



**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 ПО ДИСЦИПЛИНЕ «ТИПЫ И СТРУКТУРЫ ДАННЫХ»**

**Графы**

**Вариант №13**

**Студент: Ширяев А.А.**

**Группа: ИУ7-33Б**

**Преподаватель:**

**Силантьева А.В.**

**2024 г.**

Лабораторная работа №8 по дисциплине “Типы и структуры данных” .....	1
Условие задачи.....	3
Описание техзадачи.....	3
Описание исходных данных.....	3
Описание задачи, реализуемой программой.....	4
Способ обращения к программе.....	4
Для обращения к программе запускается файл app.exe.....	4
Описание возможных аварийных ситуаций и ошибок пользователя.....	4
Информация о городе.....	5
Информация о полосе дороги.....	5
Информация о городах и дорогах (граф).....	6
Описание алгоритма.....	6
Обоснование формы представления графа. Сложность алгоритма.....	11
Сложность.....	12
Реальная задача.....	12
Выводы по проделанной работе.....	13

Цель работы — реализовать алгоритмы обработки графовых структур: поиск различных путей, проверка связности, построение остовых деревьев минимальной стоимости.

## Условие задачи

### Вариант 13

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.

В системе двусторонних дорог за проезд каждой дороги взимается некоторая пошлина. Найти путь из города А в город В с минимальной величиной  $S+P$ , где  $S$  - сумма длин дорог пути, а  $P$  - сумма пошлин проезжаемых дорог

**ПРИМЕЧАНИЕ!** В отчёте будут использоваться обозначения для действий:

<read> - Чтение графа из файла

<find\_path> - Нахождение пути из города А в город В с минимальной величиной  $S+P$ , где  $S$  - сумма длин дорог пути, а  $P$  - сумма пошлин проезжаемых дорог

<output> - Графический вывод графа

<exit>- Выход из программы

## Описание техзадачи

### Описание исходных данных

Данные на входе: Меню. Код действия. Далее для каждого действия:

<read> - Файл с данными

<find\_path> - Граф, Название города А, Название города В

<output> - Граф

<exit>- -

Данные на выходе:

<read> - Граф

<find\_path> - Изображение с визуализацией графа с выделенным путём, минимальная величина  $S+P$ , величина  $S$ , величина  $P$

<output> - Изображение с визуализацией графа

<exit>- -

## Описание задачи, реализуемой программой

Программа реализует ряд действий:

<read> - Чтение графа из файла

<find\_path> - Нахождение пути из города  $A$  в город  $B$  с минимальной величиной  $S+P$ , где  $S$  - сумма длин дорог пути, а  $P$  - сумма пошлин проезжаемых дорог

<output> - Графический вывод графа

<exit>- Выход из программы

## Способ обращения к программе

Для обращения к программе запускается файл app.exe.

## Описание возможных аварийных ситуаций и ошибок пользователя

Программа может не вывести результат, а вывести сообщение об ошибке. Данная ситуация может произойти при условии:

Ошибки пользователя

1. Неверный код действия
2. Выполнение действия при нехватке данных в файле (Чтение графа из файла)
3. Выполнение действия при некорректности данных в файле (Чтение графа из файла)

## Аварийные ситуации

1. Программа не смогла выделить необходимую память для работы (для чтения графа из файла)
2. Программа создала некорректный файл с деревом (если в статистике слишком много элементов)

## Описание внутренних СД

### Информация о городе

Информация о городе представляет собой структуру на языке Си, состоящую из:

```
typedef struct city city_t;  
You, 5 seconds ago | 2 authors (shaa23u669)  
struct city  
{  
    char *name;  
  
    int is_major;  
    city_t *last;  
    color_t color;  
  
    double min_value;  
    double min_fee;  
};
```

name – Название города

is\_major – Индикатор, помечающий вершину как посещённую

\*last – Указатель на предыдущий элемент. Необходим для визуализации подходящего пути на изображении

color - Значение цвета вершины города на визуализации

min\_value – Минимальная сумма длин пройденных дорог

min\_fee – Минимальная сумма пошлин пройденных дорог

## Информация о полосе дороги

Информация о полосе дороги представляет собой структуру на языке Си, содержащую:

```
typedef struct
{
    double value;
    double fee;
} lane_t;
```

value – Стоимость проезда по полосе

fee – Пошлина за проезд по полосе

## Информация о городах и дорогах (граф)

Информация о городах и дорогах (граф) представляет собой структуру на языке Си, содержащую:

```
typedef struct
{
    city_t **cities;
    size_t n;

    lane_t ***lanes;
} graph_t;
```

cities – Множество городов

n – Количество городов

lanes – Матрица полос дорог

## Описание алгоритма

Для начала считывается код действия. Далее в зависимости от выбранного действия:

