**Министерство образования Российской Федерации**

**ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ УПРАВЛЕНИЯ**

Факультет: Прикладная информатика

Кафедра: Информационные технологии и системная аналитика

**ЯЗЫКИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ**

**Домашнее задание №1:**

«Алгоритмы на графе в Java»

Вариант Е22

**Преподаватель:**

Авдеев В.С.

**Студент**:

Чубченко Г. Е.

**Группа:**

ПИ 2-3

Москва 2024

**Задача**

Алгоритм нахождения цикла (Cycle detection) на графе представляет из себя прохождение по всему графу в поисках цикличности. Таким образом суть работы была в написании подобного алгоритма для неориентированного графа (рис. 1).

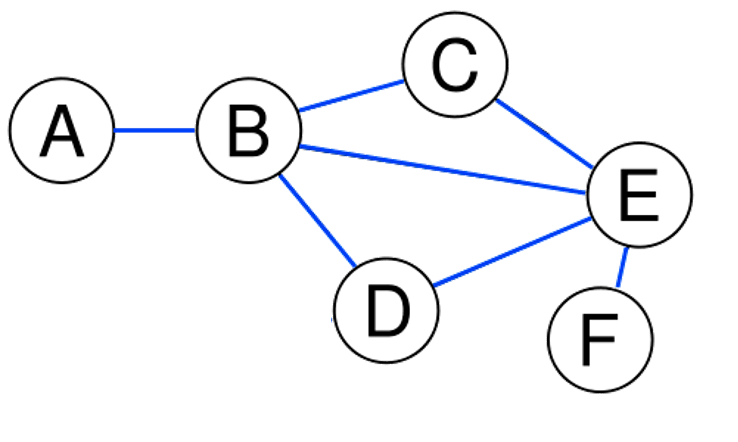


Рис. 1 Неориентированный граф (ребро = 1)

**Теоретическая часть**

Суть алгоритма в том, чтобы пробежаться по всему графу и при встрече с вершиной, в которой мы уже были вернуть в результате true. Если же после исследования графа все вершины будут уникальны – возвращаем false.

**Практическая часть**

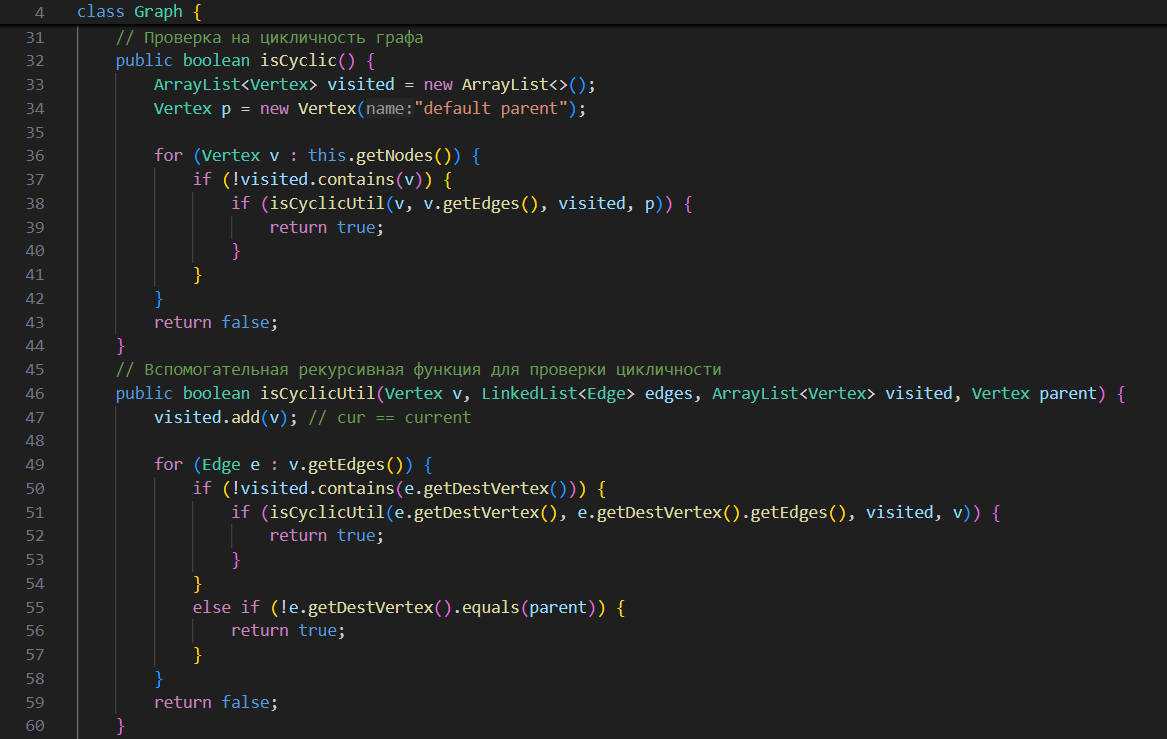
Весь алгоритм мы добавляем в уже имеющийся класс Graph (4 строка). 

