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

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

## **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8**

### **«ГРАФЫ»**

Студент Тузов Даниил Александрович

Группа ИУ7 – 32Б

Преподаватель Барышникова Марина Юрьевна  
Силантьева Александра Васильевна

# Оглавление

<u>ОПИСАНИЕ УСЛОВИЯ ЗАДАЧИ.....</u>	<u>3</u>
<u>ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ.....</u>	<u>3</u>
<u>ОПИСАНИЕ СТРУКТУР ДАННЫХ.....</u>	<u>4</u>
<u>ОПИСАНИЕ АЛГОРИТМА.....</u>	<u>5</u>
<u>ОПИСАНИЕ ФУНКЦИЙ.....</u>	<u>6</u>
<u>ОБОСНОВАНИЕ ВЫБОРА АЛГОРИТМА И СТРУКТУР ДАННЫХ.....</u>	<u>8</u>
<u>ПРИМЕР РЕАЛЬНОЙ ЗАДАЧИ.....</u>	<u>8</u>
<u>НАБОР ТЕСТОВ.....</u>	<u>8</u>
<u>ЗАМЕРЫ ВРЕМЕНИ ПОИСКА ИСКОМОГО ПУТИ.....</u>	<u>12</u>
<u>ЗАМЕРЫ РАСХОДОВ ПАМЯТИ В ОПИСАННЫХ СТРУКТУРАХ.....</u>	<u>13</u>
<u>ВЫВОД.....</u>	<u>13</u>
<u>ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ.....</u>	<u>13</u>

## **ОПИСАНИЕ УСЛОВИЯ ЗАДАЧИ**

Обработать графовую структуру в соответствии с указанным вариантом задания. Обосновать выбор необходимого алгоритма и выбор структуры для представления графов. Предложить вариант реальной задачи, для решения которой можно использовать разработанную программу.

Ввод данных – на усмотрение программиста. Результат выдать в графической форме.

Найти самый длинный простой путь в графе.

## **ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ**

### **Входные данные:**

Файл с исходным графом, граф, введенный вручную (количество вершин, количество ребер, вершина, из которой выходит ребро, вершина, в которую входит ребро), цифра, отвечающая за действие, имя файла для вывода графа.

### **Выходные данные:**

DOT-файлы с полученными графами, самый длинный путь, время поиска самого длинного простого пути в графе

### **Меню:**

- 1 — Чтение из файла
- 2 — Чтение с клавиатуры
- 3 — Поиск самого длинного простого пути
- 4 — Вывод графа в файл
- 0 — завершение программы

### **Описание программы:**

Программа позволяет ввести граф вручную или из файла, вывести его в графической форме и найти в нем и вывести самый длинный простой путь

### **Способ обращения к программе:**

Взаимодействие через консоль.

### **Аварийные ситуации:**

1. Ошибка ввода  
Код ошибки — 1
2. Ошибка памяти  
Код ошибки — 2
3. Ошибка файла  
Код ошибки — 3

## **ОПИСАНИЕ СТРУКТУР ДАННЫХ**

### Список для хранения ребер

start – из какой вершины выходит ребро

finish – в какую вершину входит ребро

cost – стоимость/длина ребра (для моей задачи = 1 всегда)

next – указатель на следующее ребро

```
typedef struct edge_type
{
    int start;
    int finish;
    int cost;
    struct edge_type *next;
```

```
} edge_t;
```

### Описание узла графа

start – номер вершины

connected – список исходящих ребер

```
typedef struct node_type
```

```
{
```

```
int start;
```

```
edge_t *connected;
```

```
} node_t;
```

### Описание графа

size – размер графа

array – массив указателей на вершины графа

```
typedef struct graph_type
```

```
{
```

```
size_t size;
```

```
node_t **array;
```

```
} graph_t;
```

## **ОПИСАНИЕ АЛГОРИТМА**

1. Пользователь выбирает действие
2. Согласно выбранному действию вызывается соответствующая функция
  1. При поиске самого длинного простого пути в графе запускается dfs для каждой вершины, который перезаписывает самый длинный простой путь.
3. Выполняется логика этой функции

4. Если во время выполнения функции возникла ошибка, то выводится сообщение, поясняющее эту ошибку. Программа аварийно завершается
5. Если ошибки не возникло, возвращаемся к 1 пункту

## ОПИСАНИЕ ФУНКЦИЙ

`void discription();` - Функция выводит описание программы

`void menu();` - Функция выводит меню пользователю

`int read_str(char **buf);` - Чтение строки в буфер buf. Возвращает код ошибки

`edge_t *edge_create(int start, int finish, int cost);` - Функция создания ребра по заданной начальной вершине start, конечной вершине finish и стоимости cost. Возвращает указатель на ребро

`void edge_clear(edge_t *edge);` - Функция очистки памяти из-под вершины edge

`void edge_print_dot(FILE *f, edge_t *edge);` - Функция печати ребра edge в формате dot-файла f

