Новый алгоритм для установления изоморфизма между графами и его компьютерная реализация

Выполнил: Калистратов Даниил, 10 класс,
МАОУ «Лицей № 38;
Руководители: Коган Г.П., кандидат физико-математических наук;
Малышев Д.С., доктор физико-математических наук, профессор, ведущий научный сотрудник НИУ ВШЭ в Нижнем Новгороде

г. Нижний Новгород

2023

Оглавление

Введение

Актуальность. На данный момент в наибольшей части алгоритмов, направленных на установление изоморфизма графов, используется задача об изоморфизме матриц. Установив изоморфность матриц смежности графов, можно утверждать об изоморфности этих графов.

Для определения изоморфизма графов нередко применяются эвристические алгоритмы, которые для некоторых классов графов не являются полиномиальными. В основе эвристических методов лежит разделение всех вершин графов на классы таким образом, что алгоритм рассматривает отображение вершин графа в другие вершины того же класса. Для этого применяются свойства вершин графа, инвариантные относительно изоморфизма. В качестве инвариантов неизменных для всего класса изоморфных графов рассматриваются числовые, векторные, матричные инварианты.

В настоящее время определение изоморфизма двух графов с очень большим числом вершин по-прежнему считается неразрешимой в вычислительном смысле проблемой, так как даже создание венгерским математиком Ласло Бабаи квазиполинимиального алгоритма для проблемы изоморфизма графов все еще не позволяет численно решать эту задачу при помощи современных компьютеров.

Целью исследования является разработка принципиально новых полиномиальных методов определения изоморфности графов, которые позволят численно решать любые востребованные (включая научные исследования) примеры данной задачи и будут работать быстрее, чем какие-либо другие известные алгоритмы для этой задачи. К известным сложным (тестовым) примерам можно отнести в этом контексте случайную пару сильнорегулярных графов с равными системами параметров.

Задачи:

1. Провести анализ научной литературы и научно-популярных источников, посвященных разработкам методов определения изоморфизма графов.

2. Ввести понятия элементного спектра и элементного мультиспектра матрицы как множества и мультимножества значений ее элементов соответственно, а также понятие замены элементного спектра на случайный.

3. Составить поэтапный план для написания программы.

4. Написать программу на языке Python.

5. Выполнить практическую проверку предложенного алгоритма для решения переборной задачи на репрезентативной выборке тестовых примеров и подтвердить либо опровергнуть

работоспособность алгоритма.

Объект исследования – теория графов, теория матриц.

Предмет исследования – изоморфизм графов, изоморфизм матриц.

Методология исследования. При проведении исследования использовались следующие методы: анализ и синтез литературных и научно-популярных источников; дедукционный, индукционный методы.

Новизна. Предложен принципиально новый полиномиальный алгоритм для задачи изоморфизма графов, который возможно применить в рамках компьютерных вычислений, так как сложность этого алгоритма заключается в вычислении заданного многочлена от квадратной матрицы смежности, степень которого не превосходит числа ее строк.

Практическая значимость. Задача об изоморфизме графов является весьма интересной переборной задачей, имеющей многообразные значимые применения во многих естественных науках – химии, физике, биологии. В практической деятельности необходимость определения изоморфности или неизоморфности графов возникает при решении задач в математической (компьютерной) химии, при проектировании электронных схем (различных представлений электронной схемы), оптимизации компьютерных программ.

Глава 1. Постановка задачи и обзор литературных источников

Изоморфизм графов 1 и 2 – это биекция между множествами вершин и рёбер данных графов, такая что две вершины в графе 1 смежны тогда и только тогда, когда они смежны в графе 2, и ребра в графе 1 имеют общую вершину тогда и только тогда, когда они имеют общую вершину в графе 2.

Если рассматривать задачу об изоморфизме графов с точки зрения теории вычислительной сложности, то она является полной (то есть такой, что к ней полиномиально сводятся все остальные задачи, находящиеся с ней в одном классе) в своем классе задач, который назван ее именем – GI (graph isomorphism). В этот класс, кроме самой задачи об изоморфизме, входят задачи об изоморфизме групп и другие подобные задачи (об изоморфизме матриц, об изоморфизме графа подграфу). В настоящее время ни один предложенный полиномиальный алгоритм ее разрешения не является признанным научным сообществом.