<read>

```
graph_error_t graph_read_from_file(graph_t **graph, char *filename)
```

- С клавиатуры считывается название файла.
- Выполняется чтение файла
- Если информация прочиталась корректно, считанные данные переносятся в граф

<find\_path>

```
graph_error_t graph_way_find_path(graph_t *graph, char *beg, char *end)
```

- Все данные городов, необходимые для нахождения пути, инициализируются удобными для алгоритма значениями
- Берётся город с наименьшей суммой длин дорог и пошлин (в начале работы алгоритма берётся начальный город).
- Полосы дорог, ведущие от взятого города, изменяют информацию о городах, являющихся концами полос: если новая информация меньше в сумме, то информация заносится в город на конце полосы
- Выбирается новый город по алгоритму из П.2 до того момента, пока не пройдёт  $n$  итераций ( $n$  – количество городов), выбранный город не будет конечным или новый город не будет найден (в графе нет дорог для прохода из начального города в конечный)

<output>

```
void graph_output(graph_t *graph, void (*additional_output)(graph_t *, FILE *))
```

- Считывается граф. Данные о нём записываются в файл на языке DOT
- Язык DOT переводится в .png файл при помощи утилиты Graphviz

<exit>

```
case CODE_EXIT:  
    graph_free(&graph);  
    flag = false;  
  
    break;
```

- Освобождается память, выделенная под динамически выделенные данные программы (Граф)
- Программа завершает свою работу

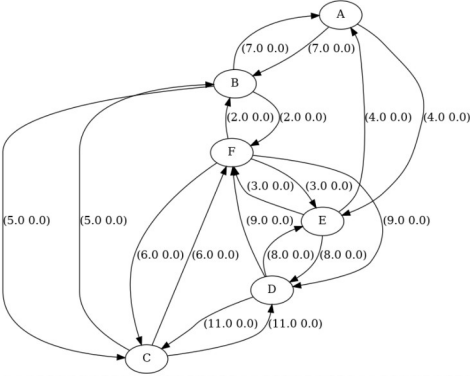
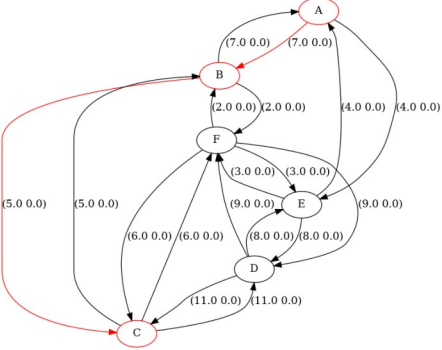
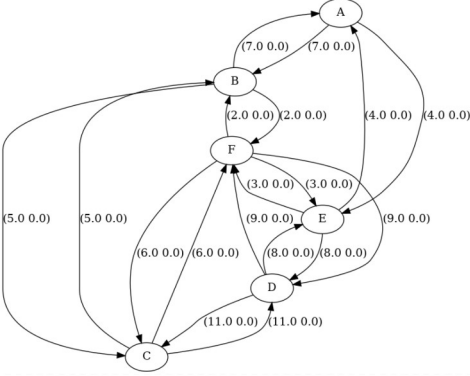
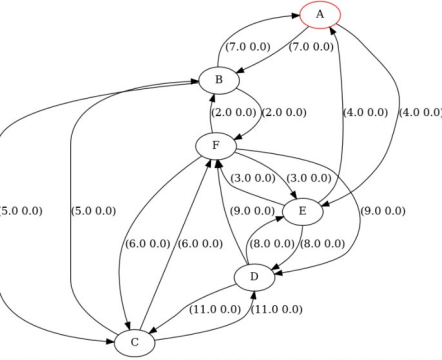
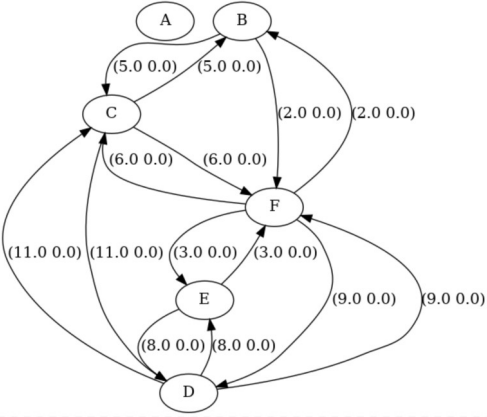
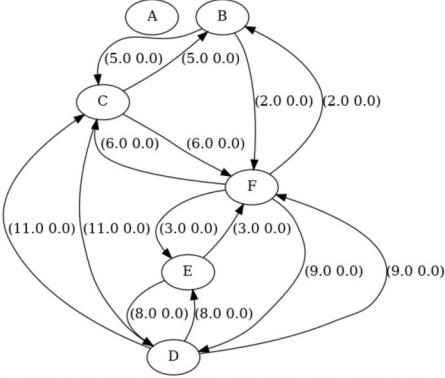


