	<p align="center"> 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) </p>
---	--

ФАКУЛЬТЕТ _____ Информатика и системы управления (ИУ) _____

КАФЕДРА _____ Программное обеспечение ЭВМ и информационные технологии (ИУ7) _____

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7 **«ОБРАБОТКА ГРАФОВ»**

Студент, группа

Криков А.В ИУ7-33Б

2020 г.

Описание условия задачи

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

Найти самый длинный простой путь в графе

Техническое задание

Входные данные:

1. **Целое число, представляющее собой количество вершин в рассматриваемом графе:** целое положительное число.
2. **Количество связей для рассматриваемой вершины**
3. **Целочисленные значения связей для рассматриваемой вершины**

Выходные данные:

1. Графическая визуализация полученного графа с помощью Graphviz, красным отмечен самый длинный простой путь.

Функция программы: программа решает задачу, указанную в условии и визуализирует найденное решение при помощи Graphviz.

Обращение к программе: запускается из терминала.

Аварийные ситуации:

1. Некорректный ввод количества вершин.
На входе: неположительное целое число, нецелое число или буква.
На выходе: сообщение «Неверное количество вершин графа!»
2. Некорректный ввод количества связей для i -ой вершины .
На входе: целое число выходящее за диапазон $[0, V]$ нецелое число, буква.
На выходе: сообщение «Неверное количество связей!»
3. Некорректный ввод связи для i -ой вершины .
На входе: нецелое число, буква или попытка провести путь из вершины в саму себя.
На выходе: сообщение «Указана неверная вершина!»

Структуры данных

Хранение списка смежности:

```
typedef struct
{
    adj_node_t **adj_matrix;
    int size;
    int *visited;
} graph_t;
```

Поля структуры:

- ***int size*** – размер матрицы (количество вершин в графе);
- ***int **matrix*** – указатель на массив указателей
- ***int *visited*** — указатель на массив посещенных вершин ;

Хранение информации о ребре графа:

Хранение цепочек рёбер:

```
typedef struct adj_node adj_node_t;
struct adj_node
{
    int vertex;
    adj_node_t *next;
};
```

Поля структуры:

- ***int vertex*** — номер вершины ;
- ***adj_node_t *next*** – указатель на следующую вершину;

Прототипы функций

void push_back(adj_node_t **adj_node, int vertex) — добавляет вершину в граф

На входе: указатель на элемент списка смежности и добавляемая вершина

На выходе: добавленная вершина в список смежности

void create_graph(graph_t *graph, int size) — инициализация графа

На входе: переменная под которую нужно выделить память, размер графа

На выходе: инициализированный граф.

int DFS(graph_t *graph, int vertex) — реализация алгоритма поиска в глубину

На входе: граф, вершина для которой находится наибольший простой путь.

На выходе: наибольший путь

void out_graph_dot(const graph_t *graph, char *name) — вывод графа в png формате

На входе: граф, имя файла в который нужно вывести граф

На выходе: граф в формате png.

int input_graph_from_console(graph_t *graph) — ввод графа из консоли

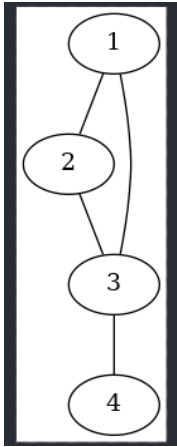
На входе: пустой граф

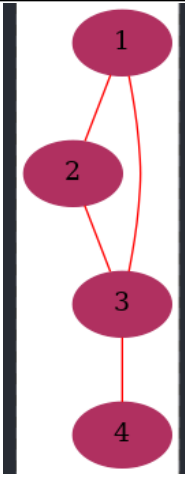
На выходе: заполненный граф

Алгоритм

Для построения самого длинного простого пути используется рекурсивная версия алгоритма DFS. Для каждой вершины наибольший путь выбирается рекурсивно путем просмотра следующей вершины и количества ее связей.

Тесты

	Тест	Пользовательский ввод	Результат
1	Некорректный ввод количества вершин	-1	Неверное количество вершин графа!
2	Некорректный ввод количества вершин	Asdfasdf	Неверное количество вершин графа!
3	Некорректный ввод количества связей для i-ой вершины	6 (при количестве вершин равном 5)	Неверное количество связей!
4	Некорректный ввод номера вершины	-3	Указана неверная вершина!
5	Некорректный ввод номера вершины	sdasdas	Указана неверная вершина!
6	Корректный ввод характеристик	4 2 2 3 2 1 3 3 1 2 4 1 3	

7	Вывод самого длинного простого пути	4 2 2 3 2 1 3 3 1 2 4 1 3	
---	---	---------------------------------------	---

Оценка эффективности

Измерения эффективности производились в тактах процессоров.

Время поиска наибольшего простого пути:

Количество элементов	Время выполнения
5	111407
10	297938
20	678212

Занимаемая память (в байтах):

Количество элементов	Занимаемая память
5	140
10	240
20	440

Контрольные вопросы

1. Что такое граф?

Граф – конечное множество вершин и соединяющих их ребер; $G = \langle V, E \rangle$. 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

Для поиска наибольшего простого пути в графе использовался алгоритм поиска в глубину а также полный перебор. Если граф содержит V вершин и E ребер, то алгоритм выполняется за $O(V + E + V^2)$. Таким образом сложность алгоритма (не учитывая константу $V + E$) будет V^2 (в худшем случае). Данный алгоритм является наиболее эффективным однако для его реализации необходимо выделять дополнительную память в виде массива просмотренных вершин.