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 24ВВВ3
Азаров М.С.
Кукушкин А.А.

Приняли:
к.т.н., доцент Юрова О.В.
к.т.н., Деев М.В.

Пенза 2025

Общие сведения.

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.

Таким образом, можно предложить следующую рекурсивную реализацию алгоритма обхода в глубину.

Вход: G – матрица смежности графа.

Выход: номера вершин в порядке их прохождения на экране.

Алгоритм ПОГ

1.1. для всех i положим NUM[i] = False пометим как "не посещенную";

1.2. **ПОКА** существует "новая" вершина v

1.3. **ВЫПОЛНЯТЬ** DFS (v).

Алгоритм DFS(v):

2.1. пометить v как "посещенную" NUM[v] = True;

2.2. вывести на экран v;

2.3. **ДЛЯ** i = 1 **ДО** size_G **ВЫПОЛНЯТЬ**

2.4. **ЕСЛИ** G(v,i) = = 1**И** NUM[i] = = False

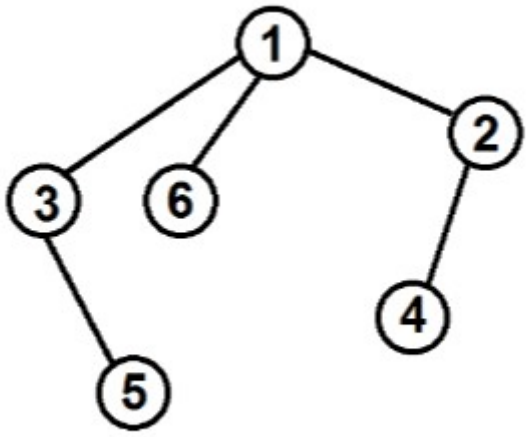
2.5. **ТО**

2.6. {

2.7. DFS(i);

2.8. }

Например, пусть дан граф (рисунок 1), заданный в виде матрицы смежности:



$$G = \begin{Bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix}$$

Рисунок 1 – Граф

Тогда, если мы начнем обход из первой вершины, то в конце работы алгоритма все вершины будут посещены. А на экран будут выведены номера вершин в порядке их посещения алгоритмом.

$$NUM = \{True \quad True \quad True \quad True \quad True \quad True\}$$

Рисунок 2 – Результат работы обхода

Цель работы – Освоить алгоритм обхода графа в глубину (DFS) на практике, научившись генерировать неориентированный граф, представлять его в виде матрицы смежности и списков смежности, реализовывать рекурсивный и нерекурсивный обход в глубину для разных форм представления графа.

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.

Код программы

C

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>


void DFS(int** G, int numG, int* visited, int current) {
	visited[current] = 1;
	printf("%3d", current);

	for (int i = 0; i < numG; i++) {
		if (G[current][i] == 1 && visited[i] == 0) {
			DFS(G, numG, visited, i);
		}
	}
}


int main() {
	int** G;
	int numG, current;
	int* visited;

	printf("Input number of vershini: ");
	scanf("%d", &numG);
	visited = (int*)malloc(numG * sizeof(int));
	G = (int**)malloc(numG * sizeof(int*));
```

```
	for (int i = 0; i < numG; i++)
		G[i] = (int*)malloc(numG * sizeof(int));

	srand(time(NULL));

	for (int i = 0; i < numG; i++) {
		visited[i] = 0;
		for (int j = i; j < numG; j++) {
			G[i][j] = G[j][i] =  (i == j ? 0 : rand() % 2);

		}
	}

	for (int i = 0; i < numG; i++) {
		for (int j = 0; j < numG; j++) {
			printf("%3d", G[i][j]);

		}
		printf("\n");
	}

	printf("input start vershinka: ");
	scanf("%d", &current);


	printf("Path: ");
	DFS(G, numG, visited, current);
	free(visited);
	for (int i = 0; i < numG; i++)
		free(G[i]);
	free(G);

	return 0;

}
```

Результаты работы программы

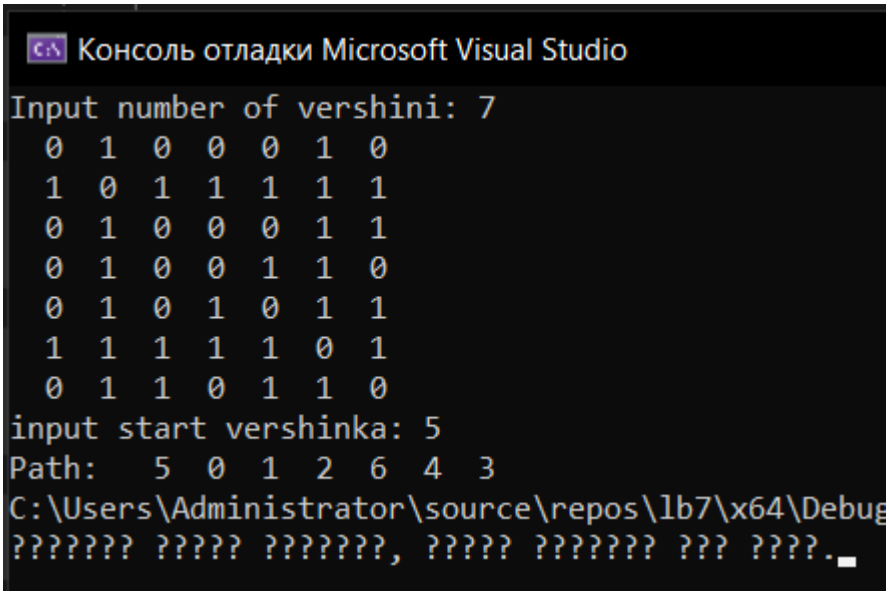


Рисунок 3 - Результат работы программы

Вывод: В ходе выполнения лабораторной работы был освоен алгоритм обхода графа в глубину (DFS) и приобретены практические навыки работы с различными представлениями графов.