## Набор тестов

В ходе выполнения лабораторной работы были написаны тесты для проверки работы программы

### Позитивные тесты

Номер теста	Входные данные	Ожидаемые выходные данные
01 - <read> Чтение корректного файла	Файл с информацией:  Rico  Maers  Luo  Dumrock  Io   1 2 3  4 5 7  6 1 9  1 14 3    1 52 2  1 2 3  4 5 7    6 1 9  1 14 3	Code: 1 Enter filename: test1.txt  GRAPH WAS READ SUCCESSFULLY

	1 52 2	
02 - <find_path> Поиск кратчайшего пути из А в С		MIN PATH S+P: 12.000000 S: 12.000000 P: 0.000000 
03 - <find_path> Поиск кратчайшего пути из города А в А		MIN PATH S+P: 0.000000 S: 0.000000 P: 0.000000 
04 - <find_path> Поиск кратчайшего пути из А в D при отсутствии путей из А в D		PATH ISN'T EXIST 

05 - <exit> Выход из программы	8	*Выход из программы*
--------------------------------	---	----------------------

## Негативные тесты

Номер теста	Входные данные	Выходные данные
01 - Код неправильный	15	INVALID CODE
02 - Код с иными символами	1.1	INVALID CODE
03 - <find_path> Нет графа	2	NO DATA
04 - <output> Нет графа	3	NO DATA
05 - <find_path> Города с таким названием нет	1 test2.txt 2 Mumbai a	INVALID CITY NAME: UNKNOWN CITY

# Обоснование формы представления графа. Сложность алгоритма.

Граф представляется в виде:

**Массив вершин**

**Кол-во вершин**

**Матрица смежности** - Представляем так, так как:

**1)** В современном мире (по крайней мере в России) существует множество дорог из одного города в другой, не пересекая другие города. Следовательно, множество дуг в большинстве случаев будет довольно плотным.

**2)** Такой код будет удобно редактировать для ввода пошлин для отдельных направлений дороги.

## Сложность

В худшем случае алгоритм совершит  $n$  итераций ( $n$  – число городов). При итерациях совершается:

- 1)  $n$  проходов по матрице смежности
- 2)  $n$  проходов по массиву городов

Релаксация совершается за  $O(1)$

Итоговая сложность алгоритма —  $O(n(n + n)) = O(n^2 + n^2)$

## Реальная задача

Данную программу можно использовать для оптимизации маршрутов в наземной транспортной системе дальнего следования.

Необходимо разработать маршрут автобуса дальнего следования таким образом, чтобы поездка из одного города в другой заняла наименьшее время.

# Выводы по проделанной работе

Граф – это конечное множество вершин и ребер, соединяющих их. При помощи графа можно решать современные актуальные проблемы, такие как нахождение кратчайшего маршрута, который должен пройти курьер для посещения всех клиентов или выяснения, можно ли обойти все достопримечательности города, не посещая ранее посещённые.

## Контрольные вопросы

### **1. Что такое граф?**

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

### **2. Как представляются графы в памяти?**

Графы в памяти могут представляться различным способом. Один из видов представления графов —

- Матрица смежности (матрица, хранящая все дуги/рёбра с весами)
- Списки рёбер (список, хранящий рёбра определённой вершины)
- Матрица инцидентности (Матрица, хранящая информацию о НАЛИЧИИ связи между вершинами)
- Список смежных вершин (список, хранящий все дуги/рёбра определённой вершины)
- и так далее

### **3. Какие операции возможны над графами?**

Основные операции по работе с 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?**

Основные способы обхода графов:

- Обход в глубину
- Обход в ширину
- Алгоритм Дейкстры
- Алгоритм Беллмана-Форда
- Алгоритм Флойда-Уоршелла
- Алгоритм Крускала
- Алгоритм Прима

#### **5. Где используются графовые структуры?**

Графовые структуры используются в задачах со сложными взаимосвязями между объектами (оптимизация пути, маршрутизация сетей)

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

- Гамильтоновы пути
- Эйлеровы пути

#### **7. Что такое каркасы графа?**

Каркасы графа - связные подграфы этого графа, содержащие все вершины графа и не имеющий циклов.