Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра ПМиК

Расчетно-графическая работа по дисциплине «Защита информации» на тему «Доказательство с нулевым знанием»

«Вариант 2»

Выполнил: студент 4 курса
ИВТ, гр. ИП-813
Бурдуковский И.А.
Проверил: Старший преподаватель кафедры ПМиК
Дьячкова Ирина Сергеевна
.

Новосибирск 2021

Содержание

Постановка задачи

Необходимо написать программу, реализующую алгоритм доказательства с нулевым знанием «Гамильтонов цикл».

Обе рассматриваемые задачи являются NP-полными и не имеют быстрых методов для решения, поэтому для тестирования необходимо будет генерировать правильные решения при помощи дополнительно разработанных программ.

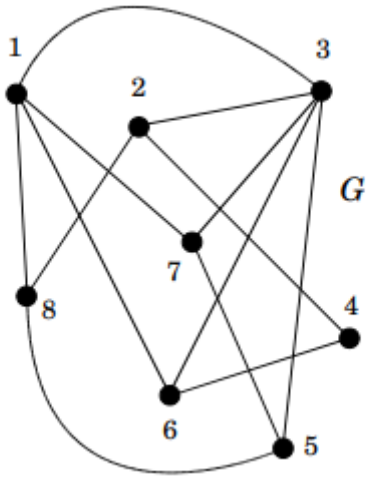
Вне зависимости от варианта задания, необходимо информацию о графах считывать из файла. В файле описание графа будет определяться следующим образом:

1) в первой строке файла содержатся два числа n < 1001 и $m \leq n^2$, количество вершин графа и количество рѐбер соответственно;
2) в последующих m строках содержится информация о рѐбрах графа, каждое из которых описывается с помощью двух чисел (номера вершин, соединяемых этим ребром);
3) в зависимости от варианта указывается необходимая дополнительная информация: в первом варианте перечисляются цвета вершин графа; во втором варианте указывается последовательность вершин, задающая гамильтонов цикл (этот пункт можно вынести в отдельный файл).

Теоретические сведения

Рассматриваемая в данном разделе задача не просто предоставляет нам возможность описать еще одну схему построения протокола доказательства с нулевым знанием, но и имеет важное теоретическое значение. Блюм (Manuel Blum) показал, что, выражаясь неформально, любое математическое утверждение может быть представлено в виде графа, причем доказательство этого утверждения соответствует гамильтонову циклу в этом графе. Поэтому наличие протокола доказательства с нулевым знанием для гамильтонова цикла означает, что доказательство любого математического утверждения может быть представлено в форме доказательства с нулевым знанием.

Определение 5.1. Гамильтоновым циклом в графе называется непрерывный путь, проходящий через все вершины графа ровно по одному разу.



Путь, проходящий последовательно через вершины 8, 2, 4, 6, 3, 5, 7, 1, представляет собой гамильтонов цикл. Действительно, в этом Криптографические протоколы пути содержатся все вершины графа, и каждая вершина посещается только один раз. ut

Ясно, что если в графе G с n вершинами гамильтонов цикл существует, то при некоторой нумерации вершин он пройдет точно через вершины с последовательными номерами 1, 2, 3, . . . , n. Поэтому путем перебора всех возможных нумераций вершин мы обязательно найдем гамильтонов цикл. Но количество возможных нумераций равно n!, и поэтому уже при умеренно больших n, например, при n = 100, такой подход становится практически нереализуемым. Доказано, что задача нахождения гамильтонова цикла в графе является NP-полной. Мы уже говорили кратко о понятии NP-полноты. Неформально, NP-полнота рассматриваемой задачи означает, что для ее решения не существуют (точнее, неизвестны) алгоритмы существенно более быстрые, чем указанный метод перебора.

Нашей задачей будет построение криптографического протокола, с помощью которого Алиса будет доказывать Бобу, что она знает гамильтонов цикл в некотором графе G так, чтобы Боб не получил никаких знаний о самом этом цикле. Иными словами, Алиса будет предоставлять Бобу доказательство с нулевым знанием. Еще раз

напомним читателю, что «нулевое знание» означает, что независимо от числа реализаций протокола доказательства Боб будет располагать точно такими же сведениями о гамильтоновом цикле, какие он мог бы получить, просто изучая представленный ему граф G.