Рис 2. Фрагмент программы, описывающий проверку цикличности графа

Первым делом мы создаем основной метод, который и будем вызывать для проверки цикличности (32 строка), никаких параметров он не принимает, а возвращает либо true (39 строка) – в случае цикличности и false (43 строка) – если циклов нет. Дальше мы создаем список объектов класса Vertex т.е. вершин (33) и вершину p с названием “default parent” которая сыграет роль костыля далее (34). Организуем цикл, который будет брать за начальную точку каждой итерации каждую вершину нашего графа (35), (это не необходимо, что я понял уже в процессе написания отчёта, т.к. мы из любой точки неориентированного графа можем попасть в любую другую) и в нём будем проверять, встречается ли цикл при движении от этой вершины во все возможные стороны (38), но перед этим проверяем, были ли мы в этой вершине при проверке на цикличность (37), это понадобится в будущем. Подробнее рассмотрим функцию isCyclicUtil() принимающий параметры: v – наша стартовая вершина, v.getEdges() – список всех ребер данной вершины, visited – АДРЕС на память, где будут храниться посещённые нами вершины, p – наша заглушка для головного цикла.

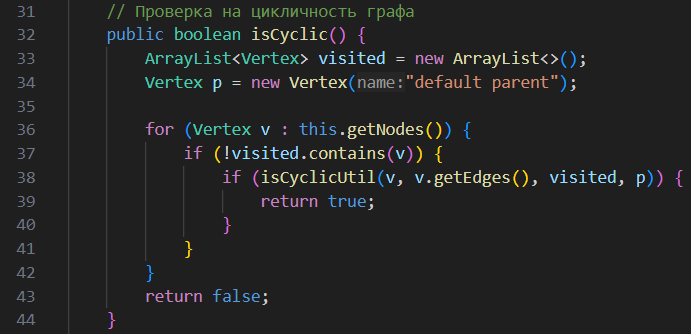


Рис. 3. Метод проверки на цикличность графа

Вспомогательная функция isCyclicUtil (46) принимает в качестве параметров Vertex v – вершина графа, LinkedList<Edge> edges – адрес на список ребер некоторой вершины, ArrayList<Vertex> visited – (причина моих небольших страданий) АДРЕС на созданный ранее список посещённых вершин, который мы будем заполнять по ходу выполнения функции, Vertex parent – вершина откуда мы пришли в v вершину (необходим для того, чтобы мы не двигались в обратную сторону при прохождении по графу). Возвращает же true – в случае, если в ходе рекурсии был найден цикл, и false – в противном случае соответственно.

