Министерство образования и науки 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 (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7
«Граф»

Студент, группа **ИУ7-33Б** **Рядинский К.В.,**

2020 г.

Описание условия задачи

Обработать графовую структуру в соответствии с указанным вариантом задания. Обосновать выбор необходимого алгоритма и выбор структуры для представления графов. Ввод данных – на усмотрение программиста. Результат выдать в графической форме.

В графе найти максимальное расстояние между всеми парами его вершин

Техническое задание

Входные данные:

1. Имя файла

Выходные данные

1. Таблица, содержащая максимальное расстояние между i и j вершинами
2. Графическое изображение графа

Функции программы

1. Поиск максимального расстояния между между всеми парами вершин графа
2. Графическое изображение графа

Обращение к программе

Запускается из терминала

Аварийные ситуации

1. Ввод несуществующего файла
2. Ввод пустого или содержащего некорректные данные (буквы, вещественные числа) файла
3. Ввод дуги с нулевой меткой
4. Превышение заданного кол-ва вершин

Структуры данных

```
typedef struct adjlist_node
{
    int v;                  — Конец дуги
    int weight;             — Метка дуги
} adjlist_node_t;

typedef struct graph
{
    int V;                  — Кол-во вершин
    int capV;               — Вместимость графа
    node_t **adj;           — Список смежности
} graph_t;
```

Структура программы

void graph_add_edge(graph_t *graph, int u, int v, int weight); — добавление дуги

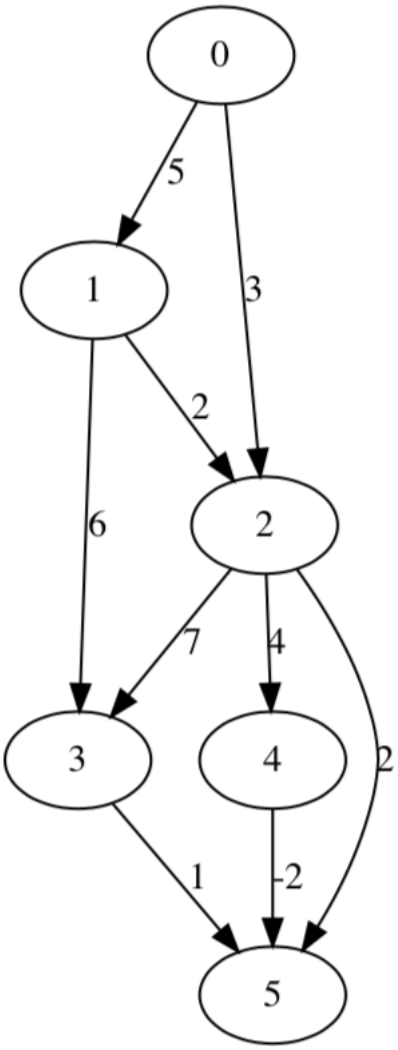
void topological_sort(graph_t *graph, int v, bool *visited, arr_stack_t *stack); — обход графа по спискам смежности

void longest_path(graph_t *graph, int s, vector_t *map); — Поиск длиннейших путей

void graph_to_jpeg(const graph_t *graph, const char *name, FILE *f, vector_t *map); — экспорт графа для графического представления

int input_graph(graph_t **__graph, vector_t **__map, FILE *f); — ввод графа

Визуализация



Алгоритм

Пользователем вводится путь к файлу с нужными данными для построения графа (начало и конец дуги, метка дуги). Далее проводится обход графа с поиском длиннейших путей по всем парам вершин. После этого пользователю выводится таблица стоимостей и время обработки с требуемой (затраченной) памятью.

Тесты

	Тест	Ввод	Вывод
1	Несуществующий файл	test.test	Ошибка, неправильное имя файл
2	Пустой файл	empty.empty	Ошибка, пустой файл
3	Файл содержит не только целые числа	badfile.txt	Ошибка чтения из файла.
4	Файл содержит метку, равную нулю	1 2 0	Ошибка, нулевая метка

Оценка эффективности

Кол-во вершин	Время обработки нс
5	37000
10	46000
15	98000

Кол-во вершин	Затраченная память нс
5	128
10	152
15	264

Контрольные вопросы

1. Что такое граф?

Граф – конечное множество вершин и соединяющих их ребер; **G = <V, E>**. Если пары **E** (ребра) имеют направление, то граф называется ориентированным; если ребро имеет вес, то граф называется взвешенным.

2. Как представляются графы в памяти?

С помощью матрицы смежности или списков смежности.

3. Какие операции возможны над графами?

Обход вершин, поиск различных путей, исключение и включение вершин.

4. Какие способы обхода графов существуют?

Обход в ширину (BFS – Breadth First Search), обход в глубину (DFS – Depth First Search).

5. Где используются графовые структуры?

Графовые структуры могут использоваться в задачах, в которых между элементами могут быть установлены произвольные связи, необязательно иерархические.

6. Какие пути в графе Вы знаете?

Эйлеров путь, простой путь, сложный путь, гамильтонов путь.

7. Что такое каркасы графа?

Каркас графа – дерево, в которое входят все вершины графа, и некоторые (необязательно все) его рёбра.

Вывод

В данном алгоритме используется способ хранения графа в виде списка связностей. Используется поиск в глубину и полный перебор вершин и ребер. Тогда, если **V** — вершины, а **E** — ребра, то асимптотическая сложность алгоритма будет **O(E + V)**.
Был выбран алгоритм обхода дерева в глубину, так как нам нужно обойти все вершины, а этот алгоритм реализуется достаточно просто с использованием списка смежностей, использование других структур данных будет не столь удобным.