Итак, допустим, что Алиса знает гамильтонов цикл в графе G. Теперь она может это доказывать Бобу и всем, кто имеет граф G, с помощью описываемого ниже протокола. Алиса может использовать это доказательство, например, для идентификации своей личности. Но прежде чем мы перейдем к описанию протокола, договоримся о некоторых обозначениях.

Мы будем обозначать графы буквами G, H , F , понимая под этим одновременно соответствующие матрицы смежности. Элемент матрицы Hij = 1, если в графе H есть ребро, соединяющее вершины i и j ; Hij = 0 в противном случае. Символом k будем обозначать конкатенацию (сцепление) двух чисел, точнее, двоичных слов, им соответствующих. Нам понадобится шифр с открытым ключом. Вообще говоря, это может быть любой шифр, но для определенности будем использовать шифр RSA. Будем считать, что Алиса сформировала систему RSA с открытыми параметрами N и d. Важно, что зашифрованные в этой системе сообщения может расшифровать только Алиса и больше никто.

Протокол доказательства состоит из следующих четырех шагов (пояснения будут даны ниже).

Шаг 1. Алиса строит граф H , являющийся копией исходного графа G, где у всех вершин новые, случайно выбранные номера. На языке теории графов говорят, что H изоморфен G. Иными словами, H получается путем некоторой перестановки вершин в графе G (с сохранением связей между вершинами). Алиса кодирует матрицу H , приписывая к первоначально содержащимся в ней нулям и единицам случайные числа rij по схеме $\tilde{H}_{ij} = r_{ij} \| H_{ij}$. Затем она шифрует элементы матрицы H˜ , получая зашифрованную матрицу F , $F_{ij} = \tilde{H}_{ij}^{d} \bmod N$.

Матрицу F Алиса передает Бобу.

Шаг 2. Боб, получив зашифрованный граф F , задает Алисе одиниз двух вопросов.

1. Каков гамильтонов цикл для графа H ?

2. Действительно ли граф H изоморфен G?

Шаг 3. Алиса отвечает на соответствующий вопрос Боба.

1. Она расшифровывает в F ребра, образующие гамильтонов цикл.

2. Она расшифровывает F полностью (фактически передает Бобу граф H˜) и предъявляет перестановки, с помощью которых граф H был получен из графа G.

Шаг 4. Получив ответ, Боб проверяет правильность расшифровки путем повторного шифрования и сравнения с F и убеждается либо в том, что показанные ребра

действительно образуют гамильтонов цикл, либо в том, что предъявленные перестановки действительно переводят граф G в граф H . Весь протокол повторяется t раз. Обсудим вначале кратко несколько вопросов по построению протокола.

1. Зачем Алиса строит изоморфный граф? Если бы она этого не делала, то Боб, получив ответ на свой вопрос номер один, узнал бы гамильтонов цикл в графе G.
2. Зачем Алиса кодирует матрицу H? С этим приемом мы уже встречались при шифровании цветов вершин графа. Дело в том, что невозможно зашифровать непосредственно нули и единицы (с помощью шифра RSA они вообще не шифруются). Даже если заменить их на какие-то произвольные числа a и b, то мы получим всего два различных шифротекста, и Бобу не составит труда понять, какой из них какому числу соответствует. Т.е. структура графа не будет скрыта. Здесь мы сталкиваемся с типичной ситуацией, когда требуется использовать так называемый рандомизированный шифр. И такой шифр строится путем добавления случайных чисел в матрицу H перед шифрованием. Закодированная матрица H˜ точно также задает граф (нечетность числа означает наличие ребра, четность — его отсутствие), но после шифрования H˜ структура графа полностью скрывается (мы используем известное свойство шифра RSA — он полностью скрывает четность числа).
3. Зачем Боб задает два вопроса? Если бы он задавал только вопрос номер один, который по смыслу протокола является основным, то Алиса, не зная в действительности гамильтонова цикла в графе G, могла бы предъявить Бобу совсем другой граф с таким же количеством вершин и искусственно заложенным в него гамильтоновым циклом. Поэтому Боб иногда просит Алису доказать изоморфизм графов H и G. Важно, что Алиса не знает заранее, какой из двух вопросов задаст Боб.
4. Почему Боб не может задать сразу двух вопросов? В этом случае он узнал бы гамильтонов цикл в G, так как ему был бы показан гамильтонов цикл в H и правило перехода от H к G.
5. Зачем Боб проверяет правильность расшифровки? Если бы он этого не делал, то Алиса на четвертом шаге могла бы предоставить ему «выгодную» для себя информацию, а не ту, которую она посылала ему на втором шаге.

