

Алгоритмы для проверки графов на изоморфизм

Фёдор Букреев

20 декабря 2019 г.

Содержание

1	Постановка задачи	1
1.1	Основные определения	1
1.2	Задача	1
2	Частные случаи	1
2.1	Планарные графы	1
2.2	Теория Бабаи-Лакса	3
2.3	Графы с ограниченной кратностью цветов	3
2.4	Графы с ограниченной кратностью собственных значений	3
3	Общий алгоритм	4

1 Постановка задачи

Перед формулировкой задачи необходимо ввести базовые определения:

1.1 Основные определения

Definition 1. Простым неориентированным графом G назовём пару множеств V, E , где V - это множество вершин графа, а E - множество неупорядоченных пар вершин, называемых рёбрами. Будем считать, что две вершины $a, b \in V$ соединены (смежны) тогда и только тогда, когда $\{a, b\} \in E$.

Definition 2. Граф G назовём связным, если для любых вершин $a, b \in V$ существует путь по рёбрам графа из a в b .

1.2 Задача

Необходимо проверить, изоморфны ли два данных графа G_1, G_2

Definition 3. Графы $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ изоморфны тогда и только тогда, когда существует отображение $f : V_1 \rightarrow V_2$ такое, что $a, b \in V_1$ смежны тогда и только тогда, когда вершины $f(a), f(b) \in V_2$ - смежны

2 Частные случаи

2.1 Планарные графы

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 $L[2]$. Отмечу, что дерево является частным случаем планарного графа.

Definition 4. Планарным называется граф, который можно изобразить на плоскости так, чтобы никакие два его ребра не пересекались.

Definition 5. Плоским назовём граф, изображённый на плоскости

Definition 6. Граф k -связен, тогда и только тогда 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 $O(n \cdot \log n)$ 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 отражением 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 $v \rightarrow v_1, v_2$ в графе, и выдадим полученным парам рёбер 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 в виде 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, соответствующую 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:

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

б) 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 описаны в оригинальной статье [3].

2.2 Теория Бабаи-Лакса

Для нескольких алгоритмов ниже потребуются общие определения и утверждения

Пусть G - группа перестановок, действующая на $\{1, \dots, n\}$, а G_m - это подгруппа G , оставляющая $\{1, \dots, m\}$ на месте. Мы получили *башню групп* $G = G_0 \geq G_1 \geq \dots \geq G_n = id$.

Пусть K_m - система представителей левых смежных классов G_m по G_{m+1} . Каждый $g \in G_m$ можно однозначно представить как $g = k_m k_{m+1} \dots k_{n-1}$, где $k_i \in K_i$. Такое представление элемента назовём *каноническим*. Систему образующих $K = \cup K_i$ назовём *сильной*. Из соображения $|G_m : G_{m+1}| \leq n - m$ видно, что $|K| \leq n^2$.

Фурст, Хопкрофт и Лакс [нужна ссылка] предложили алгоритм, строящий сильную систему образующих по произвольной системе образующих за полиномиальное время.

ТУТ ДОЛЖНО БЫТЬ ОПИСАНИЕ АЛГОРИТМА (КАСКАД)

2.3 Графы с ограниченной кратностью цветов

2.4 Графы с ограниченной кратностью собственных значений

Definition 7. Матрицей смежности графа $G = (V, E)$, $|V| = n$ назовём матрицу A размера $n \times n$, где $A[i][j] = 1$ если $(i, j) \in E$, 0 иначе.

Definition 8. Собственным значением графа назовём собственное значение его матрицы смежности.

Definition 9. Граф G назовём графом кратности m в смысле собственных значений, если ни одно из собственных значений графа не имеет кратности превышающей m .

Definition 10. Группой автоморфизмов графа $G = (V, E)$ назовём набор перестановок V , которые сохраняют рёбра.

Предисловие к алгоритму

В данном алгоритме может потребоваться вычислять и сравнивать числа с плавающей точкой (координаты собственных векторов и собственные значения). Достаточно вычислять их с точностью до n^c . Подробнее это описано в [10]

Пусть $\{e_1, \dots, e_n\}$ - стандартный базис R^n . Сопоставим каждому базисному вектору вершину нашего графа в соответствии с нумерацией вершин в матрице смежности (i-я вершина соответствует i-му вектору). Тогда группа автоморфизмов G порождает ортогональные линейные преобразования R^n (перестановки базисных векторов). Получается, что группа автоморфизмов графа - это все такие матрицы перестановок π (и только они), что π коммутирует с матрицей смежности графа ($\pi A = A\pi$).

Понятно, что в общем случае размер группы перестановок G может достигать $n!$. Однако группа перестановок графа G может быть представлена как замыкание множества перестановок размера не более n^2 [9]. Это множество мы назовём порождающим.

Данный алгоритм описан в [8] и опирается на следующие теоретические утверждения (если не указано иного, то утверждение доказано в [8]):

Lemma 1. Проблемы проверки наличия изоморфизма между графами и нахождение порождающего множества $G_1 \sqcup G_2$ полиномиально сводимы друг к другу.

Доказательство леммы выше в [7]

Lemma 2. Пусть G - граф с матрицей смежности A . Матрица перестановки π является автоморфизмом G тогда и только тогда, когда собственные подпространства A инвариантны относительно π (если $x \in S_i$, то $\pi x \in S_i$)

Алгоритм

Нахождение порождающего множества группы перестановок

Данный пункт состоит из двух частей. Результатом первой части станет сведение проблемы нахождения порождающего множества группы перестановок, к проблеме нахождения группы автоморфизмов "графа с ограниченными цветовыми группами т.е. множество вершин графа разделено на $\sqcup C_i, i \in \{1, \dots, s\}$, даны представленные явно подгруппы $H_i \leq \text{Sym}(C_i)$, действующие на соответствующие C_i , и мы хотим получить порождающее множество для $\text{Aut}(G) \cap (H_1 \times \dots \times H_s)$. Затем полученная задача сводится к "башне групп для которой существует полиномиальное решение.

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] Rudolf MATHON (March 1979), "A note on the graph isomorphism counting problem"
- [8] Babai, László; Grigoryev, D. Yu.; Mount, David M. (1982), "Isomorphism of graphs with bounded eigenvalue multiplicity Proceedings of the 14th Annual ACM Symposium on Theory of Computing, pp. 310–324, doi:10.1145/800070.802206, ISBN 0-89791-070-2.
- [9] Merrick Furst ; John Hopcroft ; Eugene Luks, "Polynomial-time algorithms for permutation groups Published in 21st Annual Symposium on Foundations of Computer Science (sfcs 1980)
- [10] Leighton F. T.; Miller G. L., "Numerical analysis of Gaussian elimination and eigenspace calculation in preparation
- [11] placeholder
- [12] placeholder
- [13] placeholder