Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

«Обработка очередей»

Студент Фролов Евгений

Группа ИУ7 – 35Б

2020 г.

Цель работы

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

Условие 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.

Определить, является ли связным заданный граф

Входные данные
Целое число - количество вершин в рассматриваемом графе:

Пара целых чисел, представляющая собой связи между вершинами (от 0 до V – 1, где V – количество вершин графа).

Выходные данные
Графическая визуализация полученного графа
Строка, которая информирует: связанный ли данный граф или нет
Возможные аварийные ситуации
Некорректный ввод.

Условие ввода кол-ва вершин: положительное целое число

Условие ввода связей вершин: диапазон целых чисел от 0 до V-1

Обращение к программе: через консоль командой ./main.exe.

Структура данных

В программе граф представляется в виде матрицы смежности B(n * n).

Матрица смежности:

```
typedef struct
{
   int size; //кол-во вершин в графе
   int **matrix; //указатель на массив указателей
} matrix_t;
```

Поиск в глубину

```
void dfs(const matrix_t matrix, const int vertex, int *visited){
    visited[vertex] = 1;

    for (int i = 0; i < matrix.size; i++)
        if (matrix.matrix[vertex][i] && !visited[i])
            dfs(matrix, i, visited);
}
```

Алгоритм

Вводятся данные, затем граф проверяется на связанность путем выполнения рекурсивного поиска в глубину для любой вершины. Если была посещена каждая вершина графа, то граф связанный.

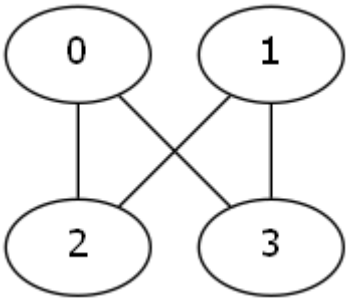
Алгоритм поиска описывается рекурсивно: перебираем все исходящие из рассматриваемой вершины ребра. Если ребро ведёт в вершину, которая не была рассмотрена ранее, то запускаем алгоритм от этой нерассмотренной вершины, а после возвращаемся и продолжаем перебирать рёбра. Возврат происходит в том случае, если в рассматриваемой вершине не осталось рёбер, которые ведут в нерассмотренную вершину. Если после завершения алгоритма не все вершины были рассмотрены, то необходимо запустить алгоритм от одной из нерассмотренных вершин. Если V – количество вершин графа, а E – количество рёбер, то алгоритм поиска в глубину имеет асимптотическую сложность O(V + E).

Тесты

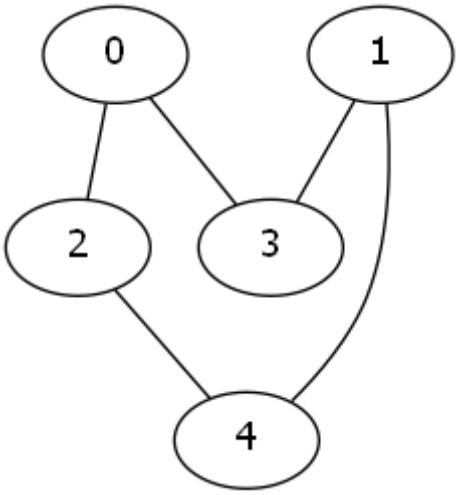
Некорректный ввод кол-ва вершин – не целое число/буква/символ

Некорректный ввод связи между вершинами – неправильно введен путь (2 2) либо элемент = V

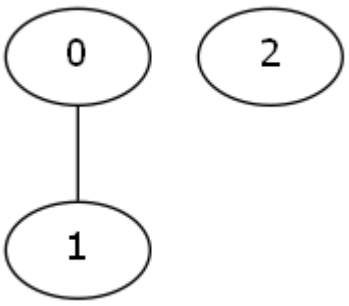
Тест: 4; 0 3 3 1 1 2 2 0 -1



Тест:5; 0 3 0 2 2 4 4 1 1 3 -1



Тест: 3; 0 1 -1



Оценка эффективности (в тиках)

Количество элементов	**Время выполнения**
5	60043
10	68573
15	72767

Занимаемая память (в байтах)

Количество элементов	**Занимаемая память**
5	144
10	488
15	1032

Вывод: в программе использовался алгоритм поиска в глубину, так как он прост в и удобен в реализации Если **V** – количество вершин графа, а **E** – количество ребер, то алгоритм поиска в глубину имеет асимптотическую сложность **O(V+E).**

Контрольные вопросы

1. Что такое граф?

Граф – конечное множество вершин и соединяющих их рёбер; **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?

Существуют различные методы представления графов в программе.

1. Матрица смежности B(n * n) – элемент b[i, j] = вес ребра, если существует ребро, связывающее вершины i и j, и = 0, если ребра не существует.
2. Список смежностей – содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с ней. Входы в списки смежностей могут храниться в отдельной таблице (в массиве), либо же каждая вершина может хранить свой список смежностей.

3. Какие операции возможны над графами?

Основные операции над графами: обход вершин и поиск различных путей: кратчайшего пути от вершины к вершине; кратчайшего пути от вершины ко всем остальным; кратчайших путей от каждой вершины к каждой; поиск эйлерова пути и гамильтонова пути, если таковые есть в графе.

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 (не обязательно все) его рёбра. Для построения каркасов графа используются алгоритмы Крускала и Прима.