УДК 519.178; 512.53

***Зяблицева Лариса Владимировна****,*

*кандидат физико-математических наук, доцент кафедры математического анализа, алгебры и геометрии Северного (Арктического) федерального университета имени М.В. Ломоносова. Автор 27 научных публикаций, в т. ч. одной монографии и трех учебных пособий*

***Пестов Сергей Алексеевич****,*

*студент Северного (Арктического) федерального университета имени М.В.Ломоносова.. Автор 2 научных публикаций*

**Алгоритм проверки изоморфизма полурешеток с использованием инвариантов теории графов**

В статье рассматриваются коммутативные идемпотентные полугруппы (полурешетки), и способы проверки их изоморфизма с помощью алгоритмов теории графов. Проблема изоморфизма является одной из важнейших проблем теории полугрупп. Эта проблема состоит в существовании алгоритма, распознающего для любых двух полугрупп из данного класса, изоморфны они или нет. Аналогичная проблема есть и в теории графов, причем там этот вопрос для некоторых классов графов решен, например, для деревьев и планарных графов.

В статье описано, как для полурешеток можно найти соответствующий им граф. В том случае, когда полученный граф является деревом, для проверки изоморфизма можно применить известные алгоритмы проверки изоморфизма деревьев. В статье сформулирован и доказан критерий того, в каком случае граф полурешетки является деревом.

Далее обосновывается выбор алгоритма проверки изоморфизма деревьев, описан этот алгоритм, представлена программа, написанная на языке *Haskell*, реализующая его.

Для того, чтобы применить выбранный алгоритм для проверки изоморфизма полурешеток, необходимо сначала полурешетке сопоставить дерево. Для этого авторами разработан и реализован также на языке *Haskell* необходимый алгоритм.

Созданная в итоге программа для двух полурешеток, заданных таблицами Кэли, работает следующим образом: она выводит структуру соответствующих полурешеткам деревьев, каноническое имя полученных деревьев, проверяет изоморфизм деревьев, а значит и полурешеток. Надо заметить, что выбор и реализация алгоритмов являются эффективными, программа в течение нескольких секунд определяет изоморфизм полурешеток с трехзначным числом элементов.

Граф произвольной полурешетки не обязательно является деревом, не обязательно планарный. В этом случае для графа полугруппы предлагается система инвариантов, при их совпадении у графов двух полугрупп, производится проверка изоморфизма, при котором вершинам одного графа сопоставляются вершины другого, для которых все показатели совпадают.

***Ключевые слова и фразы:*** *полугруппы, графы,**изоморфизм графов, изоморфизм полугрупп, полурешетки, деревья, планарные графы.*

В статье [1] рассмотрено, как известные алгоритмы теории графов можно применить для проверки изоморфизма коммутативной полугруппы идемпотентов (полурешетки). Для этого для полурешеток строится соответствующий им граф. В том случае, когда полученные графы являются деревьями, для проверки их изоморфизма можно применить алгоритмы проверки изоморфизма деревьев. В статье сформулирован и доказан критерий того, в каком случае граф полурешетки является деревом.

**Предложение 1.** Представление коммутативной полугруппы идемпотентов *(S, ∙)* в виде графа является деревом тогда и только тогда, когда частичный порядок, заданный формулой:

*e  f*  *e ∙ f = f ∙ e = e*,

является полулинейным снизу.

Далее обосновывался выбор алгоритма проверки изоморфизма деревьев, описан этот алгоритм, представлена программа, написанная на языке Haskell, реализующая его. Для того, чтобы применить выбранный алгоритм для проверки изоморфизма полурешеток, необходимо сначала полурешетке сопоставить дерево. Для этого авторами разработан и реализован также на языке Haskell необходимый алгоритм.

Описанная в [1] программа для двух полурешеток, заданных таблицами Кэли, работает следующим образом: она выводит структуру соответствующих полурешеткам деревьев, каноническое имя полученных деревьев, проверяет изоморфизм деревьев, а значит и полурешеток.

