



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

Студент Жаринов М. А.

Вариант 5

Группа ИУ7-32Б

Название предприятия НУК ИУ МГТУ им. Н. Э. Баумана

Студент _____ Жаринов М. А.

Преподаватель _____ Барышникова М. Ю.

Описание условия задачи

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

Обосновать выбор необходимого алгоритма и выбор структуры для представления графов. Предложить вариант реальной задачи, для решения которой можно использовать разработанную программу

Реальная задача:

Рассмотрим государство, состоящее из городов (вершин) и дорог (ребер) (дороги двусторонние, поэтому граф неориентирован). Глава государства – перфекционист, и хочет, чтобы из каждого города можно было добраться до любого другого и при этом только одним способом (чтобы граф был деревом). При этом у главы государства (диктатора) есть в распоряжении одна ядерная бомба. Может ли диктатор взорвать один город так, чтобы удовлетворить свой перфекционизм.

Техническое задание

Исходные данные:

Граф: Текстовый файл, содержащий данные о неориентированном графе в следующем формате: На первой строке количество вершин в графе, на второй количество ребер, дальше ребра в виде пар номеров вершин, которым инцидентно это ребро.

Выходные данные:

Входной граф, с удаленной одной вершиной и инцидентными ей ребрами, являющийся деревом.

Реализуемая задача:

Поиск вершины, при удалении которой граф становится деревом, и ее удаление

Способ обращения к программе:

Строка запуска программы ./app.exe

Аварийные ситуации:

1. Ошибка ввода при чтении файла. Код ошибки 1
2. Повторяющееся ребро в данных о графе. Код ошибки 2
3. Петля в данных о графе. Код ошибки 3
4. Недостаток свободного места в оперативной памяти. Код ошибки 4
5. Входной граф является деревом. Код ошибки 5

Внутренние структуры данных

Граф хранится в структуре `graph_t`.

```
typedef struct
{
    graph_node_t *node_list;
    int number_vertices;
    int number_edges;
} graph_t;
```

Поле `node_list` – указатель на начало списка вершин графа

Поле `number_vertices` – количество вершин в графе

Поле `number_edges` – количество ребер в графе

Вершины графа хранятся в структуре `graph_node_t`.

```
struct graph_node_t
{
    int number;
    bool visited;
    adj_node_t *head;
    graph_node_t *next;
};
```

Поле `number`– номер-название вершины

Поле `visited`– флаг посещения вершины при поиске в глубину

Поле `head`– указатель на начало списка смежных вершин данной

Поле `next`– указатель на следующую вершину в списке

Информация о вершине, смежной с данной, хранится в структуре

`adj_node_t`.

```
struct adj_node_t
{
    graph_node_t *node;
    adj_node_t *next;
};
```

Поле `node`– указатель на вершину, смежную данной в основном списке

Поле `next`– указатель на следующую вершину в списке смежности

Описание алгоритма

1. Программа запрашивает имя файла с данными о графе и считывает его
2. Программа считывает файл и проверяет его на корректность
3. Программа проверяет то, что граф не является деревом
4. Граф визуализируется
5. Программа запускает цикл по всем номерам вершин и для каждой проверяет, является ли граф с удаленной этой вершиной деревом
6. Если граф с удаленной вершиной является деревом, то эта вершина удаляется
7. Получившийся граф визуализируется

Алгоритм проверки графа на дерево

1. Производится поиск в глубину
2. Проверяется, что каждая вершина была посещена
3. Проверяется, что число вершин больше на единицу числа ребер в графе

Обоснование используемого алгоритма

Для проверки связности используется поиск в глубину из случайной (первой в списке) вершины, так как это самый быстрый алгоритм (временная сложность $O(V + E)$, где V – число вершин, E – число ребер) для нахождения всех вершин, достижимых из данной (это нужно найти, так как в связном неорграфе все вершины достижимы из любой вершины)

Обоснование используемой структуры данных

Для хранения графа используется список списков смежности, так как по сравнению с матрицей смежности эта структура данных занимает меньше места при хранении не сильно связных графов (если $E \ll V^2$).