`node_t *node_create(int start);` - Создание вершины с номером start. Возвращает указатель на вершину

`void node_insert(node_t *node, edge_t *edge);` - Добавление очередного исходящего из вершины node ребра edge

`void node_destroy(node_t *node);` - Уничтожение объекта node с предварительной очисткой памяти из-под ребер

`void node_make_empty(node_t *node);` - Функция очистки памяти из-под ребер вершины node

`void node_print_dot(FILE *f, node_t *node);` - Функция печати вершины node в dot-файл f

`graph_t *graph_create(size_t size);` - Функция создания объекта графа с размером size. Возвращает указатель на граф

`void graph_destroy(graph_t *graph);` - Функция уничтожения объекта графа с предварительной очисткой памяти из-под вершин и ребер

`void graph_print_dot(FILE *f, graph_t *graph);` - Функция печати графа graph в dot-файл f

`int graph_read(FILE *f, graph_t **graph);` - Функция чтения графа graph из файла f. Возвращает код ошибки

`void dfs(int vert, graph_t *graph, int paint[], int path[], int cur_len, int max_path[], int *max_len);` - Функция поиска в глубину, которой передается текущая вершина vertex, исходный граф graph, массив закрашенных вершин paint, массив текущего пути до текущей вершины path, текущий размер пути cur\_len, массив максимального пути в графе max\_path, размер максимального пути в графе max\_len

```
int find_longest_simple_path(FILE *f, graph_t *graph); -
```

Функция, которая находит максимальный простой путь в графе graph и печатает его в файл f. Возвращает код ошибки

## ОБОСНОВАНИЕ ВЫБОРА АЛГОРИТМА И СТРУКТУР ДАННЫХ

В качестве структуры данных я использую список смежности, потому что при такой реализации есть возможность провести **несколько** дорог от одной вершины до другой (Например, из вершины «1» можно провести два пути в вершину «2»)

В качестве алгоритма я выбрал DFS. В отличии от BFS он не требует написания дополнительных структур данных (очереди). А также в DFS легче восстанавливать искомый путь, чем в других реализациях. Еще одним плюсом DFS является то, что можно найти максимальный простой путь из **конкретно-заданной** пользователем вершины.

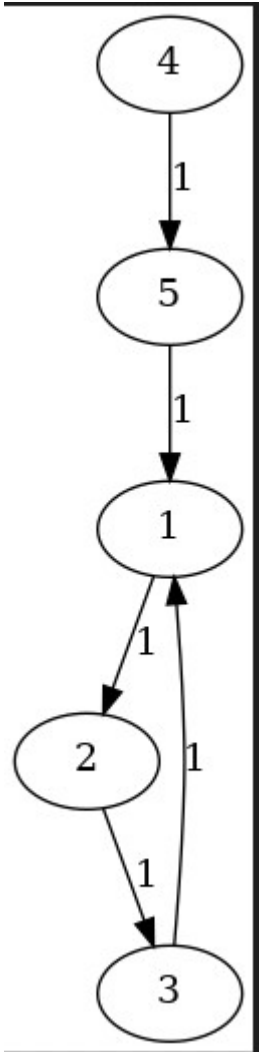
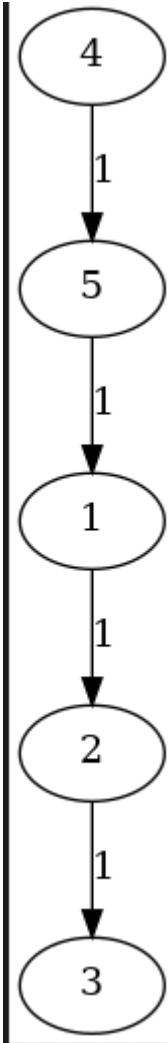

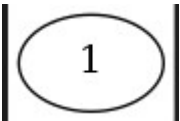
## ПРИМЕР РЕАЛЬНОЙ ЗАДАЧИ

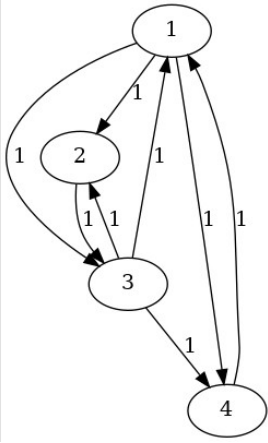
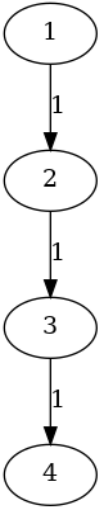
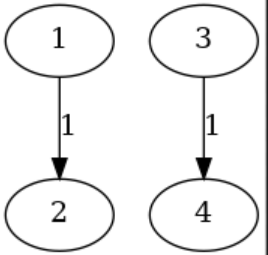
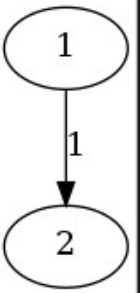
Необходимо построить линию метро так, чтобы эта линия проходила через наибольшее количество станций.