Более точно основные детали протокола обосновываются в ходе доказательства двух основных утверждений.

Д о к а з а т е л ь с т в о . Вначале покажем, что вероятность обмана в одной реализации протокола равна 1/2. Заметим, что если Алиса действительно знает гамильтонов цикл в графе G, то она может правильно ответить на любой вопрос Боба. Если же она не знает гамильтонов цикл, то самое большее, что она может сделать, — это подготовиться к ответу на первый либо на второй вопрос. В ожидании первого вопроса, она создает новый граф с искусственно заложенным в него гамильтоновым циклом. Но в этом случае она не сможет доказать его изоморфизм графу G. В ожидании второго вопроса, она строит граф, изоморфный графу G. Но в этом случае она не сможет показать в нем гамильтонов цикл. Таким образом, вероятность успешности обмана равна вероятности угадывания номера вопроса. В предположении,

что Боб задает оба вопроса с одинаковыми вероятностями, мы получаем, что вероятность обмана равна.

Д о к а з а т е л ь с т в о . Чтобы доказать, что Боб не получает никаких знаний в ходе реализации протокола, достаточно показать, что все, что он получает от Алисы, он мог бы получить сам, не вступая с ней ни в какое общение.

Рассмотрим вначале второй вопрос Боба. В ответ на этот вопрос он получает граф, изоморфный графу G. Но он сам мог строить сколько угодно изоморфных графов, и то, что присылает ему Алиса, это просто один из них.

Случай с первым вопросом не столь очевиден. В ответ на первый вопрос Боб получает гамильтонов цикл в графе, изоморфном графу G. На первый взгляд может показаться, что это дает Бобу какуюто информацию. Однако это не так. Заметим, что если в G есть

гамильтонов цикл, то при некоторой нумерации вершин существует

$$\begin{pmatrix} * & 1 & * & \cdots & * & * & * \\ * & * & 1 & \cdots & * & * & * \\ & & & \cdots & & & \\ * & * & * & \cdots & * & 1 & * \\ * & * & * & \cdots & * & * & 1 \\ 1 & * & * & \cdots & * & * & * \end{pmatrix},$$

где $*$ означает неопределенность в наличии или отсутствии ребра. Т.е. при такой нумерации гамильтонов цикл проходит через вершины в порядке возрастания номеров. Изменяя нумерацию вершин, Боб может получать из всевозможные изоморфные матрицы. Когда Алиса, отвечая на его первый вопрос, открывает гамильтонов цикл, Боб видит как раз одну из таких матриц. Таким образом, Боб не получает от Алисы никакой информации, которую он не мог бы получить сам. ut
Рассмотрим пример, иллюстрирующий все основные этапы описанного протокола.

П р и м е р. Возьмем в качестве основного граф G, изображенный на Его матрица смежности имеет вид

$$G = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & \boxed{1} \\ 0 & 0 & 1 & \boxed{1} & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & \boxed{1} & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & \boxed{1} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \boxed{1} & 1 \\ 1 & 0 & \boxed{1} & 1 & 0 & 0 & 0 & 0 \\ \boxed{1} & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & \boxed{1} & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

В матрице с помощью · показан гамильтонов цикл. Алиса выбирает некоторую случайную нумерацию вершин, скажем, 7, 4, 5, 3, 1, 2, 8,

$$H = \begin{array}{c} \\ 7 \\ 4 \\ 5 \\ 3 \\ 1 \\ 2 \\ 8 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccccc} 7 & 4 & 5 & 3 & 1 & 2 & 8 & 6 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 1 & \boxed{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \boxed{1} \\ \boxed{1} & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & \boxed{1} & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & \boxed{1} & 1 \\ 0 & \boxed{1} & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & \boxed{1} & 0 & 0 \\ 0 & 1 & 0 & \boxed{1} & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