Сложность используемого алгоритма

Для каждой вершины производится поиск в глубину и проверка на посещение всех вершин = $O(V) * (O(V + E) + O(V)) = O(2V^2 + E)$

Удаление вершины занимает $O(V + E)$

Весь алгоритм имеет временную сложность $O(2V^2 + E) + O(V + E) = O(2V^2 + 2E) = O(V^2 + E)$, где V – число вершин, E – число ребер.

Основные функции

Функция чтения графа из файла. Принимает указатель на граф и указатель на файловую переменную. Возвращает флаг успешности чтения.

```
bool read_graph(graph_t *graph, FILE *file)
```

Функция проверки графа на то, является ли он деревом. Принимает граф, номер вершины, которую нужно пропускать при проверке. Возвращает результат проверки.

```
bool is_tree(graph_t graph, int vertex_to_del)
```

Функция удаления вершины из графа. Принимает указатель на граф, номер вершины к удалению.

```
void delete_node(graph_t *graph, int number)
```

Функция экспорта графа в файл на языке DOT. Принимает граф и имя файла.

```
void show_graph(graph_t graph, const char *graph_name)
```

Функция поиска в глубину в графе. Принимает граф, номер вершины, которую нужно пропускать при поиске.

```
void depth_search(graph_node_t *graph, int vertex_to_del)
```

Функция освобождения графа. Принимает указатель на граф

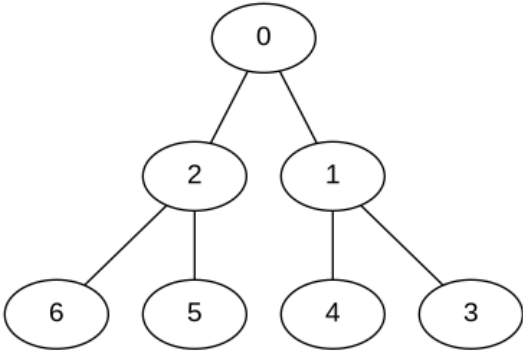
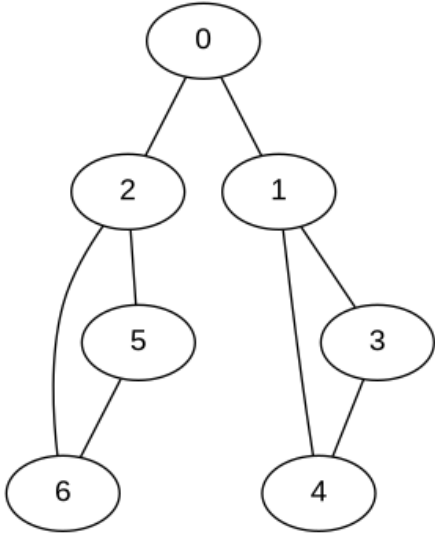
```
void free_graph(graph_node_t *graph)
```

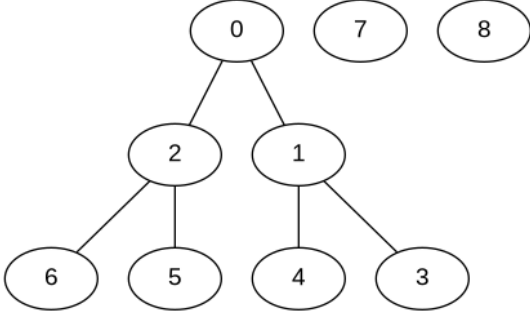
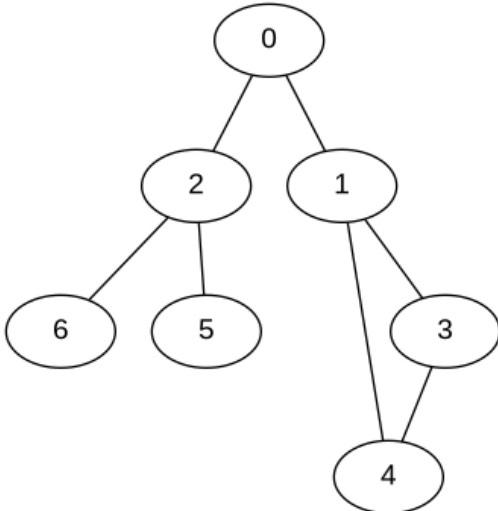
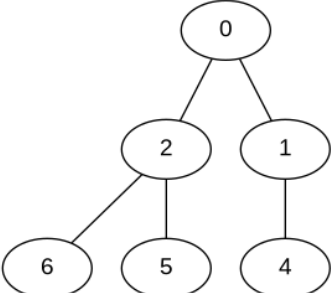
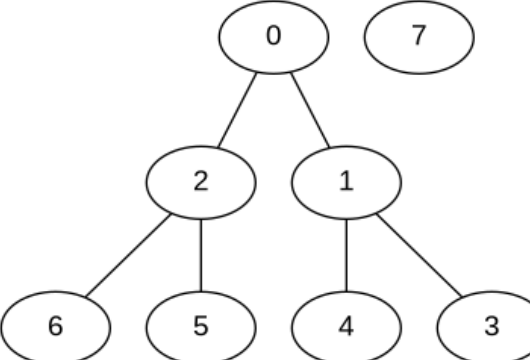
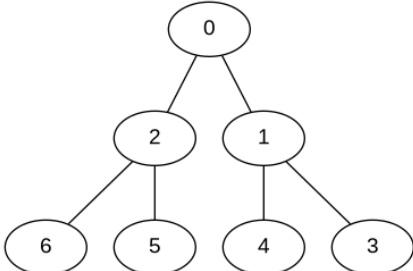
Набор тестов
Считывание графа

