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 **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

О Т Ч Е Т

По лабораторной работе № 8

Название: Графы

Дисциплина: Типы и структуры данных

Вариант: 5

Студент ИУ7-31Б (Группа) Савинова М. Г. (Фамилия И. О.)

Преподаватель Никульшина Т. А.

Москва, 2022

Оглавление

Цель работы

Реализовать алгоритмы обработки графовых структур:

- поиск различных путей;
- проверку связности;

Условие задачи

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

Описание ТЗ

1. Описание входных данных

Файл **graph.txt**, в котором записано кол-во вершин и список ребер.

2. Ограничения

- Вершина дерева: целое число от **0 до n-1** (n - кол-во вершин графа);
- Пункт меню: целое число **от 0 до 3;**

3. Описание выходных данных

Файлы **result_graph_*.gv**, (*-номер удаленной вершины) в которых записаны графы с удаленными вершинами; сообщение о возможности перехода графа в дерево с удаленной вершиной.

4. Описание задачи, реализуемой в программе

Программа реализует:

- формирование графа;
- удаление узла графа;
- поиск в глубину;
- подсчет кол-ва ребер;
- вывод графа в файл;

5. Способ обращения к программе

Обращение к программе происходит через консоль, путём запуска *app.exe* файла.

6. Описание возможных аварийных ситуаций и ошибок пользователя

Аварийная ситуация	Код завершения	Сообщение
Граф не инициализирован	1	`Нет доступного графа!`
В файле указано неверное кол-во вершин графа	3	`Неверное кол-во вершин графа!`
Ошибка при выделении памяти	4	`Ошибка выделения памяти!`
Граф при удалении вершины оказался несвязным	5	`Граф не связный!`
Граф при удалении вершины имеет циклы	6	`Граф имеет циклы!`
Выбран неверный пункт меню	7	`Неверно выбранный пункт меню!`
Не был выбран файл с данными	8	`Нет доступного файла!`
В входном файле отсутствуют данные	9	`Входной файл пуст!`

Описание внутренних структур данных

```
// структура для вершины графа
typedef struct node node_t;

struct node
{
    int vertex;      // номер вершины
    node_t *next;    // указатель на следующую вершину
};

// структура для описания графа
typedef struct
{
    int num_vertex; // кол-во вершин графа
    node_t **head;  // массив указателей вершин
} graph_t;
```

Описание алгоритма

Для представления графа в программе используется *список смежности*. Данные для заполнения матрицы смежности считываются из файла. Далее по полученному списку создается файл с расширением *.gv*, содержащий представление графа на языке *DOT*.

Пользователь выбирает *2* пункт меню, в результате которого происходит проверка всех вершин графа: сначала осуществляется *поиск в глубину*, затем подсчет кол-ва ребер и вершин. В результате полученных данных на экран выводится соответствующее сообщение. Так же каждый новый полученный граф записывается в отдельный файл с расширением *.gv* .

Тестовые данные

1. Негативные тесты

№	Наименование теста	Пользовательский ввод	Вывод
1.	Неверно выбранный пункт меню	ааа	Неверно выбранный пункт меню!
2.	В файле указано неверное кол-во вершин	(*кол-во вершин в графе: -10) 1	Неверное кол-во вершин графа!
3.	В файле указано некорректное значение вершины	(*зн-ие вершины в графе: -1) 1	Нет доступной вершины графа!
4.	Отсутствует входной файл	1	Нет доступного файла!
5.	Входной файл пуст	1	Входной файл пуст!
6.	Граф не связный (*результат в приложении)	1 2 3	Граф не связный!
7.	Граф имеет циклы	1 2 7	Граф имеет циклы!

2. Позитивные тесты

№	Наименование теста	Пользовательский ввод	Вывод
1.	Удаление вершины графа	1 2 4	Граф, с удаленной 4 вершиной, является деревом!
2.	Удаление вершины графа	1 2 8	Граф, с удаленной 4 вершиной графа, является деревом!

Выводы

В данной работе использовался способ хранения графа в виде **списка смежных вершин**, поскольку для проверки, является ли граф деревом необходимо было проверить 2 условия:

- граф является связным
- кол-во ребер меньше кол-ва вершин на 1

Первое условие можно легко проверить **поиском в глубину**. Эффективность этого алгоритма **при указанном методе хранения** графа определяется лишь общим кол-вом ребер графа. К примеру, для **матрицы смежности** при данном поиске необходимо пройти для каждой вершины по ее смежным вершинам. **Второе** условие проверяется простым проходом по массиву указателей.

*Ответы на контрольные вопросы

1. Что такое граф?

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

G = < V, E >,

где V – конечное непустое множество вершин; E – множество ребер (пар вершин).

2. Как представляются графы в памяти?

Граф в памяти представляется в виде *матрицы смежности* или *списка смежности.*

Матрица смежности A(n*n) – элемент A[i, j]=1, если существует ребро, связывающее вершины i и j, и =0, если ребра не существует.

Список смежностей содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с ней.

3. Какие операции возможны над графами?

- поиск кратчайшего пути от одной вершины к другой;
- поиск кратчайшего пути от одной вершины ко всем другим;
- поиск кратчайших путей между всеми вершинами;
- поиск эйлерова пути;

4. Какие способы обхода графов существуют?

Обход в ширину (BFS – Breadth First Search) — вершина обрабатывается путём просмотра сразу всех «новых» соседей этой вершины, которые последовательно добавляются в очередь просмотра.

Обход в глубину (DFS – Depth First Search) - начиная с некоторой вершины **v0**, происходит поиск ближайшей смежной ей вершины **v**, для которой так же осуществляется поиск в глубину до тех пор, пока не встретится ранее просмотренная вершина или не закончится список смежности вершины **v** (то есть вершина полностью обработана).

5. Где используются графовые структуры?

Графовые структуры могут использоваться в задачах, где между элементами могут быть установлены произвольные связи.
Распространенное применение — *решение задачах о путях.*

6. Какие пути в графе Вы знаете?

Эйлеровый путь - путь в графе, проходящий через каждое ребро ровно один раз. Если путь проходит по некоторым вершинам несколько раз – он называется непростым, иначе – простым.

Гамильтонов путь - простой путь, проходящий через каждую вершину ровно один раз.

7. Что такое каркасы графа?

Каркас графа – дерево, в которое входят все вершины графа, и некоторые (не обязательно все) его рёбра. Для построения каркасов графа используются алгоритмы Крускала и Прима.