ИЗВЕСТИЯ ВЫСШИХ УЧЕБНЫХ ЗАВЕДЕНИЙ

2004 МАТЕМАТИКА № 9 (508)

УДК 519.175

О.В. РАСИН

ПОЛИНОМИАЛЬНЫЙ АЛГОРИТМ РАСПОЗНАВАНИЯ ИЗОМОРФИЗМА ПОЧТИ ДЕРЕВЬЕВ

Всюду в этой статье под термином "граф" будем понимать неориентированный граф без петель и кратных ребер. Проблема изоморфизма для заданного класса графов, состоит в построении полиномиального алгоритма, который для любых двух графов из этого класса определяет являются ли они изоморфными. В настоящее время неизвестно, существует ли полиномиальный алгоритм решения проблемы изоморфизма в классе всех графов. Однако для отдельных классов такие алгоритмы найдены. Например, алгоритм Ахо–Хопкрофта–Ульмана для деревьев ([1], гл. 3) и алгоритм Бабаи–Лакса для связных графов с ограниченными степенями вершин. Последнему алгоритму посвящена большая часть монографии [2].

В данной работе построен алгоритм для проверки изоморфизма связных графов, блоки которых являются графами с ограниченными степенями вершин. При построении алгоритма использовались идеи алгоритма Ахо–Хопкрофта–Ульмана, а также методы и приемы Бабаи и Лакса.

1. Предварительные сведения

Определения используемых ниже теоретико-групповых понятий можно найти в [3]. Как обычно через S_n будем обозначать симметрическую группу степени n. Для каждого натурального числа b определим класс групп Γ_b, состоящий из конечных групп, имеющих композиционные ряды, порядки факторов которых не превосходят b. Очевидно, $\Gamma_b \subset \Gamma_{b+1}$. Доказательства следующих двух лемм можно найти в [2].

Лемма 1.1. *Если группа принадлежит классу Γ_b, то любая ее подгруппа содержится в Γ_b.*

Лемма 1.2. (a) *S_2 является группой класса Γ_2.*
(b) *S_3 и S_4 являются группами класса Γ_3.*
(c) *Если $n > 4$, S_n принадлежит Γ_b при $b = \frac{n!}{2}$.*

Если G — группа подстановок, определенная на множестве X, и $Y \subseteq X$, то через G_Y будем обозначать *стабилизатор множества Y в группе G.*

В дальнейшем будем рассматривать алгоритмы, использующие группы подстановок. Будем говорить, что известна группа подстановок $G \leq S_n$, если известно порождающее множество группы G, мощность которого не превосходит $p(n)$, где p — некоторый полином. Такие множества называются *порождающими множествами полиномиальной мощности* [2]. В данной статье описываются алгоритмы только таких множеств.

Предложение 1.1 ([2], с. 220). *Пусть подгруппа G группы S_n принадлежит классу Γ_b, где b – натуральная константа. Существует алгоритм, строящий за полиномиальное время порождающее множество для группы G_Y.*

Будем говорить, что автоморфизм ψ графа X фиксирует ребро $e = \{u, v\}$ из X, если $\{u, v\} = \{\psi(u), \psi(v)\}$. Очевидно, множество автоморфизмов, фиксирующих ребро e, является группой. Обозначать эту группу будем через $\mathrm{Aut}_e(X)$.

Предложение 1.2 ([2], с. 184). *Пусть X — связный граф, степени вершин которого не превосходят натуральной константы d, а e – ребро графа X. Существует полиномиальный алгоритм, строящий порождающее множество группы автоморфизмов $\mathrm{Aut}_e(X)$ графа X, фиксирующих ребро e.*

Предложение 1.3 ([2], с. 187). *Пусть $X = (V, E)$ — связный граф, степени вершин которого не превышают $d > 3$, e — ребро графа X. Тогда $\mathrm{Aut}_e(X) \in \Gamma_b$, где $b = 3$, если $d = 4$ или $d = 5$, и $b = \frac{(d-1)!}{2}$, если $d > 5$.*

При построении алгоритмов будем предполагать, что графы, которые здесь изучаются, задаются списками смежностей.