В 1981 году Брендан МакКей разработал достаточно быстродействующий программный пакет NAUTY, который в дальнейшем был усовершенствован и появился под названием «TRACES». В основе этих программ лежит построение канонического кода графа, не зависящего от начального порядка нумерации вершин. В указанной программе два графа изоморфны тогда и только тогда, когда совпадают их канонические коды [2].

Крупным прорывом в этой области стала опубликованная несколько лет назад известная работа венгерского математика Ласло Бабаи (László Babai), где была доказана квазиполиномиальная разрешимость исследуемой нами проблемы [1, 7]. Это исследование основано на теории групп и подхода, заключающегося в рассмотрении системы небольших подмножеств вершин исходного графа.

В ноябре 2015 года Ласло Бабаи (математик и специалист по информатике из Чикагского университета) заявил о доказательстве того, что проблема изоморфизма графов разрешима за квазиполиномиальное время. Он опубликовал предварительные версии этих результатов в трудах симпозиума по теории вычислений 2016 года и международного конгресса математиков 2018 года [1, 7]. В январе 2017 года математик на короткое время отказался от утверждений о квазиполиномиальности, то есть поставил под сомнение свои первоначальные утверждения и вместо этого указал субэкспоненциальную временную сложность. Однако через пять дней он восстановил утверждение о квазиполиномиальности задачи об изоморфизме графов. По состоянию на 2022 год полная журнальная версия статьи Ласло Бабаи еще не опубликована.

Алгоритм, предложенный Ласло Бабаи, основан на виртуальном окрашивании вершин графа и действует следующим образом. Сначала выбираются две предположительно соответствующие вершины. После чего они окрашиваются в одинаковые цвета. Затем все вершины, соединенные ребрами с окрашенными вершинами, закрашиваются в различные цвета, причем таким образом, что предположительно соответствующие вершины окрашиваются в одинаковые цвета. Таким образом, алгоритм либо окрашивает все вершины графа и тогда графы изоморфны, либо перебирает все варианты раскраски и, если ни один из них не подходит, то графы считаются не изоморфными. Бабаи изложил основные моменты своей работы в двух лекциях, а присутствующие на них эксперты в области теории графов пока не нашли ошибок в рассуждениях ученого. Между тем окончательной верификации в математическом сообществе его работа пока не получила.

Однако предложенный венгерским математиком алгоритм имеет вычислительную сложность, слишком большую для его применения в компьютерных вычислениях с целью решения некоторых практических задач.

Помимо названных исследователей, работы по поиску методов определения изоморфизма графов, в том числе установления изоморфизма подграфа заданному графу, проводили В.П. Карелин, И.Н. Пономаренко, С.В. Погожев, Г.М. Хитров, С.В. Курапов, М.В. Давидовский, В.К. Погребной, О.В. Герман, А.А. Дунаев и др. [3−5, 8–9, 12].

Так, в статье С.В. Погожева, Г.М. Хитрова «О проблеме изоморфизма графов и об одном матричном способе ее решения» [8] предложен алгоритм, решающий задачу изоморфизма графа как частный случай задачи изоморфизма графа остовному подграфу другого графа. Данный алгоритм строится на инвариантах матриц: столбцах строчных сумм, столбцах столбцовых сумм и диагоналей, записанных в виде столбцов. Этот алгоритм реализован в программе psimilar из пакета программ ZerOne.

Таким образом, до настоящего момента наибольшее число предложенных алгоритмов не подходит для использования вычислительной техникой.

Глава 2. Разработка нового алгебраического подхода к определению изоморфизма графов. Введение новых понятий

В нашей работе мы будем отталкиваться от матричного определения графа как класса перестановочно подобных квадратных (0; 1) матриц.

Предлагается полиномиальный эвристический вероятностный алгоритм решения задачи изоморфизма графов, такой, что при ответе «да» возможна ошибка с вероятностью, гипотетически (то есть как предположение эвристики) не превышающей вероятность сбоя самого компьютера, а ответ «нет» дает полную гарантию его правильности. Главная идея предлагаемого алгоритма – это рассмотрение полугруппы преобразований квадратной матрицы, перестановочных с любым преобразованием изоморфизма (то есть одинаковой перестановки ее строк и столбцов), и введение новых понятий элементного спектра и мультиспектра матрицы.

Для вещественной матрицы B через Sp(B) будем обозначать множество значений, которым равны ее элементы, а через MSp(B) – мультимножество этих значений, включающее кратность каждого значения. Назовем эти множества *элементным спектром* и *элементным мультиспектром* матрицы B соответственно.