В представленной статье продолжена работа по созданию алгоритма проверки изоморфизма для произвольных полурешеток, то есть полурешеток, графы которых могут и не быть деревом.

Рассмотрены вопросы о том, всегда ли граф полурешетки планарный, можно ли применить алгоритм проверки изоморфизма полурешеток, описанный в статье [1], для произвольных полурешеток. Описана система инвариантов для полурешеток и решен вопрос о полноте этой системы инвариантов. Итогом работы является описание и реализация алгоритма проверки изоморфизма произвольных полурешеток.

***Алгоритм кодирования деревьев строками***

В статье [1] описан алгоритм (назовем его алгоритм *A)*, с помощью которого граф, являющийся деревом, переводится в строку из нулей и единиц. Для этого алгоритма представлена его программная реализация на языке *Haskell*.

*Входные данные:* дерево *T* и вершина *r*, являющаяся корнем дерева *T*.   
*Результатом* выполнения алгоритма будет строка. Обозначим результат выполнения алгоритма *A* для дерева *T* с корнем *r* так: *A (T, r)*.  
*Шаг 1.* Если дерево состоит всего из одной вершины, то сопоставим ему строку "*01*".  
*Шаг 2.* Если в дереве более одной вершины, то рассматриваем вершины, являющиеся детьми корня дерева*: r1, …, rn.* Для каждой из этих вершин строим деревья *T1, …, Tn*, имеющие корнями вершины *r1, …, rn* .

*Шаг 3.* Для каждого из полученных деревьев *T1, …, Tn* запускается алгоритм *A*.

*Замечание.* В результате первых трех шагов всем концевым (висячим) вершинам дерева сопоставляется строка "*01*".

*Шаг 4.* Находим вершины дерева, расстояние от которых до корня максимально и которым последовательность еще не сопоставлена.

*Шаг 5.* Пусть *u* – такая вершина. Для вершины *u* находим всех детей. Для этих детей ранее были найдены соответствующие им последовательности *S1, …, Sl,.*   
*Шаг 6.* Строки *S1, …, Sl* располагаем в лексикографическом порядке (как слова в словаре). Получаем упорядоченный набор строк *s1, …, sl.* Тогда вершине *u* сопоставим строку: "*0s1...sl 1".*

*Шаг 7.* Если *u* – корень дерева *T,* то алгоритм останавливаем. Если нет, то переходим к шагу 4.

Результатом работы алгоритма будет строка *0s1...sn 1, где s1, …, sn* – последовательности, которые сопоставлены детям корня *r* дерева *T.*

Данный алгоритм предназначается для деревьев. Но реализован он таким образом, что программа применима также и для полурешеток, графы которых деревьями не являются. На таких графах программа ищет все пути, ведущие от корня до концевых вершин дерева, при этом одна и та же вершина может появиться в структуре графа несколько раз. При этом, если программа применяется к графам изоморфных полурешеток, то канонические имена будут совпадать, если же полурешетки не изоморфны, то канонические имена могут совпадать. Приведем пример таких полурешеток.

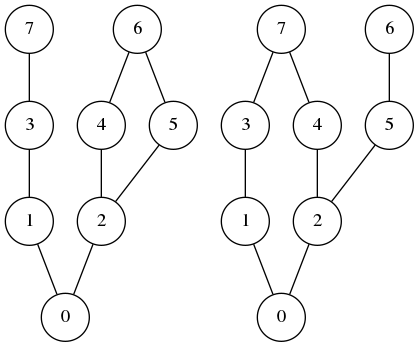


Рис. 1. Пример неизоморфных полурешеток, для которых совпадают канонические имена.

***Решение вопроса, является ли граф полурешетки планарным***

Кроме деревьев, еще одним интересным классом графов, для которых вопрос проверки изоморфизма является решенным, являются планарные графы [7]. Поэтому стоит выяснить вопрос о планарности графа произвольной полурешетки.

Известен критерий планарности графа: граф планарен тогда и только тогда, когда в нем отсутствуют подграфы, гомеоморфные одному из графов *К5* или *K3,3*.

