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
«ОБРАБОТКА ГРАФОВ»

Студент, группа **Кононенко С., ИУ7-33Б**

2019 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 минимальное (по количеству ребер) подмножество ребер, удаление которых превращает заданный связный граф в несвязный.

Техническое задание

Входные данные:

1. **Целое число, представляющее собой количество вершин в рассматриваемом графе:** целое положительное число.
2. **Пара целых чисел, представляющая собой связи между вершинами:** целые неотрицательные числа в диапазоне от **0** до **V-1**, где **V –** количество вершин графа.

Выходные данные:

1. Графическая физуализация полученного графа, где красным цветом отмечены удаленные ребра.

Функция программы: программа находит решение изложенной в условии задачи и визуализирует найденное решение при помощи фреймворка **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 «Введено недопустимое значение! Повторите попытку»
2. Некорректный ввод связей вершин в графе.
   На входе: целое число выходящее за диапазон **[0;V-1],** нецелое число, буква или попытка провести путь из вершины в саму себя.
   На выходе: сообщение «Введено недопустимое значение! Повторите попытку.»

Структуры данных

Хранение матрицы смежности:

```
typedef struct
{
   int size;
   int **matrix;
} adjmat_t;
```

Поля структуры:

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

Хранение информации о ребре графа:

```
typedef struct
{
   int fvertex, svertex;
} edge_t;
```

Поля структуры:

- ***int* fvertex, svertex –** вершины, связываемые ребром;

Хранение цепочек рёбер:

```
typedef struct
{
   int size;
   edge_t *edges;
} chain_t;
```

Поля структуры:

- ***int* size —** размер цепочки;
- ***edge_t* *edges –** указатель на массив рёбер;

Алгоритм

После ввода элементов графа граф проверяется на связность. Так как в данной задаче реализуется обработка неориентированного графа, то для проверки его на связность достаточно выполнить поиск в глубину для любой вершины и в случае, если по окончании поиска была посещена каждая вершина графа, граф является связным

После того, как была произведена проверка на связность, если граф является несвязным, то он выводится в исходном состоянии. В противном случае, начинают рассматриваться комбинации из **V** рёбер по **1, 2, …, V** ребёр и эти комбинации удаляются из графа, после каждого удаления повторяется процедура с поиском в ширину и так до тех пор, пока не будет получен несвязный граф.

Тесты

	Тест	Пользовательский ввод	Результат
1	Некорректный ввод количества вершин	0	Введено недопустимое значение! Повторите попытку.
2	Некорректный ввод количества вершин	A	Введено недопустимое значение! Повторите поппытку
3	Некорректный ввод номера вершины	5 (при количестве вершин больше или равном 5)	Введено недопустимое значение! Повторите поппытку
4	Некорректный ввод номера вершины	A	Введено недопустимое значение! Повторите поппытку
5	Некорректный ввод связи между вершинами	1 1	Путь в себя невозможен!
6	Корректный ввод характеристик	3; 0 1 / 1 2 / 2 0	Удаленные рёбра отмечены на графе красным цветом! **(вывод графического изображения в упрограмме Gwenview)**

Оценка эффективности

Измерения эффективности реализаций очереди будут производиться в единицах измерения – тактах процессорв. Для измерения была специально написана ассемблерная функция, поэтому погрешность измерений минимальна.

Время выполнения (в тактах процессора):

Количество элементов	Время выполнения
5	124407
10	337938
15	624714

Занимаемая память (в байтах):

Количество элементов	Занимаемая память
5	144
10	488
15	1032

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 данной реализации алгоритм состоит из двух этапов — поиск в глубину и полный перебор. Если **V** – количество вершин графа, а **E** – количество ребер, то алгоритм поиска в глубину имеет асимптотическую сложность **O(V+E)**, а полный перебор рёбер – $\mathbf{O(E^2)}$, таким образом общая сложность алгоритма, без учёта константы будет $\mathbf{O(E^2)}$ (в худшем случае). Можно было воспользоваться алгоритмом поиска остовного дерева при помощи поиска в глубину, но в таком случае было бы необходимо хранить информацию об уже удаленных рёбрах, что накладывает дополнительные ограничения, поэтому выбор пал в сторону более простого, но менее эффективного алгоритма полного перебора.