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:

Ганин И.Р.

Курушин Я.С.

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:

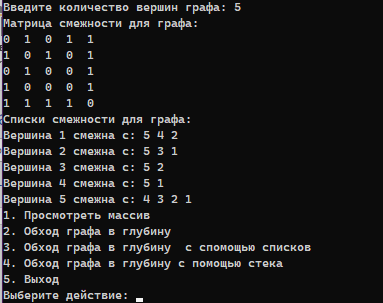


Рисунок 1- создали матрицу смежности и списки смежности

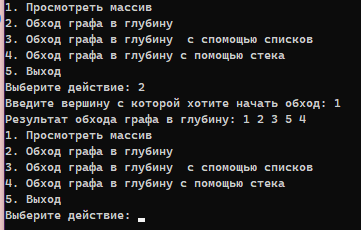


Рисунок 2- сделали обход в глубину с 1 вершины

Дополнительное задание:

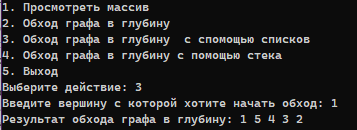


Рисунок 3- обход в глубину по спискам смежности

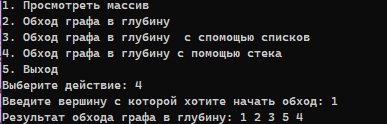


Рисунок 4- обход в глубину с помощью стека

**Вывод**: В ходе данной лабораторной работе научились совершать обход графа в глубину. Реализовали функции обхода графа, представленного списками смежности; обход в глубину без рекурсии

**Приложение А  
Листинг**

**Файл Source.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

void main() {

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int\*\* arr = 0; // матрица смежности

int\*\* incidenceMatrix = 0; // Матрица инцидентности

int i, j, rows, k, unit = 0;

printf("Введите количество вершин графа: ");

scanf("%d", &rows);

arr = (int\*\*)malloc(rows \* sizeof(int\*));

if (arr == NULL) {

printf("Не удалось выделить память!\n");

return;

}

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

arr[i] = (int\*)malloc(rows \* sizeof(int));

}

// генерировать случайные значения для матрицы смежности

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

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

if (i == j) {

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

}

else {

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

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

}

}

}

printf("Матрица смежности:\n");

// выводить матрицу смежности на экран

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

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

printf("%-2d ", arr[i][j]);

}

printf("\n");

}

printf("Размер графа равен: %d\n", rows);

int edgeCount = 0;

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

for (i = j + 1; i < rows; i++) {

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

edgeCount++;

}

}

}

int\* rArr = 0;

rArr = (int\*)malloc(rows \* sizeof(int\*));

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

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

if (arr[i][j] == 0) {

unit = unit;

}

else {

unit++;

}

}

rArr[i] = unit;

//printf("%d\n", rArr[i]);

unit = 0;

}

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

if (rArr[i] == 1) {

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

}

else if (rArr[i] == rows - 1) {

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

}

else if (rArr[i] == 0) {

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

}

}

unit = edgeCount;

incidenceMatrix = (int\*\*)malloc(rows \* sizeof(int\*));

if (incidenceMatrix == NULL) {

printf("Не удалось выделить память!\n");

return;

}

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

incidenceMatrix[i] = (int\*)malloc(edgeCount \* sizeof(int));

}

// Построение матрицы инцидентности на основе матрицы смежности

edgeCount = 0;

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

for (j = i + 1; j < rows; j++) {

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

for (k = 0; k < rows; k++) {

if (k == i || k == j) {

incidenceMatrix[k][edgeCount] = 1;

}

else {

incidenceMatrix[k][edgeCount] = 0;

}

}

edgeCount++;

}

}

}

// Вывод матрицы инцидентности на экран

printf("Матрица инцидентности:\n");

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

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

printf("%-2d ", incidenceMatrix[i][j]);

}

printf("\n");

}

int\* resultArr;

int p = 0;

resultArr = (int\*)malloc(rows \* sizeof(int\*));

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

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

if (incidenceMatrix[i][j] == 0) {

p = p;

}

else {

p++;

}

}

resultArr[i] = p;

// printf("%d\n", resultArr[i]);

p = 0;

}

printf("Размер графа равен: %d\n", rows);

p = 0;

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

if (resultArr[i] == 1) {

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

}

else if (resultArr[i] == rows - 1) {

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

}

else if (resultArr[i] == 0) {

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

}

}

// Освобождение памяти

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

free(arr[i]);

}

free(arr);

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

free(incidenceMatrix[i]);

}

free(incidenceMatrix);

}