Описание теста	Содержимое файла	Вывод
Некорректное количество вершин в графе	0 1 0 1	Слишком мало вершин в графе!
Некорректное число ребер в графе	2 -1 0 1	Ошибка чтения файла!
Петля в графе	2 1 0 0	Петля невозможна в неорграфе!
Повторяющееся ребро	3 3 0 1 1 2 1 0	Ребро уже есть в графе!

Ребро с несуществующей вершиной	3 1 1 4	Ошибка чтения файла!
Граф гарантированно является деревом	1 0	Слишком мало вершин в графе!

Преобразование в дерево

Описание теста	Входной граф	Выход
Граф уже является деревом	 <pre> graph TD 0((0)) --- 2((2)) 0 --- 1((1)) 2 --- 6((6)) 2 --- 5((5)) 1 --- 4((4)) 1 --- 3((3)) </pre>	Граф уже является деревом!
В графе несколько циклов	 <pre> graph TD 0((0)) --- 2((2)) 0 --- 1((1)) 2 --- 5((5)) 2 --- 6((6)) 5 --- 6 1 --- 3((3)) 1 --- 4((4)) 3 --- 4 </pre>	Граф нельзя превратить в дерево удалением одной вершины

<p>В графе несколько изолированных вершин</p>		<p>Граф нельзя превратить в дерево удалением одной вершины</p>
<p>В графе один цикл</p>		<p>Граф можно превратить в дерево удалением вершины 3</p> 
<p>В графе одна изолированная вершина</p>		<p>Граф можно превратить в дерево удалением вершины 7</p> 

Вывод

При решении задач, которые требуют хранения данных с структурой, похожей на сеть, например сеть дорог между городами в стране, целесообразно использовать такую структуру данных, как граф.

Ответы на контрольные вопросы

1. Что такое граф?

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

2. Как представляются графы в памяти?

Граф обычно представляется в виде матрицы смежности или списка смежности

3. Какие операции возможны над графами?

Добавление и удаление вершин и ребер.

Поиск пути между двумя вершинами.

Поиск минимального остовного дерева.

Проверка связности графа

4. Какие способы обхода графов существуют?

Поиск в глубину, поиск в ширину

5. Где используются графовые структуры?

Моделирование компьютерных сетей, оптимизация транспортных сетей, поиск стратегий в играх, использование в конечных автоматах

6. Какие пути в графе Вы знаете?

Простой путь: Путь, в котором все вершины уникальны.

Цикл: Путь, который начинается и заканчивается в одной и той же вершине.

Кратчайший путь: Путь с минимальной суммарной длиной ребер между двумя вершинами.

Эйлеров путь: Путь, который проходит через каждое ребро ровно один раз.

Гамильтонов путь: Путь, который проходит через каждую вершину ровно один раз.

7. Что такое каркасы графа?

Каркасы графа, или остовные деревья, — это подграфы, которые соединяют все вершины графа без циклов.