**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8
# Обработка деревьев

## Вариант 2

Студент Шатохина Т.П.

Группа ИУ7 – 31Б

Преподаватель Барышникова М.Ю.

*2023 г.*

## *Условие задачи*

Определить, является ли связным заданный граф.

Пункты меню:

1 - ввести матрицу смежности

2 - считать матрицу смежности из файла

3 - вывести матрицу смежности

4 - вывести граф

5 - определить является ли граф связным

0 - выход из программы

## *Техническое задание*

1) **Входные данные*:*** целые числа.

2) **Выходные данные**: в зависимости от пункта меню программы

1. и 2. Нет выходных данных

3. Таблица смежности

4. Граф в формате png

5. Сообщение о том связен граф или нет

0. Сообщение «Программа успешно завершена.»

## *Аварийные ситуации:*

1) Выбор несуществующего пункта меню

2) Ошибка выделения памяти

3) Неверное количество вершин графа

4) Неверная длина пути между вершинами графа

## *Описание структур данных*

*Структура для реализации графа:*

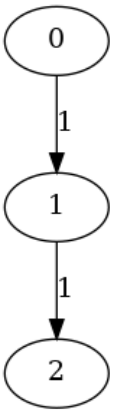
```
typedef struct graph_struct graph_struct_t;

struct graph_struct
{
    int size;               // количество вершин в графе

    int **matrix;           // матрица смежности
    int **reverse_matrix;    // обратная матрица смежности
};
```

Структура для реализации очереди, используемой в поиске в ширину:

```
typedef struct queue_node_struct queue_node_t;

struct queue_node_struct
{
    int data;             // данные узла
    queue_node_t *next;   // указатель на следующий элемент
};

typedef struct queue_struct queue_struct_t;

struct queue_struct
{
    queue_node_t *start; // указатель на начало очереди
    queue_node_t *end;   // указатель на конец очереди
};
```

***В реализации программы использовался алгоритм обхода в ширину, потому что таким образом быстрее обходить маттрицу смежности.***

## *Пример работы программы*




Пункт 1.Ввод матрицы смежности:
0 1 0
0 0 1
0 0 0

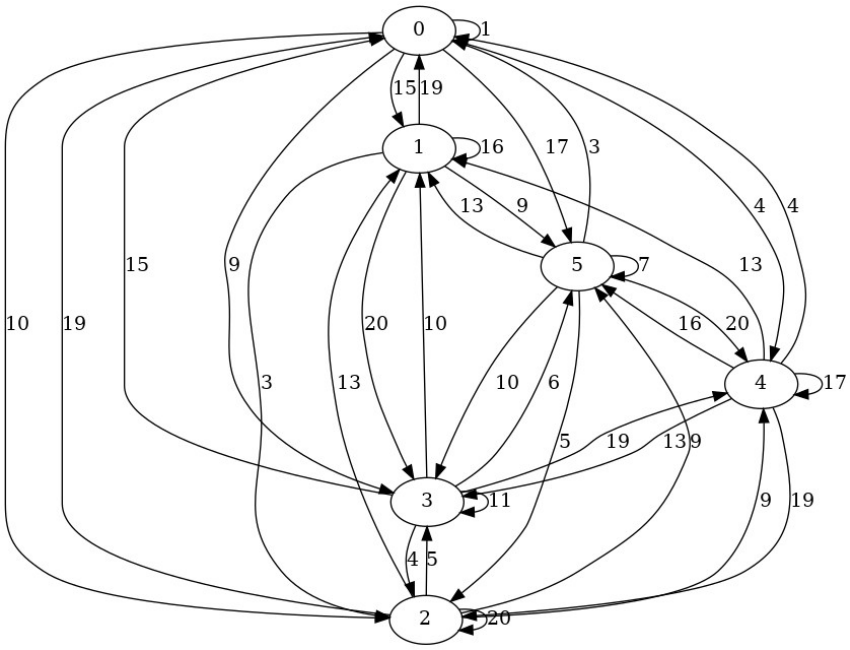
Граф выглядит следующим образом:

Данный граф не является связным.
Если бы мы не делали проверку на связность обратного графа, то он бы считался связным, так как мой алгоритм начинается с 0 вершины.

Пункт 2. Генерация матрицы смежности:

1 15 10 9 4 17
19 16 3 20 0 9
19 13 20 5 9 9
15 10 4 11 19 6
4 13 19 13 17 16
3 13 5 10 20 7

Граф:




Является связным

## *Тестирование*

*Позитивные тесты.*

| Входные данные | Действия программы | Выходные данные |
|---|---|---|
| Пункт меню 1<br>*Ввод матрицы смежности* | Корректная работа программы<br>Создание матрицы смежности и обратной к ней | Ожидание следующего ключа |
| Пункт меню 2<br>Ввод корректного числа — кол-во вершин в графе | Корректная работа программы<br>Создание матрицы смежности и обратной к ней | Ожидание следующего ключа |
| Пункт меню 3<br>Если матрица смежности существует | Корректная работа программы<br>Вывод на экран матрицы смежности | Ожидание следующего ключа |
| Пункт меню 3/4/5<br>Если матрицы смежности не существует | Корректная работа программы<br>Вывод на экран информации, что надо ввести матрицу смежности в 1-2 пунктах | Ожидание следующего ключа |
| Пункт 4<br>Если матрица смежности существует | Корректная работа программы<br>Вывод на экран png изображения графа | Ожидание следующего ключа |
| Пункт 5<br>Если матрица смежности существует | Корректная работа программы<br>Вывод информации о | Ожидание следующего ключа |

| | | |
|---|---|---|
| | связности графа | |

*Негативные тесты*

| Входные данные | Действия программы | Выходные данные |
|---|---|---|
| Ввод числа 10 | Информация о неверном ключе меню | Завершение программы |
| Пункт 1 ввод отрицательных чисел или букв | Информация о неверном количестве вершин в графе или неверном размере пути между вершинами | Завершение программы |

***Вывод:*** я использую матрицу смежности для хранения графа в программе, поэтому в моем случае сложность алгоритма обхода графа в ширину — $O(V^2)$, тут V — количество вершин графа. Этот алгоритм использовался потому что наиболее эффективен в решении поставленной задачи.

**Контрольные вопросы**

*1. Что такое граф?*

Граф – это конечное множество вершин и ребер, соединяющих их, т. е.

G = < V,E >, где V – конечное непустое множество вершин; E – множество ребер (пар вершин).

*2. Как представляются графы в памяти?*

Либо с помощью матрицы смежности, либо с помощью списка смежности.

*3. Какие операции возможны над графами?*

Вывод графа, обход графа, поиск в графе.

*4. Какие способы обхода графов существуют?*

Обход в ширину и обход  в глубину.

*5. Где используются графовые структуры?*

В системах навигации и дорог, в теории сетей.

*6. Какие пути в графе Вы знаете?*

Эйлеров путь, Гамильтонов путь, непростой путь.

*7. Что такое каркасы графа?*

Это деревья, в которые входят все вершины графа и некоторые его рёбра.