Для шифрования матрицы будем использовать систему RSA с параметрами N = 55, d = 3. Вначале закодируем матрицу H . В рамках данного примера просто припишем слева к каждому элементу матрицы выбираемую случайно с равными вероятностями цифру из множества {1, 2, 3, 4, 5}:

$$\tilde{H} = \begin{pmatrix} 50 & 20 & 11 & 31 & 21 & 40 & 20 & 10 \\ 40 & 30 & 50 & 20 & 10 & 41 & 50 & 21 \\ 41 & 30 & 50 & 11 & 30 & 20 & 51 & 40 \\ 11 & 10 & 41 & 30 & 51 & 41 & 30 & 21 \\ 31 & 20 & 40 & 11 & 50 & 10 & 41 & 31 \\ 50 & 41 & 20 & 21 & 40 & 10 & 21 & 50 \\ 40 & 30 & 31 & 50 & 41 & 21 & 30 & 40 \\ 20 & 41 & 10 & 51 & 41 & 20 & 30 & 40 \end{pmatrix}.$$

Теперь мы шифруем матрицу H˜ , возводя каждый ее элемент в куб по модулю 55:

$$F = \begin{pmatrix} 40 & 25 & 11 & 36 & 21 & 35 & 25 & 10 \\ 35 & 50 & 40 & 25 & 10 & 06 & 40 & 21 \\ 06 & 50 & 40 & 11 & 50 & 25 & 46 & 35 \\ 11 & 10 & 06 & 50 & 46 & 06 & 50 & 21 \\ 36 & 25 & 35 & 11 & 40 & 10 & 06 & 36 \\ 40 & 06 & 25 & 21 & 35 & 10 & 21 & 40 \\ 35 & 50 & 36 & 40 & 06 & 21 & 50 & 35 \\ 25 & 06 & 10 & 46 & 06 & 25 & 50 & 35 \end{pmatrix}.$$

(При внимательном просмотре матрицы F может показаться, что использованный нами шифр плохо скрывает исходную матрицу . Криптографические протоколы Это объясняется тем, что, во-первых, модуль 55 слишком мал и, во вторых, в матрице H˜ много чисел, не взаимно простых с модулем.

Ход работы

Реализованные методы в программе rgr.py

def check_prime(p):
Функция проверяет число на простоту

def generate_prime(left, right):
Функция возвращает случайное простой число из передаваемой границы

def generate_coprime(p):
Функция возвращает случайное число взаимно-простое с передаваемым

def pow_module(a, x, p):
Функция возводит в степень по модулю

def gcd(a, b):
Функция - Алгоритм Евклида

def gcd_modified(a, b):
Функция - Модифицированный Алгоритм Евклида

def Fill_Graph()
Функция, создающая сам граф в виде смежной матрицы. Сначала создается матрица NxN заполненная нулями. Затем прочитывается файл с информацией о путях графа и эти пути заполняют матрицу: выставляются единицы(ребра) соединяющие наши вершины. Функция возвращает количество вершин N, ребер M, сам граф Graph и его копию, для проверки в последующих функциях.

def All_Path(Graph)
Функция создает словарь из путей вершин. В функцию поступает граф в виде смежной матрицы. Проходя по двум циклам, алгоритм создает и записывает в словарь вершину (она же служит ключом словаря) и ее пути до других вершин.

Из функции возвращается словарь вида:

1: [3, 6, 7, 8]

2: [3, 4, 8]

3: [1, 2, 5, 6, 7]

def Search_Gamilton_Cycle(G, size, pt, path=[])
Данная функция ищет Гамильтонов цикл по графу. На вход поступают:
1)Словарь связей вершин
2)Количество вершин, которое необходимо обойти
3)Выбор стартовой вершины
4) Пустой список, в который в будущем будет записан цикл прохождения по графу.