2. Изоморфизм цветных графов с ограниченными степенями вершин

Определение 2.1. Пусть $G = (V, E)$ — граф, l — натуральное число. Произвольная функция вида $f : V \to \{1, \ldots, l\}$ называется *цветной функцией* графа G, а $f(v)$, $v \in V$, — *цветом* вершины v. Граф, на множестве вершин которого определена цветная функция, называется *цветным графом*. Для цветных графов будем использовать следующие обозначения: $G = (V, E, f)$, G_f или (G, f).

Определение 2.2. Пусть $G_1 = (V_1, E_1, f_1)$, $G_2 = (V_2, E_2, f_2)$ — цветные графы. Биективное отображение $\phi : V_1 \to V_2$ будем называть 0-*изоморфизмом цветных графов* G_1 и G_2, если для любой пары вершин u и v графа G_1 их образы $\phi(u)$ и $\phi(v)$ смежны в графе G_2 тогда и только тогда, когда u и v смежны в G_1.

Легко видеть, что любой 0-изоморфизм цветных графов является изоморфизмом соответствующих графов.

Определение 2.3. Пусть $G_1 = (V_1, E_1, f_1)$, $G_2 = (V_2, E_2, f_2)$ — цветные графы. Биективное отображение $\phi : V_1 \to V_2$ будем называть *изоморфизмом цветных графов* G_1 и G_2, если выполнены условия

1) ϕ является 0-изоморфизмом;

2) $f_1(v) = f_2(\phi(v))$ для любой вершины v графа G_1.

Определение 2.4. Пусть $G_1 = (V_1, E_1, f_1)$, $G_2 = (V_2, E_2, f_2)$ — цветные графы, а i — некоторый цвет. Биективное отображение $\phi : V_1 \to V_2$ будем называть i-*изоморфизмом цветных графов* G_1 и G_2, если выполнены условия

1) ϕ является 0-изоморфизмом;

2) $f_1(v) = i$ тогда и только тогда, когда $f_2(\phi(v)) = i$ для любой вершины v графа G_1.

Если отображение является i-изоморфизмом, то будем говорить, что оно сохраняет цвет i.

Степенью вершины в цветном графе будем называть число вершин, смежных с ней.

В этом параграфе приведем обобщение алгоритма Бабаи–Лакса для цветных графов.

Задача 2.1. Даны два связных цветных графа $G_1 = (V_1, E_1, f_1)$ и $G_2 = (V_2, E_2, f_2)$, степени вершин которых ограничены некоторой константой d, где $f_1 : V_1 \to C$, $f_2 : V_2 \to C$, и $C = \{1, \ldots, l\}$. Изоморфны ли цветные графы G_1 и G_2?

В этом параграфе построим полиномиальный алгоритм, решающий задачу 2.1. Эта задача полиномиально сводима к задаче поиска группы автоморфизмов связного цветного графа с ограниченными степенями вершин. Зафиксируем ребро $e_1 = \{u_1, v_1\}$ графа G_1. Возьмем некоторое ребро $e_2 = \{u_2, v_2\}$ графа G_2 такое, что либо $f_1(u_1) = f_2(u_2)$ и $f_1(v_1) = f_2(v_2)$, либо $f_1(u_1) = f_2(v_2)$ и $f_1(v_1) = f_2(u_2)$. Добавим в граф G_1 (граф G_2) вершину w_1 (вершину w_2), удалим ребро e_1 (ребро e_2) и добавим ребра $\{u_1, w_1\}$ и $\{v_1, w_1\}$ (ребра $\{u_2, w_2\}$ и $\{v_2, w_2\}$). Построенный граф обозначим

G_1' (обозначим G_2'). Объединяя граф G_1' с G_2' и прибавляя ребро $\{w_1, w_2\}$, получим связный граф G. Определим на графе G цветную функцию f следующим образом:

$$f(v) = \begin{cases} f_1(v), & v \in V_1; \\ f_2(v), & v \in V_2; \\ l+1, & v \in \{w_1, w_2\}. \end{cases}$$

Легко видеть, что граф (G, f) является цветным графом, степень каждой вершины которого ограничена числом d, при $d \geq 3$. При $d < 3$ степень каждой вершины в G не превосходит трех.

