Algotithms for Graph Isomorphism problem

Фёдор Букреев

5 декабря 2019 г.

Содержание

1 Постановка задачи

Перед формулировкой задачи необходимо ввести базовые определения:

1.1 Основные определения

Definition 1. *Простым неориентированным графом*

2 Частные случаи

2.1 Планарный граф

Задача проверки двух планарных графов на изоморфизм не просто лежит в P, но решается за линейное время(в модели с произвольным доступом) [1] и лежит в классе Ł[2].

Definition 2. *Планарным называется граф, который можно изобразить на плоскости так, чтобы никакие его два ребра не пересекались.*

Definition 3. *Плоским назовём граф, изображенный на плоскости*

Definition 4. *Граф k-связен, тогда и только когда он имеет больше чем k вершин и после удаления менее чем k любых вершин граф остаётся связным.*

Я приведу первый алгоритм в этой области, созданный для несколько меньшего класса графов - для планарных 3-связных графов[3], созданный Lois Weinberg. Этот алгоритм был улучшен и обобщён (Hopcroft и Tarjan [6]) до алгоритма, работающего за $O(n \cdot logn)$ для **произвольного** планарного графа. Затем данный алгоритм был ускорен до линейного времени (Hopcroft and Wong [1]). Что же особенного в планарных 3-связных графах? Дело в том, что в то время, как в общем случае порядок группы автоморфизмов графа может быть равен $|V|!$, доказано[4], что размер группы автоморфизмов для планарных 3-связных графов равен $4|E|$. Это позволяет существенно сократить перебор вариантов.

Предисловие к алгоритму

Алгоритм, описанный ниже опирается на несколько теоретических утверждений о простых планарных 3-связных графах, которые следует упомянуть до его описания:

1) Согласно [5], такой граф имеет уникальное представление на плоскости. Стоит отметить, что именно подразумевается под уникальностью. Имеется ввиду, что его грани, ограниченные рёбрами

уникальны в смысле набора рёбер, которые их задают. То есть "зеркальный"граф, полученный обращением циклов граней, тоже может являться представлением исходного графа на плоскости.

2) Как было отмечено выше, в работе [4] показано, что порядок группы автоморфизмов такого графа не превосходит $4|E|$

Алгоритм

На вход алгоритм получает простой *плоский* ненаправленный 3-связный граф $G = (V, E)$. В случае компьютерной программы удобно считать, что полученный граф - это не изображение на плоскости, а структура данных, где каждой вершине сопоставлен набор рёбер в порядке обхода против часовой стрелки таким образом, что изображение данного графа не имеет пересекающихся рёбер.

Каждому автоморфизму графа мы сопоставим некоторый однозначно задающий его код, сгенерированный во время обхода графа по эйлеровому циклу. Для того, чтобы гарантировать наличие эйлерового цикла мы продублируем каждое ребро $v \to v_1, v_2$ в графе, и выдадим полученым парам ребер противоположные направления. То есть из ребра $\{a, b\}$ мы получим направленные рёбра $(a, b), (b, a)$, $a, b \in V$. В полученном ориентированном графе G' входящая степень любой вершины равна её исходящей степени, а значит эйлеров цикл существует. Обойдём данный граф по следующему правилу:

0) Все вершины графа разделим на 2 типа: *старые* и *новые*, изначально все вершины графа - новые. Выберем начальное ребро и направление на нём в графе или его зеркальной копии. Это $2 \cdot 2|E|$ вариантов. Первой вершиной обхода станет первая вершина ребра, согласно выбранному направлению, второй - вторая.

1) Попадая в *новую* вершину, отмечаем её как старую и двигаемся по ребру, имеющему минимальный от нашего угол в направлении против часовой стрелки. В случае компьютерной программы, это будет следующее после того, по которому мы пришли к вершине, ребро в наборе.

2) Попадая в *старую* вершину по ребру, противоположное направление которого ещё не было посещено, возвращаемся по противоположному ребру.

3) Попадая в *старую* вершину по ребру, обратное которому уже было пройдено, двигаемся по ребру, имеющему минимальный от нашего угол в направлении против часовой стрелки и при этом непосещённому ранее.

Во время данного обхода каждый раз встречая *новую* вершину, присваиваем ей номер. То есть начальной вершине присвоен номер 1, второй - номер 2 и так далее. Попадая в очередную вершину, после присвоения номера(если оно было необходимо), добавляем в код текущего обхода номер посещённой вершины. Таким образом, для каждого обхода получим код длины $2|E| + 1$ который удобно представить ввиде вектора.

После повторения процедуры для всех $4|E|$ обходов, получим таблицу кодов размера $4|E| \times 2|E| + 1$. Затем сортируем все полученные коды в лексикографическом порядке. Сгенерировав(и отсортировав) такую матрицу для обоих графов, вопрос об изоморфизме сводится к проверке равенства первых столбцов в этих матрицах, то есть графы изоморфны тогда и только тогда, когда их лексикографически минимальные коды совпадают.

Послесловие к алгоритму

Стоит отметить, что если графы оказались изоморфны, то обратив процедуру обхода, описанную в алгоритме можно получить нумерацию вершин в каждом графе, сответствующую изоморфизму.

Алгоритм так же опирается на следующие теоретических утверждения:

1) Простые 3-связные планарные графы G_1, G_2 изоморфны тогда и только тогда, когда множества их кодов совпадают

2) Простые 3-связные планарные графы G_1, G_2 изоморфны тогда и только тогда, когда выполнено одно из двух условий:

a) Первые строки матриц их кодов(минимальные в лексикографическом смысле коды) совпадают

b) Любые 2 строки их матриц совпадают

Отмечу, что второе утверждение позволяет сократить количество вычислений в большом количестве случаев.

Кроме того, на основе данного алгоритма можно проверять на изоморфизм и ориентированные графы. Необходимые для этого модификации описанны в оригинальной статье [3].

3 Общий алгоритм

Список литературы

[1] Hopcroft, John; Wong, J. (1974), "Linear time algorithm for isomorphism of planar graphs Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, pp. 172–184, doi:10.1145/800119.803896.

[2] Datta, S.; Limaye, N.; Nimbhorkar, P.; Thierauf, T.; Wagner, F. (2009), "Planar graph isomorphism is in log-space 2009 24th Annual IEEE Conference on Computational Complexity, p. 203, arXiv:0809.2319, doi:10.1109/CCC.2009.16, ISBN 978-0-7695-3717-7.

[3] Lois Weinberg, "A Simple and Efficient Algorithm for Determining Isomorphism of Planar Triply Connected Graphs"

https://ieeexplore.ieee.org/document/1082573/

[4] Lois Weinberg, "On the Maximum Order of the Automorphism Group of a Planar Triply Connected Graph"

https://epubs.siam.org/doi/abs/10.1137/0114062

[5] H. Whitney, "2-isomorphic graphs Amer. J. Math., 55 (1933) 245–254,

https://www.jstor.org/stable/2371127?seq=1

[6] E. Hopcroft and R.E. Tarjan. "Efficient planarity testing". Journal of the ACM, 21, 1974.

[7] placeholder

[8] placeholder

[9] placeholder