Программа начинает с заданной вершины, записывает ее в список, затем проверяет связи этой вершины с другими вершинами. Если связи есть, то начинает шагать вперед, записывая вершины в список, до тех пор, пока элементов в списке не станет равно количеству вершин N нашего графа. Запись осуществляется без повтора вершин. В случае, если цикл зашел в тупиковую вершину и ему некуда шагнуть, кроме

как назад, он вернет список, в котором будет видно, что цикл не был пройден и вновь попытается найти цикл.

def RSA_Matrix_Encode(NewGraph, x, p, N)
Функция кодирует элементы матрицы при помощи алгоритма RSA и возвращает результат.

def Gamilton_Cycle()
Основная часть программы.
Алгоритм программы эмулирует общение двух людей.
Сначала функция получает значения количества вершин N, ребер M, граф G и его копию граф H.
Вторым шагом, по основному графу находится Гамильтонов цикл. Затем столбцы смежной матрицы переставляются и порядковые переставления записываются в список, получая Изоморфный граф.
Затем к каждому элементу матрицы дописывается число от 1 до N (случайное число на каждый элемент столбца матрицы одно и то же, но для каждого столбца уникально), чтобы каждый элемент можно было закодировать по алгоритму RSA.
После кодирования, матрица готова к отправке Бобу.
Боб, получив матрицу, задает один из двух вопросов Алисе:

1) Каков Гамильтонов цикл для графа?
2) Действительно ли граф H изоморфен G

Выбрав первый вопрос, Алиса отправляет Бобу Гамильтонов цикл и матрицу H. Боб преобразует матрицу H, выбирая элемент по порядку из Гамильтонова цикла и проходя по алгоритму RSA. Полученную матрицу он сравнивает с той, что получил изначально, она же матрица F. Если они идентичны, то Боб проходит по матрице при помощи Гамильтонова цикла. Если это возможно, то алгоритм отработал верно.

Выбрав второй вопрос, Алиса отправляет Бобу матрицу H, которая еще не закодирована. Боб кодирует матрицу. Если эта матрица равна той, что он получил изначально, то Боб продолжает преобразовывать матрицу.
Следующим шагом он отрбрасывает от каждого элемента матрицы все разряды, оставляя разряд единиц в качестве элемента матрицы. Следом Алиса отправляет ему список переставленных порядков матрицы и Боб переставляет столбцы матрицы в соответствии с этим списком. Тем самым Боб сможет убедиться, что матрицы идентичны.

Результат работы программы

```
Стартовый граф:
    1   2   3   4   5   6   7   8
 1  0   0   0   0   0   0   0   0
 2  0   0   0   0   0   0   0   0
 3  0   0   0   0   0   0   0   0
 4  0   0   0   0   0   0   0   0
 5  0   0   0   0   0   0   0   0
 6  0   0   0   0   0   0   0   0
 7  0   0   0   0   0   0   0   0
 8  0   0   0   0   0   0   0   0

Граф после заполнения Алисой:
    1   2   3   4   5   6   7   8
 1  0   0   1   0   0   1   1   1
 2  0   0   1   1   0   0   0   1
 3  1   1   0   0   1   1   1   0
 4  0   1   0   0   0   1   0   0
 5  0   0   1   0   0   0   1   1
 6  1   0   1   1   0   0   0   0
 7  1   0   1   0   1   0   0   0
 8  1   1   0   0   1   0   0   0

N = 8, M = 13

Переходы:
1: [3, 6, 7, 8]
2: [3, 4, 8]
3: [1, 2, 5, 6, 7]
4: [2, 6]
5: [3, 7, 8]
6: [1, 3, 4]
7: [1, 3, 5]
8: [1, 2, 5]

Гамильтонов цикл: [1, 3, 6, 4, 2, 8, 5, 7]
```

Рис.1 Заполнение первоначального графа Алисой

```
-------------------------------------
Действия первого абонента - Алисы
-------------------------------------
Алиса рандомит вершины графа: [4, 5, 6, 7, 8, 3, 2, 1]

Изоморфный граф:
    1   2   3   4   5   6   7   8
 1  0   0   1   0   0   0   1   0
 2  0   0   0   1   1   1   0   0
 3  1   0   0   0   0   1   0   1
 4  0   1   0   0   0   1   0   1
 5  0   1   0   0   0   0   1   1
 6  0   1   1   1   0   0   1   1
 7  1   0   0   0   1   1   0   0
 8  0   0   1   1   1   1   0   0

Перед кодировкой алгоритмом RSA
Припишем рандомное число из списка [6, 5, 8, 3, 2, 7, 4, 1]

Подготовленная матрица до RSA:
    1   2   3   4   5   6   7   8
 1  60  50  81  30  20  70  41  10
 2  60  50  80  31  21  71  40  10
 3  61  50  80  30  20  71  40  11
 4  60  51  80  30  20  71  40  11
 5  60  51  80  30  20  70  41  11
 6  60  51  81  31  20  70  41  11
 7  61  50  80  30  21  71  40  10
 8  60  50  81  31  21  71  40  10

Алиса генерирует ключи;
Ключ N:  450037907881631623
Ключ c:  243121709217860487
```