Если существует автоморфизм ψ цветного графа G такой, что $\psi(w_1) = w_2$, то ограничение отображения ψ на множество вершин графа G_1 является изоморфизмом цветных графов G_1 и G_2.

Таким образом, мы должны решить следующую задачу.

Задача 2.2. Дан связный цветной граф $G = (V, E, f)$, степень каждой вершины которого не превосходит d. Пусть e — некоторое ребро этого графа, а $C = \{1, \ldots, l+1\}$ – набор цветов вершин графа, т. е. $f : V \to C$. Необходимо определить группу автоморфизмов цветного графа G, сохраняющих ребро e.

Если найдем полиномиальный алгоритм, решающий задачу 2.2, то, пройдя в худшем случае по всем ребрам графа G_2, установим изоморфны или нет цветные графы G_1 и G_2 за полиномиальное время.

Пусть $\mathrm{Aut}_e^0(G)$ обозначает группу 0-автоморфизмов графа G, которые фиксируют ребро e, а $\mathrm{Aut}_e^{C_i}(G)$ — группу автоморфизмов графа G, которые сохраняют все цвета из множества $C_i = \{1, \ldots, i\}$ и фиксируют ребро e. Через W_j обозначим множество вершин цвета j в графе G.

Алгоритм 2.1. *Поиск группы автоморфизмов цветного графа с ограниченными степенями вершин.*

Вход: связный цветной граф $G = (V, E, f)$, где $f : V \to \{1, \ldots, l\}$, степени вершин которого не превосходят d, и ребро $e = \{v_1, v_2\}$.

Выход: порождающее множество группы автоморфизмов $\mathrm{Aut}_e^C(G)$ цветного графа G, фиксирующих ребро e.

1. Находим порождающее множество L_0 группы $\mathrm{Aut}_e^0(G)$, и положим $\mathrm{Aut}_e^{C_0}(G) = \mathrm{Aut}_e^0(G)$. Перейти к п. 2.
2. $i := 1$. Перейти к п. 3.
3. Находим порождающее множество L_i стабилизатора множества W_i в группе $\mathrm{Aut}_e^{C_{i-1}}(G)$. Очевидно, $\mathrm{Aut}_e^{C_{i-1}}(G)_{W_i}$ сохраняет все цвета из множества C_i, поэтому $\mathrm{Aut}_e^{C_i}(G) = \mathrm{Aut}_e^{C_{i-1}}(G)_{W_i}$. Перейти к п. 4.
4. Если $i = l$, то алгоритм заканчивает работу и возвращает L_i, иначе $i := i + 1$ и перейти к п. 2.

Теорема 2.1. *Алгоритм* 2.1 *находит порождающее множество группы* $\mathrm{Aut}_e^C(G)$ *за полиномиальное время.*

Полиномиальность алгоритма следует из предложений 1.1, 1.2, 1.3 и леммы 1.1. □

Алгоритм 2.2. *Распознавание изоморфизма цветных графов с ограниченными степенями вершин.*

Вход: связные цветные графы $G_1 = (V_1, E_1, f_1)$ и $G_2 = (V_2, E_2, f_2)$, где $f_1 : V_1 \to \{1, \ldots, l\}$ и $f_2 : V_2 \to \{1, \ldots, l\}$, степени вершин которых не превосходят d.

Выход: *true*, если цветные графы изоморфны, *false*, если они не изоморфны.

1. Возьмем в графе G_1 ребро $e_1 = \{u_1, v_1\} \in E_1$. Перейти к п. 2.
2. $E' := E_2$. Перейти к п. 3.
3. Если $E' \neq \emptyset$, то берем $e_2 = \{u_2, v_2\} \in E'$, $E' := E' \setminus e_2$ и переходим к п. 4, в противном случае перейти к п. 9.

