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:

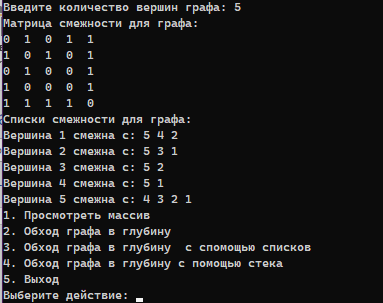


Рисунок 1- создали матрицу смежности и списки смежности

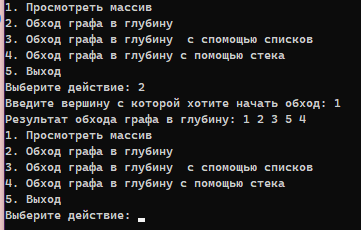


Рисунок 2- сделали обход в глубину с 1 вершины

Дополнительное задание:

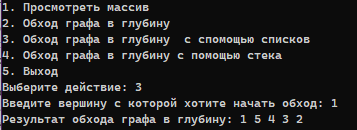


Рисунок 3- обход в глубину по спискам смежности

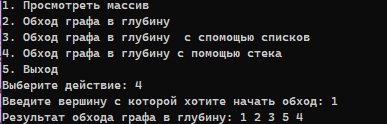


Рисунок 4- обход в глубину с помощью стека

**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>

#include <string.h>

typedef struct Node {

int inf;

struct Node\* next;

}Node;

void addSp(Node\*\* arr\_sp, int\*\* arr, int rows) {

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

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

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

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->inf = j + 1;

newNode->next = arr\_sp[i];

arr\_sp[i] = newNode;

}

}

}

}

void printSp(Node\*\* arr\_sp, int rows) {

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

printf("Вершина %d смежна с: ", i + 1);

Node\* current = arr\_sp[i];

while (current != NULL) {

printf("%d ", current->inf);

current = current->next;

}

printf("\n");

}

}

void cross(int\*\* arr, int rows1, int\* arr\_visited, int rows) {

arr\_visited[rows1] = 1; // Помечаем текущую вершину как посещенную

printf("%d ", rows1 + 1);

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

if (arr[rows1][i] == 1 && arr\_visited[i] != 1) {

cross(arr, i, arr\_visited, rows);

}

}

}

void crossSp(Node\*\* arr\_sp, int rows1, int\* arr\_visited) {

arr\_visited[rows1] = 1; // Помечаем текущую вершину как посещенную

printf("%d ", rows1 + 1);

Node\* current = arr\_sp[rows1];

while (current != NULL) {

int temp = current->inf;

if (!arr\_visited[--temp]) {

crossSp(arr\_sp, temp, arr\_visited);

}

current = current->next;

}

}

void crossStack(int\*\* arr, int\* arr\_visited, int rows1, int rows) {

int\* stack = (int\*)malloc(rows \* sizeof(int)); // Стек

int top = -1; // Вершина стека

stack[++top] = rows1; // Помещаем начальную вершину в стек

while (top != -1) {

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

if (arr\_visited[temp] != 1) {

arr\_visited[temp] = 1; // Помечаем вершину как посещенную

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

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

if (arr[temp][i] == 1 && arr\_visited[i] != 1) {

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

}

}

}

}

}

void main() {

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int\*\* arr = 0; // матрица смежности

int\* arr\_visited = 0; // матрица посещённых вершин

int i, j, rows;

printf("Введите количество вершин графа: ");

scanf("%d", &rows);

Node\*\* arr\_sp = NULL;

// Выделение памяти для массива списков смежности

arr\_sp = (Node\*\*)malloc(rows \* sizeof(Node\*));

if (arr\_sp == NULL) {

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

return;

}

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

arr\_sp[i] = NULL;

}

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));

}

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

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

arr\_visited[i] = 0; // Инициализируем все вершины как непосещенные

}

// 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

}

}

}

// выводить матрицу смежности на экран

printf("Матрица смежности для графа:\n");

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

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

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

}

printf("\n");

}

printf("Списки смежности для графа:\n");

addSp(arr\_sp, arr, rows);

printSp(arr\_sp, rows);

int choice, rows1 = 0;

do {

printf("1. Просмотреть массив\n");

printf("2. Обход графа в глубину\n");

printf("3. Обход графа в глубину с спомощью списков\n");

printf("4. Обход графа в глубину с помощью стека\n");

printf("5. Выход\n");

printf("Выберите действие: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Матрица смежности для графа:\n");

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

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

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

}

printf("\n");

}

printSp(arr\_sp, rows);

break;

case 2:

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

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

arr\_visited[i] = 0; // Инициализируем все вершины как непосещенные

}

printf("Введите вершину с которой хотите начать обход: ");

scanf("%d", &rows1);

if (rows1 < 1 || rows1 > rows) {

printf("Некорректный номер вершины!\n");

break;

}

printf("Результат обхода графа в глубину: ");

cross(arr, rows1 - 1, arr\_visited, rows);

printf("\n");

free(arr\_visited);

break;

case 3:

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

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

arr\_visited[i] = 0;

}

printf("Введите вершину с которой хотите начать обход: ");

scanf("%d", &rows1);

if (rows1 < 1 || rows1 > rows) {

printf("Некорректный номер вершины!\n");

break;

}

printf("Результат обхода графа в глубину: ");

memset(arr\_visited, 0, sizeof(int) \* rows);

crossSp(arr\_sp, rows1 - 1, arr\_visited);

printf("\n");

free(arr\_visited);

break;

case 4:

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

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

arr\_visited[i] = 0;

}

printf("Введите вершину с которой хотите начать обход: ");

scanf("%d", &rows1);

if (rows1 < 1 || rows1 > rows) {

printf("Некорректный номер вершины!\n");

break;

}

printf("Результат обхода графа в глубину: ");

crossStack(arr, arr\_visited, rows1 - 1, rows);

printf("\n");

free(arr\_visited);

break;

case 5:

system("cls");

printf("До свидания!\n");

break;

default:

printf("Некорректный выбор!\n");

break;

}

} while (choice != 5);

// Освобождение памяти

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

free(arr[i]);

}

free(arr);

}