Рис.2 Преобразование графа

```
Матрица после RSA, для Боба:
    1   2   3   4   5   6   7   8
 1  402199386122675776  71940143396850711   413895197286004174  375169053709444434  74748914507834204   105922755062827207  246709742252585182  395846489957143112
 2  402199386122675776  71940143396850711   245034479614197179  428435915869502869  291917452398434305  117286799883151922  238833863656876094  395846489957143112
 3  228734954366875964  71940143396850711   245034479614197179  375169053709444434  74748914507834204   117286799883151922  238833863656876094  443684515129071026
 4  402199386122675776  236440561546151157  245034479614197179  375169053709444434  74748914507834204   117286799883151922  238833863656876094  443684515129071026
 5  402199386122675776  236440561546151157  245034479614197179  375169053709444434  74748914507834204   105922755062827207  246709742252585182  443684515129071026
 6  402199386122675776  236440561546151157  413895197286004174  428435915869502869  74748914507834204   105922755062827207  246709742252585182  443684515129071026
 7  228734954366875964  71940143396850711   245034479614197179  375169053709444434  291917452398434305  117286799883151922  238833863656876094  395846489957143112
 8  402199386122675776  71940143396850711   413895197286004174  428435915869502869  291917452398434305  117286799883151922  238833863656876094  395846489957143112
```

Рис.3 Преобразование графа Алисой

```
-------------------------------------
Действия Второго абонента - Боба
-------------------------------------
Боб получил матрицу F

Какой вопрос выберет Боб?
1.'Алиса, каков Гамильтонов цикл для графа H?'
2.'Алиса, действительно ли граф H изоморфен G?'
1
-'Алиса, покажи Гамильтонов цикл?'
Алиса отсылает Бобу Гамильтонов цикл
Гамильтонов цикл: [1, 3, 6, 4, 2, 8, 5, 7]
Боб проходит по Матрице H-штрих, преобразуя соответствующий элемент цикла
Если элементы Матрицы F равны этим преобразованным, то Боб пытается по данному циклу пройти свой граф
Если ему это удается, то Все впорядке. Если нет, то возникла ошибка в алгоритме

Матрицы идентичны, т.к. flag = True
Гамильтонов цикл Алисы: [1, 3, 6, 4, 2, 8, 5, 7]
Гамильтонов цикл Боба: [1, 3, 6, 4, 2, 8, 5, 7]

Process finished with exit code 0
```

Рис.4 Первый вопрос Алисе от Боба

```
--------------------------------------
Действия Второго абонента - Боба
--------------------------------------
Боб получил матрицу F

Какой вопрос выберет Боб?
1.'Алиса, каков Гамильтонов цикл для графа H?'
2.'Алиса, действительно ли граф H изоморфен G?'
2
-'Докажи изоморфизм, Алиса?'

Алиса отсылает Бобу матрицу, которая еще не преобразования в RSA
И рандом столбцов, который она использовала после получения Гамильтонова цикла

Боб проверяет матрицы: сравнивает матрицы F и H-штрих путем
повторного шифрования и сравнения матриц

Матрицы идентичны, т.к. flag = True
Далее Боб отбрасывает все разряды кроме единичного от каждого элемента матрицы и получает матрицу,
изоморфную стартовой

    1   2   3   4   5   6   7   8
 1  0   1   0   1   0   1   1   0
 2  1   0   0   1   1   0   1   1
 3  0   0   0   0   0   0   1   1
 4  1   1   0   0   1   0   0   0
 5  0   1   0   1   0   1   0   0
 6  1   0   0   0   1   0   0   1
 7  1   1   1   0   0   0   0   0
 8  0   1   1   0   0   1   0   0

Затем Боб переставляет столбцы в соответствии с полученной нумерацией от Алисы

Рандом Алисы вершин графа: [1, 3, 4, 7, 5, 8, 6, 2]

Затем Алиса проверяет граф H,преобразуя его по ряду Алисы, и исходный граф Алисы
Если они идентичны, то из графа H мы получили граф G

Матрицы идентичны, т.к. flag = True

Process finished with exit code 0
```

