Отчет по 7 лабораторной работе
По дисциплине «Типы и структуры данных»

Подготовил Пересторонин Павел
Группа ИУ7-33Б
Вариант 15

Цель работы

Реализовать алгоритмы обработки графовых структур:
поиск различных путей, проверку связности, построение остовых деревьев минимальной стоимости.

Техническое задание

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

Входные данные

Имя файла, в котором хранятся кол-во вершин в графе и ребра графа (каждое ребро хранится как пара вершин, которую оно соединяет)

Выходные данные

Информация, можно ли удалением вершины превратить граф в дерево или нет.

Возможные аварийные ситуации

Некорректное имя файла, некорректно заданые данные в файле.

Структуры данных

Узел для стека и списка смежности:

```
typedef struct node_t
{
   ntype data;
   struct node_t *next;
} node_t;
```

data — целочисленное значение
next — указатель на следующий элемент в списке.

Список смежности:

node_t **adj_list; - массив указателей на первый элемент списка смежности (где на i-ой позиции хранится указатель на список смежности для i-ой вершины)

Стек:

node_t *head; - указатель на вершину стека.

Алгоритм.

Алгоритм задачи можно разделить на 2 части:
1. Поиск в глубину с нахождением фундаментальных циклов и подсчет для каждой вершины количества фундаментальных циклов, в которые она входит.
2. Для каждой вершины, которая входит в максимальное количество фундаментальных чисел, проверка упрощенным поиском в глубину (то есть уже без нахождения фундаментальных циклов, а просто проверка связности и отсутствия циклов) является ли граф, порожденный старым без этой вершины связным и ациклическим (то есть деревом и при этом не лесом).

Тесты.

1. На вход поступает дерево.

```
Введите файл, для считывания данных: test_files/test1
Граф уже дерево.
```

2. Граф, где можно убрать несколько вершин.

```
Введите файл, для считывания данных: test_files/test2
Можете убрать вершину #0 и получите дерево
Можете убрать вершину #1 и получите дерево
Можете убрать вершину #2 и получите дерево
```

3. Граф, который нельзя преобразовать.

```
Введите файл, для считывания данных: data2.txt
Невозможно сделать преобразование в дерево путем удаления вершины
```

4. Граф, который можно преобразовать в дерево путем удаления одной вершины.

```
Введите файл, для считывания данных: data2.txt
Можете убрать вершину #0 и получите дерево
```

Расчеты памяти и времени.

N — кол-во вершин
M — кол-во ребер

Предельная длина цикла = N - 1 (объединяет все вершины), таким образом для хранения вершин, входящий в какой-либо фундаментальный цикл нужно выделить N по памяти (если делать это статически).
Далее, нужно выделить (M - 1) * 2 для хранения массива списков смежности для каждой вершины (каждое ребро хранится дважды, что уменьшает время доступа к нему, но увеличивает память).
Далее, нужно выделить место под массив на N элементов для хранения информации о том, проверена ли вершина или нет (для поиска в глубину).
Стек, для хранения вершин при поиске в глубину, размер = N.

Итог: при статическом хранении данных (что дает выигрыш по времени) требуется O(3N+2M) памяти или O(N+M).

Так как в данном случае я использую только один поиск в глубину, а далее для вершины, входящей в максимальное кол-во фундаментальных циклов я применяю его еще раз, то временная сложность моего алгоритма O(2(N+M)) или O(N+M).

Выводы по проделанной работе

В данной работе я столкнулся с задачей, для которой удалось найти алгоритм, не являющийся полным перебором и позволяющий найти удовлетворяющую нас информацию быстрее, чем полный перебор. Полный перебор занял бы у меня $O(N^2+NM)$ по времени, в то время как я добился O(N+M). Однако в моем случае пришлось хранить больше информации и я немного проиграл по памяти (величина порядка N).
Учитывая, что данная задача вполне может встретиться в реальной жизни (например, имеется город, в котором множество компаний ведут дела между собой, производя обмены продуктами, однако существует проблема, связанная с тем, что компании получают свои же продукты (то есть в связях есть циклы). Пусть решение этой проблемы — удаление одной из компаний с рынка. Тогда задача заключается в следующем: можно ли убрать с рынка одну компанию таким образом, чтобы все остальные компании могли все так же вести дела между собой (через посредников в том числе; то есть чтобы граф был связным), и при этом не получали бы свои же продукты (то есть чтобы граф был ациклическим)), мы достигли хорошего результата.

Контрольные вопросы

1. Что такое граф?

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

2. Как представляются графы в памяти?

Матрица смежности, список смежности.

3. Какие операции возможны над графами?

Поиск кратчайшего пути от одной вершины к другой (если он есть);

Поиск кратчайшего пути от одной вершины ко всем другим;

Поиск кратчайших путей между всеми вершинами;

Поиск эйлерова пути (если он есть);

Поиск гамильтонова пути (если он есть).

4. Какие способы обхода графов существуют?

Обход (поиск) в глубину и обход (поиск) в ширину.

5. Где используются графовые структуры?

Графовые структуры используются во многих жизненных ситуациях, в которых есть множество каких-то объектов, и они как-то между собой связаны (конечный автомат, карта метрополитена, навигация и др.)

6. Какие пути в графе Вы знаете?

Простой путь, замкнутый путь, гамильтоновый путь, эйлеров путь.

7. Что такое каркасы графа?

Подграф некоторого графа, содержащий все его вершины и при этом являющийся деревом.