Алгоритм:

1. Для данных двух простых неориентированных графов G1 и G2 с n вершинами, матрицы смежности которых A_1 и A_2 соответственно, построим две последовательности вещественных матриц $A_1^{(i)}$ и $A_2^{(i)}$, так что $A_1^{(0)} = A_1$ и $A_2^{(0)} = A_2$.

2. Устанавливаем, верно ли равенство $MSp(A_1^{(i)}) = MSp(A_2^{(i)})$. Если это не так, графы G_1 и G_2 определенно не изоморфны. Если равенство верно переходим к шагу 3.

3. Создаем случайный вектор y, и в обеих матрицах $A_1^{(i)}$ и $A_2^{(i)}$ заменяем каждый элемент, значение которого является j-м элементом в $Sp(A_1^{(i)})$, на y_j.

4. Выбираем случайный многочлен p(t) степени n–1 и задаем следующую пару матриц посредством равенств $A_1^{(i+1)} = p(A_1^{(i)})$ и $A_2^{(i+1)} = p(A_2^{(i)})$. После чего начинаем алгоритм заново.

Завершаем весь процесс, либо когда мультиспектры элементов двух текущих матриц становятся различными, либо когда количество элементов их общего спектра элементов больше не увеличивается от шага к шагу, то есть $MSp(A_1^{(k)}) = MSp(A_1^{(k-1)})$.

Таким образом, наш алгоритм решает задачу об изоморфизме графов, используя задачу об изоморфизме матриц смежности данных графов. В предложенном алгоритме мы сравниваем множества элементов матриц смежности графов, тем самым определяя, изоморфны наши графы или нет.

Глава 3. Программная реализация алгоритма

3.1. Составление поэтапного плана программы

С целью реализации указанного алгоритма необходимо написать компьютерную программу. Мы остановились на языке Python. Предварительно был составлен поэтапный план:

1. Сравнение мультиспектров:

1.1. Разделение каждой матрицы на элементы с последующим занесением их в список (упорядоченную изменяемую коллекцию объектов). В соответствии с определением назовем данные списки *элементными спектрами матриц* (см. главу 2)

1.2. Упорядочивание элементов в каждом из списков (при помощи встроенной функции «sorted»).

1.3. Сравнение соответственных элементов в списках.

1.4. При отличии списков друг от друга – завершение работы программы, вывод об отсутствии изоморфизма между графами. В ином случае выполняется переход к пункту 2 настоящего плана.

2. Создание числовой последовательности и замена элементов матриц:

2.1. Создание списка, содержащего подряд идущие числа. В дальнейшем данный список будем называть *последовательностью*.

2.2. Замена элементов в элементных спектрах матриц на соответственные элементы из последовательности.

2.3. Замена элементов матрицы на элементы из преобразованных элементных спектров.

3. Вычисление многочлена от матрицы:

3.1. Умножение и сложение матриц с использованием модуля NumPy.

3.2. С полученными после реализации пункта 3.1 матрицами производим действия, указанные в пунктах 1–2.

Блок-схема вышеприведенного поэтапного плана представлена на рисунке 1.