Рис.5 Второй вопрос Алисе

Листинг кода

```
from collections import defaultdict
import random


# проверка на простоту числа используя теореме Ферма
def check_prime(p):
    if p <= 1:
        return False
    elif p == 2:
        return True
    a = random.randint(2, p - 1)
    # print(p, "-", a)
    if pow_module(a, (p - 1), p) != 1 or gcd(p, a) > 1:
        return False
    return True


# генерируем простое число в указанных границах
def generate_prime(left, right):
    while True:
        p = random.randint(left, right)
        # print("--", p)
        if check_prime(p):
            return p


# генерируем взаимно-простое число
def generate_coprime(p):
    result = random.randint(2, p)
    # print(result)
    while gcd(p, result) != 1:
        result = random.randint(2, p)
        # print(result)
    return result


# возведение в степень по модулю
def pow_module(a, x, p):
    result = 1
    a = a % p
    if a == 0:
        return 0;
    while x > 0:
        if x & 1 == 1:  # если крайний правый бит степени равен lib
            result = (result * a) % p
        a = (a ** 2) % p
        x >>= 1  # побитово смещаем степень
    return result


# Алгоритм Евклида, для нахождения наибольшего общего делителя
def gcd(a, b):
    while b != 0:
        r = a % b
        a = b
        b = r
    return a
```

```
# Обобщённый Алгоритм Евклида, для нахождения наибольшего общего делителя и двух неизвестных уравнения
def gcd_modified(a, b):
    U = (a, 1, 0)
    V = (b, 0, 1)
    while V[0] != 0:
        q = U[0] // V[0]
        T = (U[0] % V[0], U[1] - q * V[1], U[2] - q * V[2])
        U = V
        V = T
    return U


def Fill_Graph():
    try:
        with open('Path.txt', 'r') as f:
            nums = f.read().splitlines()
    except OSError:
        print("Ошибка открытия файла с графами")
        return -1
    # print(nums)

    N = int(nums[0])
    M = int(nums[1])
    id = list()
    firststr = "\t"

    for i in range(N):
        id.append(i + 1)
        firststr += str(i + 1) + "\t"
    # print(id)
    # print(firststr)

    Graph_G = [[0] * N for i in range(N)]
    Graph_H = [[0] * N for i in range(N)]
    # print(Graph_G)

    print("Стартовый граф:")
    print(firststr)
    for i in range(len(Graph_G)):
        print(" " + str(i + 1), end="\t")
        for j in range(len(Graph_G[i])):
            Graph_G[i][j] = 0
            print(Graph_G[i][j], end="\t")
        print("\n", end="")
    print()

    # заполнение графа Алисой
    for k in range(M):
        line = nums[k + 2]
        split_line = line.split(",")
        i = int(split_line[0])
        j = int(split_line[1])
        Graph_G[i - 1][j - 1] = 1
        Graph_G[j - 1][i - 1] = 1
        Graph_H[i - 1][j - 1] = 1
        Graph_H[j - 1][i - 1] = 1
    # print(Graph_G)
    # print(Graph_H)

    print("Граф после заполнения Алисой:")
```

```
    print(firststr)
    for i in range(len(Graph_G)):
        print(" " + str(i + 1), end="\t")
        for j in range(len(Graph_G[i])):
            print(Graph_G[i][j], end="\t")
        print("\n", end="")

    return N, M, Graph_G, Graph_H, firststr


def All_Path(Graph):
    Dict = defaultdict(list)
    for i in range(len(Graph)):
        for j in range(len(Graph[i])):
            if Graph[i][j] == 1:
                Dict[i].append(j + 1)
    return Dict


def Search_Gamilton_Cycle(G, size, pt, path=[]):
    if pt not in set(path):
        path.append(pt)
