "Обход графа с помощь списка смежности был выполнен!" << endl;

cout << "Результат:";

BypassSpisok(MassSpis, ver2 - 1, visit);

cout << endl;

delete[] visit;

break;

case 3:

visit = new int[ver];

if (visit == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

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

{

visit[i] = 0;

}

do

{

cout << endl;

cout << "Введите вершину, с которой хотите начать:";

cin >> ver3;

if (ver3 < 1 || ver3 > ver)

printf("Некорректный номер вершины! Повторите попытку.\n");

else

prov3 = 0;

} while (prov3);

cout << "Обход графа без рекурсии был выполнен!" << endl;

cout << "Результат:";

BypassNoRek(smej,visit,ver3,ver);

cout << endl;

delete[] visit;

break;

case 4:

system("cls");

cout << "До свидания!";

break;

default:

cout << endl;

cout << "Неверный выбор! Повторите попытку.";

cout << endl;

break;

}

} while (choice != 4);

delete[] MassSpis;

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

{

delete[] smej[i];

}

delete[] smej;

}