4. Если $f_1(u_1) = f_2(u_2)$ и $f_1(v_1) = f_2(v_2)$, либо $f_1(v_1) = f_2(u_2)$ и $f_1(u_1) = f_2(v_2)$, то перейти к п. 5. Иначе перейти к п. 3.
5. Строим вспомогательный цветной граф H. Сначала $H := (G_1 \cup G_2)$. Затем добавляем к H вершины w_1 и w_2. После этого добавляем к графу H два ребра, соединяющие w_1 с концами ребра e_1, и два ребра, соединяющие w_2 с концами ребра e_2, добавляем ребро $\{w_1, w_2\}$ и удаляем из H ребра e_1 и e_2. Цвета вершин графа H, отличных от w_1 и w_2, остаются теми же, какими они были в графах G_1 и G_2. Вершины w_1 и w_2 окрашиваются в цвет $l + 1$. Перейти к п. 6.
6. $e := \{w_1, w_2\}$. Перейти к п. 7.
7. С помощью алгоритма 2.1 находим порождающие множество L группы автоморфизмов $\mathrm{Aut}_e(H)$ связного цветного графа H, фиксирующих ребро e. Перейти к п. 8.
8. Проверяем все элементы множества L, если среди них есть хотя бы один, переставляющий w_1 и w_2, то алгоритм заканчивает работу и возвращает *true*. В противном случае переходим к п. 3.
9. Алгоритм заканчивает работу и возвращает *false*.

Теорема 2.2. *Алгоритм* 2.2 *за полиномиальное время определяет будут ли изоморфны два заданных цветных графа, степени вершин которых ограничены константой* d.

Полиномиальность алгоритма вытекает из полиномиальности алгоритма 2.1.

Таким образом, проверка изоморфизма связных цветных графов, степени вершин которых ограничены, может быть произведена за полиномиальное время.

Приведем алгоритм работы процедуры *IsoPartition*. Пусть задан набор связных цветных графов $P = \{G_1, \ldots, G_s\}$, степени вершин которых ограничены. *IsoPartition* разбивает множество графов на изоморфные классы P_i, $i = 1, \ldots, t$, т. е. графы G_{j_1} и G_{j_2} будут принадлежать одному и тому же классу P_i тогда и только тогда, когда они изоморфны.

Алгоритм 2.3. *IsoPartition.*

Вход: дан набор связных цветных графов $P = \{G_1, \ldots, G_s\}$, степени вершин которых ограничены константой d.

Выход: разбиение множества графов P на изоморфные классы $\{P_1, \ldots, P_t\}$.

1. $P_1 := \{G_1\}$; $t := 1$; $i := 1$. Перейти к п. 2.
2. Если $i = s$, то алгоритм заканчивает работу, иначе перейти к п. 3.
3. $i := i + 1$, $j := 0$. Перейти к п. 4.
4. $j := j + 1$. Если $j \leq t$, то перейти к п. 5. Иначе перейти к п. 6.
5. Пусть H — некоторый граф из множества P_j. С помощью алгоритма 2.2 проверяем изоморфны или нет графы G_i и H. Если они изоморфны, то $P_j := P_j \cup \{G_i\}$ и переходим к п. 2, в противном случае перейти к п. 4.
6. $t := t + 1$; $P_t := \{G_j\}$.

Очевидна следующая

Теорема 2.3. *Процедура IsoPartition работает за полиномиальное время.*

3. Изоморфизмы графов, блоки которых являются графами с ограниченными степенями вершин

Пусть $\mathcal{G}_d$ — класс связных графов, степени вершин которых не превосходят фиксированного натурального числа d. Как было отмечено выше, существует алгоритм, решающий проблему изоморфизма в классе $\mathcal{G}_d$. Обозначим через $B\mathcal{G}_d$ класс связных графов, блоки которых принадлежат классу $\mathcal{G}_d$. Иными словами, $B\mathcal{G}_d$ состоит из связных графов, блоки которых являются графами с ограниченными степенями вершин. В этом параграфе представлен алгоритм, решающий проблему изоморфизма для класса $B\mathcal{G}_d$.

Определение 3.1. Пусть G — связный граф с множеством блоков $\{B_1, \ldots, B_s\}$ и множеством точек сочленения $\{c_1, \ldots, c_l\}$. Граф $bc(G)$, вершинами которого являются элементы множества $\{B_1, \ldots, B_s, c_1, \ldots, c_l\}$ и две вершины которого смежны, если одна соответствует блоку B_i, а другая точке сочленения c_j, причем $c_j \in B_i$, называется *графом блоков и точек сочленения* графа G.