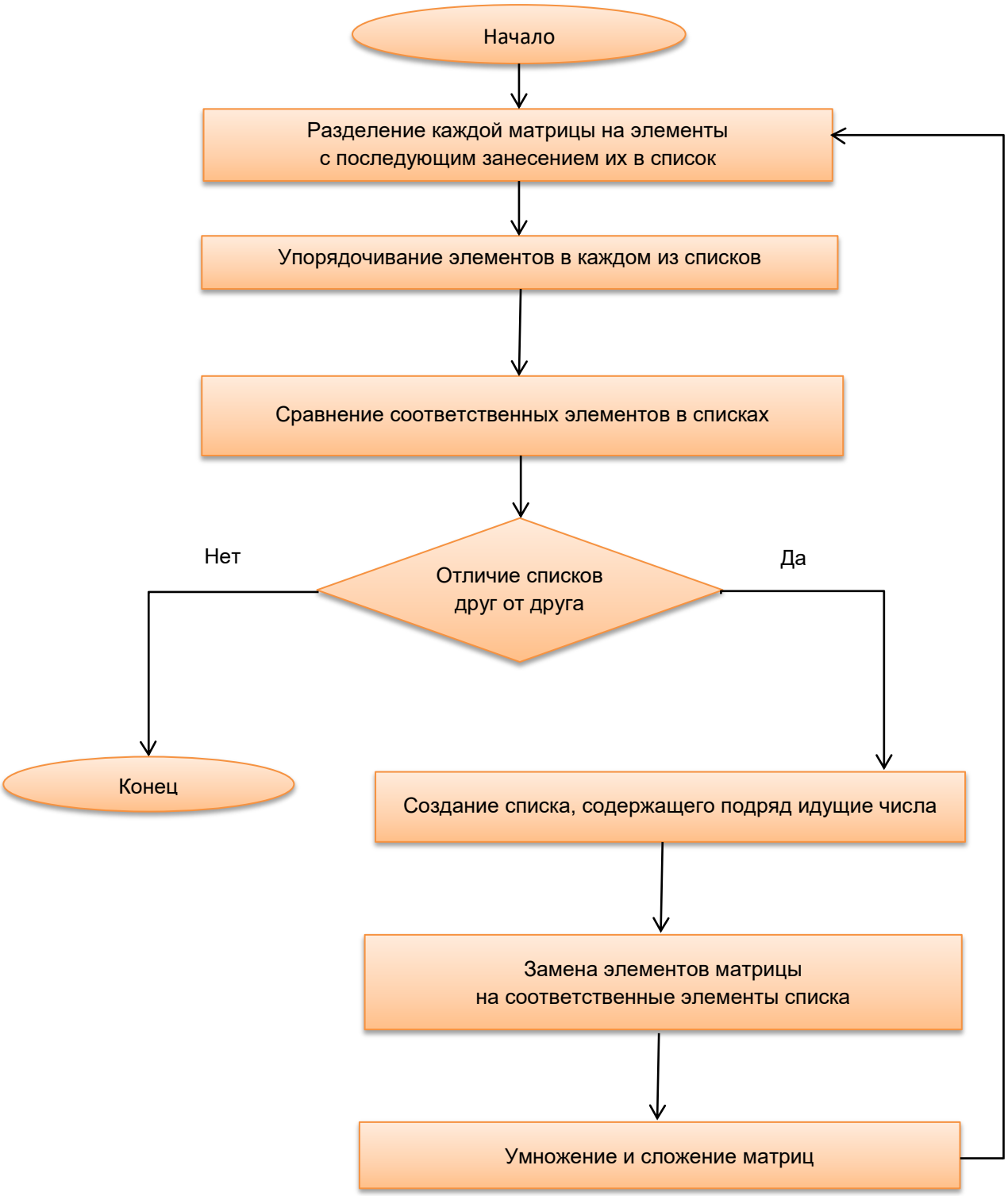

Рис. 1. Блок-схема поэтапного плана к компьютерной программе

3.2. Написание программы

На рисунке 2 представлена основная программа, написанная для реализации алгоритма. На вход программы подаются две квадратные матрицы размером n×n, после чего запускается цикл, который будет выполнять действия согласно блок-схеме.

```
isomorfism = True
n = int(input())
matrixa = []
matrixb = []
for i in range(n):
    string = input()
    matrixa.append(string.split(' '))
for i in range(n):
    string = input()
    matrixb.append(string.split(' '))
for i in range(n * n - 1):
    elementa = list(razdelenie_matrix(matrixa))
    elementb = list(razdelenie_matrix(matrixb))
    spa = poryadok(elementa)
    spb = poryadok(elementb)
    mspa = multispector(spa, matrixa)
    mspb = multispector(spb, matrixb)
    if len(mspa) == len(mspb):
        if not (sravnenie(mspa, mspb)):
            isomorfism = False
            break
        else:
            vector = spisok(spa)
            matrixa = zamena(spa, vector, matrixa)
            matrixb = zamena(spb, vector, matrixb)
            matrixa = np.array(matrixa)
            matrixb = np.array(matrixb)
            matrixa = polinom(matrixa)
            matrixb = polinom(matrixb)
    else:
        isomorfism = False
if isomorfism:
    print("Графы изоморфны")
else:
    print("Графы не изоморфны")
```

Рис. 2. Основная программа

Рассмотрим подробнее написанные невстроенные функции. Первая функция «razdelenie_matrix» (рис. 3) создает множество из элементов матрицы и убирает повторяющиеся элементы.

```
def razdelenie_matrix(matrix):
    element = set()
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            element.add(matrix[i][j])
    return element
```

Рис. 3. Функция «razdelenie_matrix»