Приведем пример полурешетки, граф которой не является планарным.

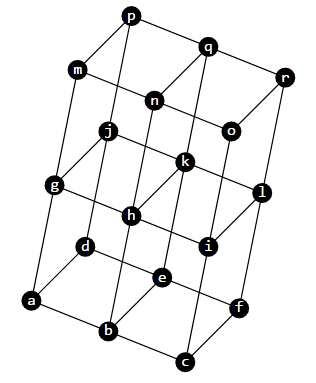


Рис. 2. Полурешетка, граф которой не является планарным.

В соответствующей такому графу полурешетке элемент *c* является точной нижней гранью, элемент *p* – точной верхней гранью. Изображенный на рисунке граф планарным не является, так как есть подграф этого графа, гомеоморфный двудольному графу K33. В одной доле этого графа расположим вершины *n, j, k*, в другой – вершины o*, q, h*.

Заметим, что при удалении всего одного ребра *(h, k)* в указанном графе он становится планарным.

***Инварианты для графа, изображающего полурешетку***

Предварительно рассмотрим для графа следующую систему инвариантов. Будем называть графы, изображающие полурешетки, *s*-графами.

1. Число вершин графа.

2. Число ребер графа.

3. Следующий инвариант является упорядоченной последовательностью, составленной следующим образом.

Пусть *v* является вершиной *s*-графа. Пусть этой вершине в полурешетке соответствует элемент *a*. Вершине *v* сопоставим три числа и одну последовательность. Опишем их.

а) Первое число – длина кратчайшей цепи от вершины до корня.

б) Второе число – число вершин графа, удовлетворяющих следующему условию: им соответствует в полурешетке элемент *b*, такой, что *b = inf (a, b)* и нет такого элемента *c*, отличного от *a* и *b*, что *с = inf (a, с)* и *b = inf (с, b)*. (Число вершин, непосредственно предшествующих вершине *v*).

в) Третье число – число вершин графа, удовлетворяющих следующему условию: им соответствует в полурешетке элемент *b*, такой, что *а = inf (a, b)* и нет такого элемента *c*, отличного от *a* и *b*, что *a = inf (a, с)* и *c = inf (с, b)*. (Число вершин, непосредственно следующих за вершиной *v*).

г) Сопоставим вершине упорядоченную по возрастанию последовательность длин цепей от вершины *v* до всех концевых вершин. Такая последовательность всегда будет состоять из *t* элементов, где *t* – это число концевых вершин. Если вершина концевая, то один из элементов последовательности будет равен 0.

В результате каждой вершине *v* графа сопоставляется последовательность следующего вида*: k, p, r, (f1, …, ft).* Упорядочим последовательности, сопоставленные вершинам графа лексикографически, по возрастанию, начиная с первого числа.

В итоге получим последовательность: *k1, p1, r1, (f11, …, f1t); k2, p2, r2, (f21, …, f2t);…; km, pm, rm, (fm1, …, fmt) (*здесь *m* – число вершин графа).

Полученная в результате последовательность будет являться инвариантом графа. Назовем его *инвариант 3*.

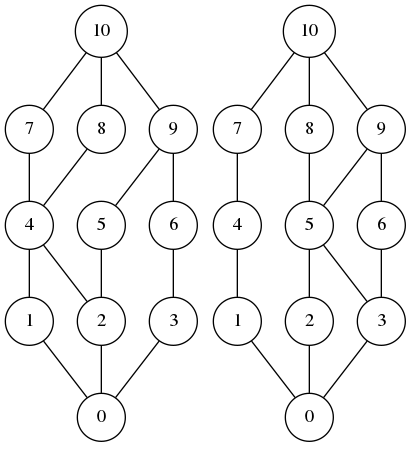
 Данная система инвариантов полной не является. Приведем пример неизоморфных полурешеток, для которых последовательности совпадают, кроме того, проверка для них алгоритмом, созданным для деревьев, также дает одинаковый результат.