Как хорошо известно, граф блоков и точек сочленения любого связного графа является деревом [4], причем вершинами степени 1 могут быть только вершины, которые соответствуют блокам графа G. Граф блоков и точек сочленения будем в дальнейшем называть *деревом блоков и точек сочленения.*

Пусть G — связный граф. Через C обозначим центр дерева $bc(G)$ (определение центра можно найти в [5]). Как известно, центр дерева состоит либо из одной вершины, либо из двух смежных вершин. Посмотрим, что представляет из себя центр графа $bc(G)$. Возможны три случая.

1. Центр состоит из одной вершины B_0, которая соответствует некоторому блоку графа G.

2. Центр состоит из одной вершины c_0, которая соответствует некоторой точке сочленения графа G.

3. Центр состоит из двух вершин B_0 и c_0, первая из которых соответствует некоторому блоку графа G, а вторая – точке сочленения.

Если для $bc(G)$ выполняются случаи 1 или 2, то будем рассматривать его как корневое дерево, корнем которого является вершина центра. Если же для $bc(G)$ выполняется случай 3, то будем рассматривать его как корневое дерево, корнем которого является вершина центра, соответствующая точке сочленения.

Напомним некоторые термины, используемые при рассмотрении корневых деревьев. Вершины степени 1 в $bc(G)$, не принадлежащие центру, называются *листьями. Высота дерева* — это длина самого длинного пути из корня в какой-нибудь лист. *Глубина вершины* — это длина пути из вершины в корень.

Пусть $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ — графы из $B\mathcal{G}_d$. В приводимом здесь алгоритме, важную роль будут играть деревья блоков и точек сочленения $bc(G_1)$ и $bc(G_2)$. Очевидно, необходимым условием изоморфизма любой пары графов, является изоморфизм их деревьев блоков и точек сочленения. Кроме того ясно, что при изоморфизме деревьев центр одного дерева переходит в центр второго. Поэтому имеет смысл рассматривать только следующие три случая.

A. Центр $bc(G_1)$ состоит из одной вершины, которая соответствует блоку графа G_1. Центр $bc(G_2)$ состоит из одной вершины, которая соответствует блоку графа G_2.

B. Центр $bc(G_1)$ состоит из одной вершины, которая соответствует некоторой точке сочленения графа G_1. Центр $bc(G_2)$ состоит из одной вершины, которая соответствует некоторой точке сочленения графа G_2.

C. Центры $bc(G_1)$ и $bc(G_2)$ состоят из двух вершин.

Если для центров деревьев $bc(G_1)$ и $bc(G_2)$ ни один из этих случаев не выполняется, то графы G_1 и G_2 не изоморфны.

Ниже под *уровнем вершины* будем понимать разность между высотой дерева и глубиной этой вершины. Необходимо отметить, что вершины дерева блоков и точек сочленения, находящиеся на одном уровне, либо все соответствуют точкам сочленения, либо все соответствуют блокам.

Перейдем к изложению алгоритма. В ходе своей работы алгоритм приписывает метки (натуральные числа) вершинам деревьев $bc(G_1)$ и $bc(G_2)$ таким образом, что корням этих деревьев будут приписаны одни и те же метки тогда и только тогда, когда графы G_1 и G_2 изоморфны. Сначала приписываются метки вершинам, находящимся на самом "нижнем" (нулевом) уровне, затем используя эти метки приписываем метки вершинам, находящимся на уровень выше и т.д., пока не дойдем до корней. Необходимо иметь в виду, что сопоставление меток для вершин, которые соответствуют точкам сочленения, и для вершин, которые соответствуют блокам, осуществляется по-разному.

Предположим, что вершинам приписаны метки, которые находятся на расстоянии $k+1$ от корня. Для вершин, находящихся на расстоянии k от корня, производим следующие действия.

