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
ГРАФЫ
ДИСЦИПЛИНА: ТИПЫ И СТРУКТУРЫ ДАННЫХ

Студент, группа **Боренко А. ИУ7-32Б**

2020 г.

Цель работы

Цель работы: 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.

Описание условия задачи

Вариант 5

Задан граф - не дерево. Проверить, можно ли превратить его в дерево удалением одной вершины вместе с ее ребрами.

Описание Технического Задания

Описание исходных данных и результатов

Исходные данные:

Файл

Реультаты:

Граф исходный

Графы — деревья / Сообщение о невозможности превратить граф в дерево.

Описание задачи, реализуемой программой

Ввод названия файла с описанием графа. Проверка условия для графа. Вывод основного графа и графов-деревьев в png. Замер среднего времени, ушедшего на обработку графа.

Способ обращения к программе

Обращение к программе происходит через терминал. В командную строку вводится «./program».

Описание возможных аварийных ситуаций и ошибок пользователя

1. Ввод не существующего файла

Описание внутренних структур данных

Граф

```
typedef struct edge edge_t;
typedef struct graph graph_t;
typedef struct edges_array edges_array_t;
typedef struct peaks_array peaks_array_t;

struct edge
{
	int p1;
	int p2;
};
```

```
struct edges_array
{
	edge_t *elements;
	int len;
};

struct peaks_array
{
	int *elements;
	int len;
};

struct graph
{
	edges_array_t *edges;
	peaks_array_t *peaks;
};
```

Вспомогательные структуры данных для обработки графа

```
struct array
{
	int *elements;
	int len;
};

typedef struct stack stack_t;

struct stack
{
	int *start;
	int *ps;
	int *end;
};
```

Описание алгоритма

Общий алгоритм

1. Считать имя файла, введенное пользователем.
2. Считать данные из файла в граф
3. Для каждой вершины
    1. Удалить текущую вершину
    2. Проверить граф на связность
    3. Проверить выражение |V| = |E| + 1
    4. Если граф связный и удовлетворяет выражению
        1. Записать вершину в массив ответов
4. Для каждой выписанной вершины построить граф и вывести в dot
5. Провести 100000 повторений алгоритма делая замеры времени и считая среднее время
6. Вернуть кол-во вершин, удалив которые можно из заданого графа получить дерево.

Основные функции

```
// Вывод файла в dot
void output_graph_as_dot(graph_t g, char *filename);

// Работа с массивом (создание, удаление, вывод)
array_t *create_array(int size);
void delete_array(array_t *s);
void output_array(array_t s);

// Проверка графа на связность
int check_graph_connectivity(graph_t g);

// Работа с ребром (создание, удаление, вывод)
edge_t *tw_create_edge(int p1, int p2);
void tw_delete_edge(edge_t *e);
void tw_output_edge(edge_t e);

// Работа с массивом ребер (создание, удаление, вывод)
edges_array_t *tw_create_edges_array(int len);
void tw_delete_edges_array(edges_array_t *arr);
void tw_output_edges_array(edges_array_t arr);

// Работа с массивом вершин(создание, удаление, вывод)
peaks_array_t *tw_create_peaks_array(int len);
void tw_delete_peaks_array(peaks_array_t *arr);
void tw_output_peaks_array(peaks_array_t arr);

// Работа с графом (создание, удаление, вывод)
graph_t *tw_create_graph(int peaks_count, int edges_count);
void tw_delete_graph(graph_t *g);
void tw_output_graph(graph_t g);

// Чтение файла
int read_file(char *filename, graph_t **g);

// Глубокое копирование графов
int copy_graphs(graph_t src, graph_t **dst);
// Удаление  вершины
void delete_peak(graph_t *g, int peak_num);
// Проверка выражения |V| = |E| + 1
int check_calculation(graph_t g);
```

Тесты

Негативные

Некорректный файл

Отсутствующий файл

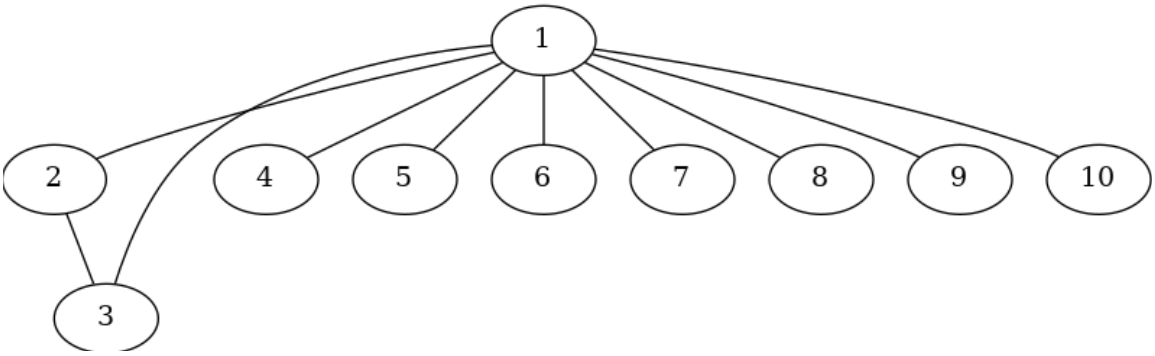
Граф дерево

Позитивные

Граф, который можно сделать деревом.

Граф, который нельзя сделать деревом.

Примеры выводы графиков



Выводы о проделанной работе

5 Вершин 1 Связь Время 4 мсек

5 Вершин 5 связей Время 5 мсек

5 Вершин 10 связей Время 6 мсек

10 Вершин 1 связь Время 5 мсек

10 Вершин 10 связей Время 17 мсек

10 Вершин 45 связей Время 53 мсек

При увеличении кол-во вершин значительно вырастает необходимая память. При увеличении ребер — время обработки.

Расчетная память = sizeof(int) * n + sizeof(int)*2 * m + sizeof(int*)*2

n — кол-во вершин, m — кол-во дуг

Выбор способа хранения графа.

В этой задаче не важен вес ребер графа, поэтому матрицы (смежности, инцидентности) не обязательны. Матрицы занимают много места, поэтому можно хранить массив ребер. Также необходимо хранить названия вершин, т. к. граф может быть несвязным. Поэтому помимо массива дуг, должен быть и массив вершин.

Алгоритм поиска в глубину

Алгоритм выбран из-за своей понятности и простоты реализации. При обработке не полностью заполненных графов алгоритм достаточно быстрый.

Возможная задача

Данный алгоритм может использоваться при переносе графовых баз данных в базы данных — деревья. В дереве можно выбрать корень так, чтобы быстрее их находить. К тому же в деревьях может быть построена иерархическая струтура, можно засекречивать ветку, начинающеюся с данного документа.

Ответы на вопросы

1. Что такое граф?

Это набор значений (вершин) и связей между ними.

2. Как представляются графы в памяти?

В зависимости от реализации.

Возможные представления: Матрица смежности, список дуг, матрица инцидентности.

3. Какие операции возможны над графами?

Обход графа, поиск кратчайшего пути, проверка на связность, подсчет вершин и дуг, поиск в глубину, поиск Эйлерова пути, поиск Гамильтонова пути.

4. Какие способы обхода графов существуют?

Обход в глубину, обход в ширину.

5. Где используются графовые структуры?

Поиск суммарных дорог между городами, чтобы длина пути

была минимальной.

6. Какие пути в графе Вы знаете?

Гамильтонов, Эйлеров, кратчайший, длинейший.

7. Что такое каркасы графа?

Остовное дерево графа — это связный подграф, содержащий все вершины графа и не имеющий циклов (дерево).