Рис. 3. Графы неизоморфных полурешеток, которым соответствуют одинаковые инварианты 1,2,3 и совпадают канонические имена.

***Алгоритм проверки изоморфизма s-графов***

Далее описан алгоритм, с помощью которого для любых двух коммутативных полугрупп идемпотентов (полурешеток) можно определить, изоморфны ли они.

*Входные данные:* две полурешетки *S* и *S’*.   
 *Результатом* выполнения алгоритма будет ответ “полурешетки не изоморфны” или “полурешетки изоморфны”. Также в случае изоморфизма выдается биективное отображение элементов одной полурешетки в другую.

1. Находим для для полурешеток *S* и *S’* соответствующие им графы так, как это было описано в [1].
2. Сравниваем число вершин и ребер в графах. Если они не совпадают для двух графов, то ответ: полурешетки не изоморфны. Если они совпадают, то переходим к шагу 3.
3. Если графы иявляются деревьями, то применяем ранее описанный алгоритм проверки изоморфизма деревьев.
4. Если графы не являются деревьями, то преобразуем каждый из них в два графа и *,* и *.* Опишем, как строятся графыи *.* Рассмотрим максимальные подграфы графа *G*, имеющие корнем вершину, являющуюся корнем графа *G* и лишь одну вершину, смежную с корнем*.* Будем называть такие подграфы ветвями s-графа. Смотрим, нет ли в пересечении каких- нибудь из ветвей вершины, кроме корня. Если есть, то добавляем к одному из них ребра и вершины другого, которых нет в первом. Полученный в результате подграф также будем называть ветвью графа. Граф преобразуем в два графа и , корень графа *G* становится единственной общей вершиной и корнем этих подграфов. Граф является связным подграфом графа G, который содержит ветви графа, не содержащие циклов. Граф является связным подграфом графа *G*, который состоит из ветвей, содержащих циклы.

Аналогичным образом получаем графы и из графа *.*

1. Проверяем на изоморфизм графы *и* , которые являются деревьями. Если графы не являются изоморфными, то ответ: полурешетки не изоморфны. Если графы изоморфны, то переходим к шагу 6.
2. Рассматриваем графы и *.* Сначала проверяем их алгоритмом для деревьев. Если канонические имена графов совпадают, то переходим к шагу 7. Если нет, то ответ: полурешетки не изоморфны.
3. Находим графов и инварианты 3. Сравниваем полученные последовательности. последовательности не совпадают для двух графов, то ответ: полурешетки не изоморфны. Если они совпадают, то переходим к шагу 8.
4. Ищем изоморфизм графов. При этом сопоставляем вершинам одного графа вершины другого, если у них совпадают последовательности. В итоге перебор возникает только для тех вершин, для которых совпадает инвариант 4. Если изоморфизм найден, то ответ: полурешетки изоморфны, если нет, то ответ: полурешетки не изоморфны.

Список литературы

1. Л. В. Зяблицева, С. Ю. Корабельщикова, И. Н. Попов. Некоторые специальные полугруппы и их гомоморфизмы. – Архангельск: ИПЦ САФУ, 2013. – 128 с.
2. В. А. Артамонов, В. Н. Салий, Л. А. Скорняков. Общая алгебра. Том 2. М.: Наука, Физматлит, 1991. – 480 с.
3. А. В. Ахо, Д. Э. Хопкрофт, Д. Д. Ульман. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. – 521 с.
4. И. Н. Пономаренко. Проблема изоморфизма графов: Алгоритмические аспекты: [Электронный ресурс]. Санкт-Петербург, 2010, 57с. URL: [http://logic.pdmi.ras.ru/csclub/sites/default/files/ graph\_isomorphism\_ponomarenko\_lecture\_notes.pdf](http://logic.pdmi.ras.ru/csclub/sites/default/files/%20graph_isomorphism_ponomarenko_lecture_notes.pdf) (дата обращения: 18.09.2015)
5. Гаврилов Г. П., Сапоженко А. А. Задачи и упражнения по дискретной математике: Учеб. пособие. М.: ФИЗМАТЛИТ, 2006. – 416 с.
6. Alexander Smal. Explanation for “Tree isomorphism” talk: [Electronic resource]. Saint-Petersburg, 2008, 10 p. URL: http://www14. informatik.tumuenchen.de/konferenzen/Jass08/courses/1/smal/Smal\_ Paper.pdf (дата обращения: 10.09.2015)
7. Хопкрофт Дж. Е., Тарьян Р. Е. Изоморфизм планарных графов. Кибернетический сборник. Вып 12. М: Мир, 1975. – 208 с.