В случае, когда эти вершины соответствуют точкам сочленения, каждой из них ставим в соответствие упорядоченный по возрастанию кортеж меток сыновей. Затем, используя алгоритм лексикографической сортировки ([1], гл. 3), упорядочиваем все эти кортежи. Получаем упорядоченный набор кортежей. Вершинам, которые представлены первым кортежем, ставим в качестве метки число 1. Вершинам, которые представлены вторым отличающимся кортежем, — число 2 и т. д. Таким образом, одинаковые метки получат только те вершины, у которых одинаковые кортежи.

Теперь рассмотрим случай, когда вершины, находящиеся на расстоянии k, соответствуют блокам. Пусть p — вершина дерева блоков и точек сочленения, находящаяся на расстоянии k от корня, B — блок графа, которому она соответствует. Допустим, что при помечивании вершин, находящихся на расстоянии $k+1$ от корня, использовалось t меток.

1) Пусть p не является листом. Обозначим точки сочленения блока B, которые соответствуют сыновьям вершины p, через $v_1, v_2, \ldots, v_r$, а метки этих сыновей через $l_1, l_2, \ldots, l_r$. Предположим, что v_0 — точка сочленения блока, которая соответствует отцу вершины p в дереве блоков и точек сочленения. Определим на вершинах блока B цветную функцию f следующим образом:

$$f(v) = \begin{cases} l_j, & \text{если } v = v_j; \\ t+1, & \text{если } v = v_0; \\ t+2, & \text{если } v \neq v_j,\ j \in \{0, 1, \ldots, r\}. \end{cases}$$

Полученный цветной граф будем обозначать (B, f).

2) Пусть p является листом. Обозначим точку сочленения блока, которая соответствует отцу вершины p в дереве блоков и точек сочленения через v_{i_0}. Определим на вершинах блока B цветную функцию f следующим образом:

$$f(v) = \begin{cases} t+1, & \text{если } v = v_0; \\ t+2, & \text{если } v = v_j. \end{cases}$$

Полученный цветной граф будем обозначать (B, f).

Таким образом, каждой вершине, находящейся на расстоянии k от корня, сопоставляется цветной граф.

Получаем набор цветных графов $\{(B_1, f_1), \ldots, (B_s, f_s)\}$ с ограниченными степенями вершин. С помощью алгоритма *IsoPartition* разбиваем их на классы $P_1, \ldots, P_s$, где два цветных графа принадлежат одному и тому же классу тогда и только тогда, когда они изоморфны. Если блок $B \in P_i$, то вершину p, которой он соответствует, помечаем меткой i.

Представим эти рассуждения в виде алгоритма.

Алгоритм 3.1. *Распознавание изоморфизма связных графов, блоки которых являются графами с ограниченными степенями вершин.*

Вход: G_1 и G_2 – связные графы, блоки которых являются графами с ограниченными степенями вершин.

Выход: возвращает *true*, если G_1 и G_2 изоморфны, возвращает *false*, если G_1 и G_2 не изоморфны.

1. С помощью поиска в глубину находим блоки и точки сочленения графов G_1 и G_2 (см. [4]) и строим для каждого из этих графов деревья блоков и точек сочленения $bc(G_1)$ и $bc(G_2)$.
2. Находим центры деревьев $bc(G_1)$ и $bc(G_2)$ (см. [5]). Если имеет место один из случаев A, B или C, то переходим к п. 3. В противном случае графы не изоморфны, и алгоритм заканчивает работу, возвращая *false*.
3. Находим h_1 — высоту дерева $bc(G_1)$ и h_2 — высоту дерева $bc(G_2)$.
4. Если $h_1 \neq h_2$, то $bc(G_1)$ и $bc(G_2)$ не изоморфны, поэтому и графы G_1 и G_2 не изоморфны и алгоритм завершает свою работу, возвращая *false*. В противном случае, если $h_1 = h_2$, то $h := h_1$, $k := h$; (k будет пробегать все значения от 0 до h) и переходим к п. 5.