После сортировки множества элементов в работу вступает функция «multispector» (рис. 4), которая, получая на вход матрицу и ее элементный спектр, возвращает элементный мультиспектр в виде двумерного списка. Каждым его элементом является список, содержащий число из матрицы и количество вхождений этого числа в матрицу.

```
def multispector(sp, matrix):
    all = []
    msp = []
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            all.append(matrix[i][j])
    msplow = []
    for i in range(len(sp)):
        msplow.append(all.count(sp[i]))
    for i in range(len(sp)):
        msp.append([sp[i], msplow[i]])
    return msp
```

Рис. 4. Функция «multispector»

После вычисления элементного мультиспектра обеих матриц выполняется функция «sravnenie» (рис. 5), которая принимает элементные мультиспектры двух матриц и возвращает True или False в зависимости от того идентичны полученные данные или нет. Стоит также сказать, что за количество итераций цикла в функции принято количество элементов одного из подаваемых на вход списков. Таким образом, если длины списков различны, то программа выдаст

ошибку. Этот случай решен в основном цикле программы, и когда длины элементных мультиспектров различаются, действие цикла останавливается, а графы считаются не изоморфными.

```
def sravnenie(mspa, mspb):
    isomorf = True
    for i in range(len(mspa)):
        for j in range(2):
            if int(mspa[i - 1][j - 1]) != int(mspb[i - 1][j - 1]):
                isomorf = False
    return isomorf
```

Рис. 5. Функция «sravnenie»

Если действие программы не прекратилось, и элементные мультиспектры матриц оказались равны, вызывается функция «zamena» (рис. 6), действие которой заключается в замене элементов в матрице на соответственные элементы из списка с подряд идущими числами.

```
def zamena(spa, spisok, matrix):
    for j in range(len(matrix)):
        for g in range(len(matrix[j])):
            for i in range(len(spa)):
                if spa[i] == matrix[j][g]:
                    matrix[j][g] = spisok[i]
    return matrix
```

Рис. 6. Функция «zamena»

Крайней функцией в нашей программе является «polinom» (рис. 7), она возводит матрицу в степень, равную количеству ее строк –1. Умножение матрицы реализовано при помощи библиотеки «nampy». «Polinom» возвращает полученную после вычислений матрицу, с которой начинает заново работу основной цикл программы.

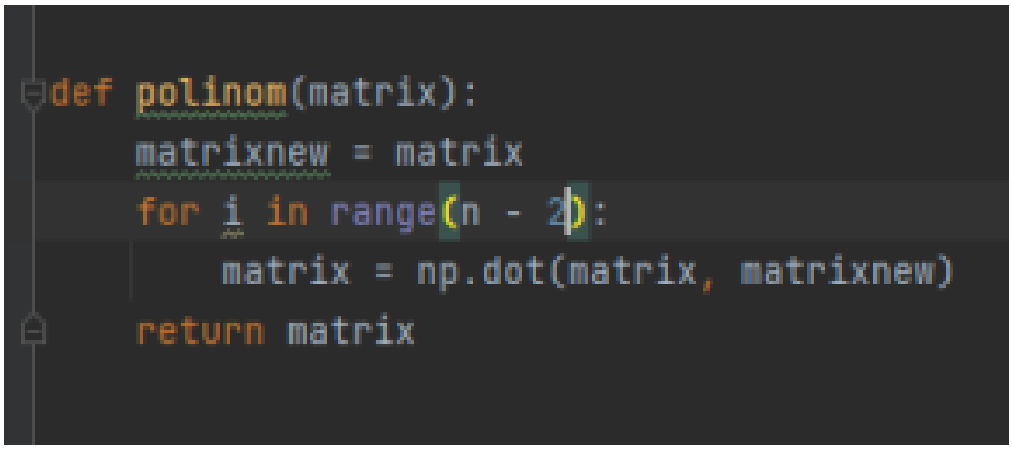

Рис. 7. Функция «polinom»

Глава 4. Практическая проверка применения разработанного алгоритма определения изоморфизма графов

Лучшей проверкой любого предлагаемого алгоритма для решения переборной задачи является его практическая проверка на известных сложных примерах или случайных примерах из класса, представляющего определенный интерес для исследователей.