**References**

1. L. V. Zyablitseva, S.U. Korabelshchikova, I. N. Popov. Some special semigroups and their homomorphisms*.* – Arkhangelsk: NARFU, 2013. – 128 p.
2. V. A. Artamonov, V. N. Saliem, L. A. Skornyakov. General algebra. Volume 2. M.: Nauka, Fizmatlit, 1991. – 480 p.
3. A. V. Aho, J. E. Hopcroft, J. D. Ullman. The construction and analysis of computational algorithms. M.: Mir, 1979. – 521 p.
4. I. N. Ponomarenko. The problem of graphs isomorphism: Algorithmic aspects: [Electronic resource]. St. Petersburg, 2010, 57 p. URL: http://logic.pdmi.ras.ru/csclub/sites/default/files/graph\_isomorphism\_ponomarenko\_lecture\_notes.pdf
5. G. P. Gavrilov, A. A. Sapozhenko Tasks and exercises in discrete mathematics: Proc. allowance. M.: FIZMATLIT, 2006. – 416 p.
6. Alexander Smal. Explanation for “Tree isomorphism” talk: [Electronic resource]. Saint-Petersburg, 2008,10 p. URL: [http://www14.informatik.tu-muenchen.de/ konferenzen /Jass08/courses/1/smal/Smal\_Paper.pdf](http://www14.informatik.tu-muenchen.de/konferenzen/Jass08/courses/1/smal/Smal_Paper.pdf)

**Application of algorithms for checking the isomorphism of graphs in the theory of semigroups**

***Zyablitseva Larisa Vladimirovna***

*Associate Professor of the Department of Mathematical Analysis, Algebra and Geometry, Institute of Mathematics, Information and Space Technologies, Northern (Arctic) Federal University named after M. V. Lomonosov, Arkhangelsk, Russian Federation;*

***Pestov Sergei Alekseevich***

*Student of the Northern (Arctic) Federal University named after M. V. Lomonosov, Arkhangelsk, Russian Federation;*

**Аbstract.** One of the most interesting problems in the theory of semigroups is the problem of isomorphism for this class of semigroups. This problem consists in the existence of an algorithm that recognises that for any two semigroups of a given class, they are isomorphic or not. A similar problem exists in graph theory, and there's this question for some classes of graphs solved.   
 The article considers semigroups, for checking isomorphism which we can apply known algorithms for checking the isomorphism of graphs. This semigroups that are semilattices. Described as for such semigroups can be found corresponding graph. This graph can be a tree, and in this case for checking isomorphism of such semigroups it is possible to apply known algorithms for checking the isomorphism of trees. The article is formulated and proved a criterion, in which case the graph of the semilattice is a tree.   
Further justify the choice of algorithm for testing isomorphism of trees, describes the algorithm, presents a program written in Haskell that implements it.   
In order to apply the algorithm for checking isomorphism of semilattices, it is first necessary to compare the semilattice tree. For this purpose the authors have developed and implemented also in Haskell. an efficient algorithm.   
The program for the two semilattices given by Cayley tables, displays the structure of the corresponding trees, the canonical name of the obtained trees, checks the isomorphism of trees, and thus the semilattices. It should be noted that the choice and implementation of algorithms are effective, in a few seconds determines the isomorphism of semilattices with a three-digit number of items.   
  
Key words and phrases: semigroup; graphs; isomorphism of graphs and semigroups; semilattice; trees.