Первым делом мы добавляем в список посещенных вершин текущую (47). Далее мы организовываем цикл, в котором проходимся по всем ребрам текущей вершины (49). В 50 строке мы проверяем, посещали ли мы уже вершину, с которой связана текущая. Если следующая вершина уже была посещена, и она не является родительской, т.е. той, от которой мы пришли, то мы обнаружили цикл, следовательно, возвращаем true (55-56). Если же ни одно из этих условий не выполнено – переходим к следующему ребру.

Подробнее рассмотрим проверку в 50 строке: если следующая вершина не была ранее посещена, значит мы можем двигаться дальше, образуя рекурсию (51), которая принимает следующие параметры: e.getDestVertex() – следующая вершина относительно текущей v связанная ребром e, e.getDestVertex().getEdges() – список ребер следующей вершины, visited – АДРЕС на память, где уже храниться некоторая информация о посещенных вершинах, v – текущая вершина, родительская, т.е. та, от которой мы пришли.

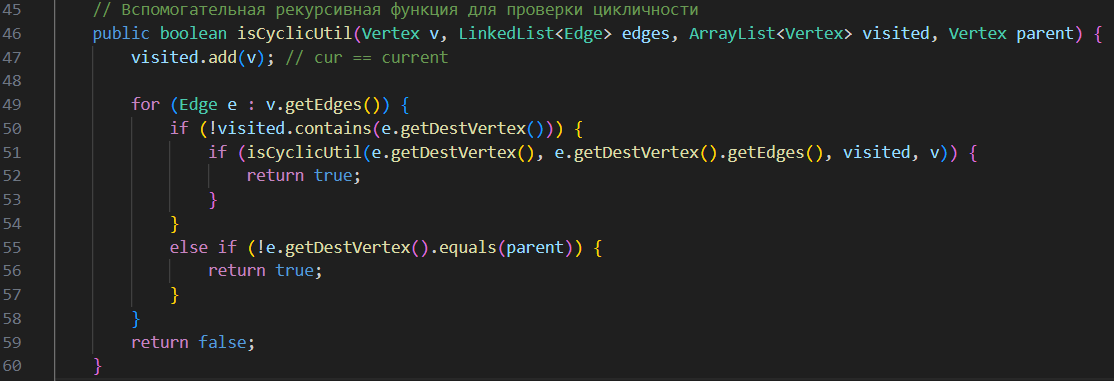


Рис. 4. Вспомогательная рекурсивная функция для проверки цикличности

**Вывод**

В итоге всех наших манипуляций, тестов и отладок получаем верный для любого ненаправленного графа алгоритм, который выводит «Граф цикличен» если он таковым является, и «Граф не цикличен» если не является соответственно (рис. 5-7).

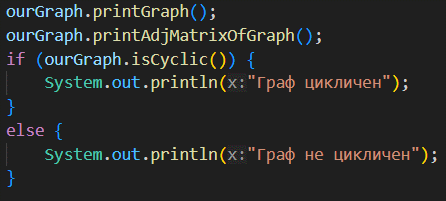


Рис. 5. Оформление вывода

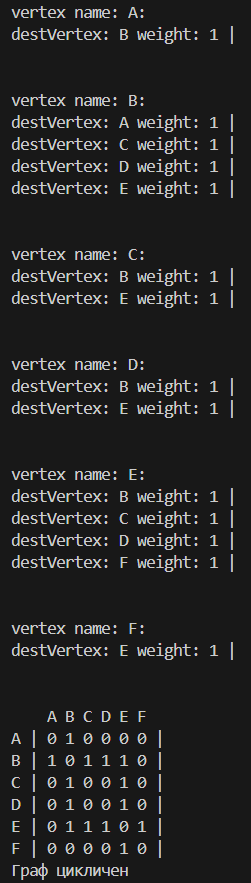


Рис. 6. Полученный результат

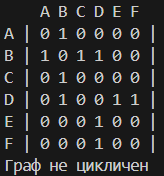


Рис. 7. Пример нецикличного графа A – B( C ) – D( E F )