5. Если $h = 0$, то G_1 и G_2 являются графами степени, вершины которых ограничены. Поэтому можем применить к ним алгоритм Бабаи–Лакса. Если графы изоморфны, то алгоритм возвращает *true*. В противном случае алгоритм возвращает *false*. Алгоритм заканчивает работу.
6. Пусть $Z_k^1 = \{p_1^1, \ldots, p_s^1\}$ — множество вершин $bc(G_1)$, находящихся на расстоянии k от корня, а $Z_k^2 = \{p_1^2, \ldots, p_{s'}^2\}$ — множество вершин $bc(G_2)$, находящихся на расстоянии k от корня. Если $s \neq s'$, то графы не изоморфны и алгоритм заканчивает работу, возвращая *false*. Пусть $s = s'$. Если вершины уровня k деревьев $bc(G_1)$ и $bc(G_2)$ соответствуют блокам, то переходим к п. 7. В противном случае переходим к п. 8.
7. Пусть $B_1^1, \ldots, B_s^1$ — блоки графа G_1, которые соответствуют вершинам из Z_k^1, а $B_1^2, \ldots, B_s^2$ — блоки графа G_2, которые соответствуют вершинам из Z_k^2. Каждому из блоков сопоставляем цветной граф так, как это описывалось в рассуждениях перед алгоритмом. Получаем набор цветных графов $\{(B_i^1, f_i^1), (B_j^2, f_j^2) : i, j = 1, \ldots, s\}$. С помощью алгоритма 2.3 разбиваем их на изоморфные классы $P_1, \ldots, P_r$. Если блок B_i^j попадает в класс $P_{i'}$, вершине p_i^j дерева $bc(G_j)$, которая ему соответствует, приписывем метку i'. Перейти к п. 9.
8. Каждой вершине p_i^j, $i \in \{1, \ldots, s\}$, $j \in \{1, 2\}$, ставим в соответствие упорядоченный по возрастанию кортеж меток ее сыновей. С помощью лексикографической сортировки упорядочиваем эти кортежи. Пусть $C = (C_1, \ldots, C_r)$ — упорядоченная последовательность этих кортежей. Вершинам, которые представлены первым кортежем в C, сопоставляем 1. Вершинам, которые представлены вторым отличающимся кортежем, сопоставляем 2 и т. д. Таким образом, каждая из вершин p_i^j, $i = 1, \ldots, r$, получает метку. Перейти к п. 9.
9. Если $k > 0$, то $k := k - 1$ и перейти к п. 6. Если $k = 0$, то перейти к п. 10.
10. Если метки, приписанные корням $bc(G_1)$ и $bc(G_2)$, совпадают, то графы G_1 и G_2 изоморфны и алгоритм заканчивает работу, возвращая *true*. В противном случае графы не изоморфны, и алгоритм заканчивает работу, возвращая *false*. □

Теорема 3.1. *Графы G_1 и G_2 из класса $B\mathcal{G}_d$ изоморфны тогда и только тогда, когда алгоритм* 3.1 *возвращает true.*

Доказательство. Пусть $h_1(h_2)$ — высота дерева $bc(G_1)(bc(G_2))$. Если $h_1 \neq h_2$, то графы не изоморфны и алгоритм возвращает *false*.

Пусть $h_1 = h_2 = h$. Доказательство проведем индукцией по h. Если $h = 0$, то, очевидно, утверждение теоремы справедливо. Пусть теорема доказана для всех $h < k + 1$, где k — некоторое натуральное число. Пусть r_1 и r_2 — корни деревьев $bc(G_1)$ и $bc(G_2)$ соответственно, $s_1^1, \ldots, s_m^1$ — сыновья r_1, а $s_1^2, \ldots, s_{m'}^2$ — сыновья r_2. Если $m \neq m'$, то графы не изоморфны и алгоритм возвращает *false* (см. п. 8). Пусть $m = m'$.

Введем обозначение. Пусть p_i — вершина дерева $bc(G_i)$, $i \in 1, 2$, и T_{p_i} — поддерево $bc(G_i)$ с корнем в p_i. Через V_{p_i} обозначим подмножество множества вершин G_i, состоящее из точек сочленения и вершин всех блоков графа G_i, которые соответствуют вершинам из T_{p_i}. Очевидно, T_{p_i} являются деревом блоков и точек сочленения подграфа $G(V_{p_i})$ графа G, порожденного множеством вершин V_{p_i}.