Предположим, имеются два графа G_1 и G_2 с матрицами смежности (рис. 2):

$$A_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \text{ и } A_2 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

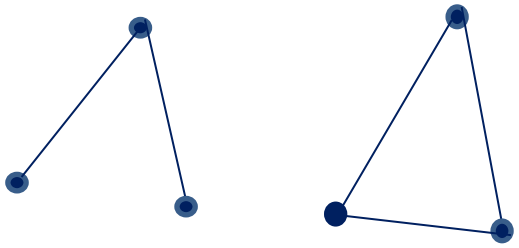

Рис. 8. Графы G_1 G_2, изоморфизм которых требуется установить

Для использования разработанного нами алгоритма построим матрицы смежности этих графов. После чего будем действовать согласно нашему алгоритму. Возьмем случайный многочлен p(t) степени 2, $p(t) = t^2 - 2t$. Затем вычислим:

$$p(A_1{}^{(0)}) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} - 2 \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 2 & -2 \\ 1 & -2 & 1 \end{pmatrix}$$

$$p(A_2{}^{(0)}) = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} - 2 \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

Далее сравниваем элементные мультиспектры матриц $MSp(A_1{}^{(1)}) = \{(-2)^4, 1^4, 2^1\}$, $MSp(A_2{}^{(1)}) = \{(-1)^6, 2^3\}$,

Как мы видим, $MSp(A_1{}^{(1)})$ не равен $MSp(A_2{}^{(1)})$, а значит графы G_1 и G_2 не изоморфны.

Удостовериться в этом также можно посмотрев на представленные графы. Так, в графе G_1 два ребра, а в графе G_2 – три, уже поэтому они не изоморфны.

При большем количестве вершин в графах необходимо будет выполнить намного больше действий, поэтому для практической проверки предложенного алгоритма нужно написать программу, реализующую данный алгоритм.

Заключение

Данная работа предлагает принципиально новый полиномиальный алгоритм, позволяющий решать задачу изоморфизма графов при помощи вычислительной техники.

Выводы:

1. Проведенный анализ литературных источников показал, что проблема изоморфизма графов остается актуальной, но несмотря на это простой, единый алгоритм (с возможностью применения вычислительными машинами), до сих пор не найден.

2. Предложенный нами алгоритм способен установить изоморфизм двух заданных графов, что подтверждено путем тестирования программы, написанной для реализации разработанного алгоритма.

Список использованных источников и литературы

1. Babai L. Graph Isomorphism in Quasipolynomial Time. 2015. arXiv:1512.03547, Bibcode:2015arXiv151203547B.

2. McKay B.D. Practical graph isomorphism // Congressus Numeratium. 1981. Vol. 30. P. 45–87.

3. Герман О.В., Дунаев А.А. Задача изоморфизма графов в системе нечеткого распознавания // Труды БГТУ. 2016. № 6. С. 181–184.

4. Карелин В.П. Задача распознавания изоморфизма графов. Прикладное значение и подходы к решению // Вестник Таганрогского института управления и экономики. 2015. № 1. С. 102–106.

5. Курапов С.В., Давидовский М.В. Вычислительные методы определения инвариантов графа // International Journal of Open Information Technologies. 2021. Vol. 9, № 2. P. 1–8.

6. Мехович А.П., Караулова Т.Б. Элементы теории графов: методические рекомендации. Витебск: ВГУ имени П.М. Машерова, 2020. 48 с.

6. 7. Опубликован быстрый алгоритм для задачи изоморфизма графов. URL: https://m.habr.com/ru/post/273231/ (дата обращения: 05.05.2021).

7. 8. Погожев С.В., Хитров Г.М. О проблеме изоморфизма графов и об одном матричном алгоритме ее решения // Вестник СПбГУ. Серия 10. 2008. Вып. 4. С. 80–83.

8. 9. Погребной В.К. Решение задачи определения изоморфизма графов, представленных атрибутными матрицами // Известия Томского политехнического университета. 2012. Т. 321, № 5. С. 52–56.

9. 10. Пономаренко И.Н. Проблема изоморфизма графов: алгоритмические аспекты (записки к лекциям). Санкт-Петербург, 2010. 57 с.

10. 11. Проверка изоморфности двух графов и поиск изоморфных подграфов: подход на основе анализа NB Paths [Электронный текст]. URL:https://habr.com/ru/post/491846/ (дата обращения: 05.05.2021).

11. 12. Трофимов М.И., Смоленский Е.А. Применение индексов электроотрицательности органических молекул в задачах химической информатики // Известия Академии наук. Серия химическая. 2005. С. 2166–2176.