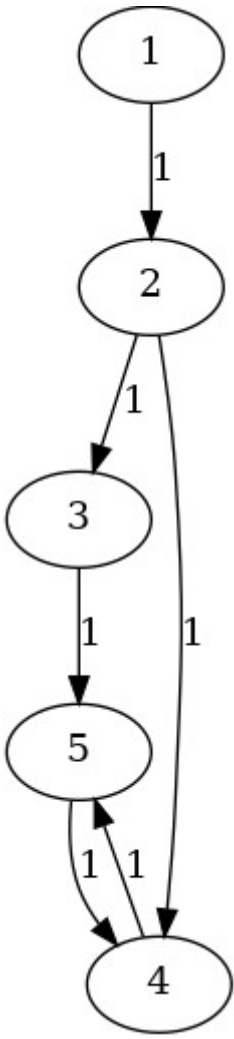
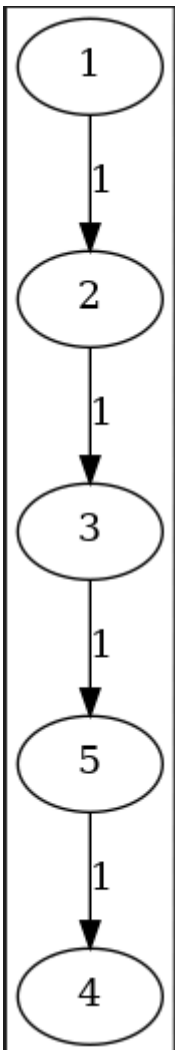
## НАБОР ТЕСТОВ

№	Описание теста	Ввод пользователя	Ответ программы
1	Выбор действия	0	Выход из программы
2	Выбор действия	1	Чтение графа из файла
3	Выбор действия	2	Ввод графа вручную
4	Выбор действия	3	Поиск максимального простого



			пути в графе
5	Выбор действия	4	Экспорт графа в дот-файл
6	Поиск максимального простого пути в графе		
7	Поиск максимального простого пути в графе	 5 несвязных вершин	

8	Поиск максимального простого пути в графе		
9	Поиск максимального простого пути в графе		

10	Поиск максимального простого пути в графе		
11	Ввод графа	5 4 1 2 2 3 3 4	Некорректный ввод количества ребер
12	Ввод графа	5 2 1 6 1 5	Некорректный ввод ребра
13	Ввод графа	5 1 a 1	Некорректный ввод ребра

14	Ввод графа	5 а	Некорректный ввод количества ребер
15	Ввод графа	А 5	Некорректный ввод количества вершин
16	Ввод имени файла	Несуществующий файл	Ошибка файла

### **ЗАМЕРЫ ВРЕМЕНИ ПОИСКА ИСКОМОГО ПУТИ**

В ходе эксперимента для получения результатов проводилось 1000 замеров по времени. Усредненный результат представлен в таблице

Числа в таблице представлены в **микросекундах**

Количество вершин в графе	Время работы алгоритма
1	1.4
5	5.2
10	13.8
100	92.4
500	1725.2
1000	6957.2

## ЗАМЕРЫ РАСХОДОВ ПАМЯТИ В ОПИСАННЫХ СТРУКТУРАХ

Числа в таблицах представлены в **байтах**

	Граф	Вершина	Ребро
Память	16	12	20

## ВЫВОД

В ходе лабораторной работы я познакомился с графами на примере задачи о поиске максимального простого пути в графе с помощью алгоритма DFS. Я также выяснил, что мой алгоритм работает примерно за  $O(n*n)$ , где  $n$  – количество вершин. В теории же алгоритм работает за  $O(n * m)$ , где  $n$  – количество вершин а  $m$  – количество ребер. Что в целом похоже на правду

## ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

### 1. Что такое граф?

Граф – это конечное множество вершин и ребер

### 2. Как представляются графы в памяти?

Либо в виде матрицы смежности, либо в виде списка смежности

### 3. Какие операции возможны над графами?

Поиск кратчайшего пути от одной вершины к другой (если он есть); поиск кратчайшего пути от одной вершины ко всем другим; поиск кратчайших путей между всеми вершинами; поиск эйлера пути (если он есть); поиск гамильтонова пути (если он есть)

### 4. Какие способы обхода графов существуют?

Обход в глубину (DFS), обходы в ширину (BFS)

### 5. Где используются графовые структуры?

Например, при построении коммуникационных линий между городами

## **6. Какие пути в графе Вы знаете?**

Эйлеров путь – путь, содержащий все вершины и гамильтонов путь – путь, содержащий все ребра

## **7. Что такое каркасы графа?**

Связный подграф этого графа, содержащий все вершины графа и не имеющий циклов.