По предположению индукции вершины s_i^j и $s_{i'}^{j'}$, где $i, i' \in \{1, \ldots, m\}$, $j, j' \in \{1, 2\}$, получат одинаковые метки тогда и только тогда, когда подграфы $G(V_{s_i^j})$ и $G(V_{s_{i'}^{j'}})$ изоморфны. Отсюда вытекает утверждение теоремы. □

Теорема 3.2. *Алгоритм* 3.1 *работает за полиномиальное время.*

Утверждение теоремы сразу следует из того, что алгоритм 2.3 работает за полиномиальное время.

4. Изоморфизмы почти деревьев с параметром k

Определение 4.1. Связный граф G называется *почти деревом с параметром* k, если он обладает таким остовным лесом T, что каждый блок графа G содержит не более чем k не принадлежащих лесу T ребер.

В этом параграфе будет показано, что для каждого натурального k алгоритм 3.1 решает проблему изоморфизма почти деревьев с параметром k.

Для доказательства этого факта нам понадобятся следующие утверждения. Напомним, что вершина дерева называется *висячей*, если ее степень равна 1.

Лемма 4.1. *Пусть H — поддерево некоторого дерева T. Тогда в дереве H висячих вершин содержится не больше, чем в T.*

Доказательство этой леммы не представляет труда. □

Лемма 4.2. *Пусть G — двусвязное почти дерево с параметром k, а T — некоторое остовное дерево графа G. Тогда в дереве T число висячих вершин не превышает $2k$.*

Доказательство. Пусть $\{e_1, \ldots, e_p\}$ — множество ребер графа G, не входящих в T. По определению почти дерева имеем $p \leq k$. Обозначим через C_{e_i}, $i = 1, \ldots, p$, цикл, порожденный в графе G ребром e_i и простой цепью, соединяющей концы данного ребра в остовном дереве T. Поскольку G двусвязный граф, любая висячая вершина остова T принадлежит некоторому циклу C_{e_i}, $i = 1, \ldots, p$. По построению каждый цикл C_{e_i} может содержать не более двух висячих вершин остова T. Следовательно, число висячих вершин дерева T не превосходит $2p \leq 2k$. □

Лемма 4.3. *Пусть G — двусвязное почти дерево с параметром k. Тогда степень каждой вершины графа G не превосходит $2k$.*

Доказательство. Пусть v — вершина графа G. Обозначим через T дерево поиска в ширину в графе G из вершины v. По лемме 4.2 число висячих вершин не превосходит $2k$. Пусть H — поддерево из T, состоящее из вершины v и всех вершин множества $N_T(v)$ — окрестности вершины v в дереве T. Очевидно, $N_T(v) = N_G(v)$, где $N_G(v)$ — окрестность v в графе G. По лемме 4.1 имеем $|N_T(v)| \leq 2k$. Откуда следует $|N_G(v)| \leq 2k$. □

Теорема 4.1. *Для каждого k существует полиномиальный алгоритм, решающий проблему изоморфизма для почти деревьев с параметром k.*

Это непосредственно следует из леммы 4.3 и теорем 3.2 и 3.1.

В заключение автор выражает благодарность своему научному руководителю В.А. Баранскому за внимание к работе и замечания, способствовавшие ее улучшению.

Литература

1. Ахо Х., Хопкрофт Дж., Ульман Дж. *Построение и анализ вычислительных алгоритмов.* – М.: Мир, 1979. – 536 с.
2. Hoffmann C.M. *Group-theoretics algorithms and graph isomorphism* // Lect. Notes Comput. Science. – 1982. – V. 136. – № 8. – 311 p.
3. Каргаполов М.И., Мерзляков Ю.И. *Основы теории групп.* – М.: Наука, 1972. – 240 с.
4. Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. *Лекции по теории графов.* – М.: Наука, 1990. – 384 с.
5. Евстигнеев В.А., Касьянов В.Н. *Теория графов: алгоритмы обработки деревьев.* – Новосибирск: ВО Наука. Сибирская издательская фирма, 1994. – 360 с.
6. Асанов М.О., Баранский В.А., Расин В.В. *Дискретная математика: графы, матроиды, алгоритмы.* – Ижевск: НИЦ "Регулярная и хаотическая динамика", 2001. – 288 с.

Уральский государственный университет

Поступила 11.11.2002