        if len(path) == size:
            return path
        for pt_next in G.get(pt - 1, []):
            res_path = [i for i in path]
            candidate = Search_Gamilton_Cycle(G, size, pt_next, res_path)
            if candidate is not None:
                return candidate


def RSA_Matrix_Encode(NewGraph, x, p, N):
    TempGraph = [[0] * N for i in range(N)]
    for i in range(len(NewGraph)):
        for j in range(len(NewGraph[i])):
            TempGraph[i][j] = pow_module(NewGraph[i][j], x, p)

    return TempGraph


def Gamilton_Cycle():
    N, M, Graph_G, Graph_H, firststr = Fill_Graph()

    List_Path = list()
    New_Graph_H = [[0] * N for i in range(N)]
    New_Graph_G = [[0] * N for i in range(N)]
    LeftRandom = list(range(1, N + 1))
    random.shuffle(LeftRandom)

    print(f"\nN = {N}, M = {M}")

    # Находим все переходы из вершин
    Dict = All_Path(Graph_G)
    print("\nПереходы:")
    for i in range(len(Dict)):
        print(f"{i + 1}: {Dict[i]}")

    # Поиск Гамильтонова цикла
    Path = Search_Gamilton_Cycle(Dict, N, 1, List_Path)
    print(f"\nГамильтонов цикл: {Path}\n")
```

```
print("---------------------------------------")
print("Действия первого абонента - Алисы")
print("---------------------------------------")

NewList = list(range(N))
random.shuffle(NewList)
NewList_str = [i + 1 for i in NewList]
print(f"Алиса рандомит вершины графа: {NewList_str} \n")

k = 0
z = 0
for i in NewList:
    for j in NewList:
        # print(f"[{i}][{j}]: {Graph_G[int(i)][int(j)]}")
        New_Graph_H[k][z] = Graph_G[int(i)][int(j)]
        z += 1
    z = 0
    k += 1

print("Изоморфный граф: ")
print(firststr)
for i in range(len(New_Graph_H)):
    print(" " + str(i + 1), end="\t")
    for j in range(len(New_Graph_H[i])):
        print(New_Graph_H[i][j], end="\t")
    print("\n", end="")

print("\nПеред кодировкой алгоритмом RSA\n" +
      f"Припишем рандомное число из списка {LeftRandom}")
k = 0
z = 0
for i in LeftRandom:
    for j in LeftRandom:
        New_Graph_H[k][z] = int(str(j) + str(New_Graph_H[k][z]))
        z += 1
    z = 0
    k += 1

print("\nПодготовленная матрица до RSA:")
print(firststr)
for i in range(len(New_Graph_H)):
    print(" " + str(i + 1), end="\t")
    for j in range(len(New_Graph_H[i])):
        print(New_Graph_H[i][j], end="\t")
    print("\n", end="")

print("\nАлиса генерирует ключи;")
P = generate_prime(0, 10 ** 9)
# print("P = ", P)
Q = generate_prime(0, 10 ** 9)
# print("Q = ", Q)
N_encode = P * Q
print("Ключ N: ", N_encode)
Phi = (P - 1) * (Q - 1)
# print("Phi = ", Phi)

d = generate_coprime(Phi)

c = gcd_modified(d, Phi)[1]
```

```
    if c < 0:
        c += Phi
    print("Ключ c: ", c)

    Graph_F = RSA_Matrix_Encode(New_Graph_H, d, N_encode, N)

    print("\nМатрица после RSA, для Боба:")
    print(firststr)
    for i in range(len(Graph_F)):
        print(" " + str(i + 1), end="\t")
        for j in range(len(Graph_F[i])):
            print(Graph_F[i][j], end="\t")
        print("\n", end="")

    print()
    print("---------------------------------------")
    print("Действия Второго абонента - Боба")
    print("---------------------------------------")
    print("Боб получил матрицу F\n")
    print("Какой вопрос выберет Боб?")
    print("1.'Алиса, каков Гамильтонов цикл для графа H?'")
    print("2.'Алиса, действительно ли граф H изоморфен G?'")
    answer = int(input())

    if answer == 1:
        print("-'Алиса, покажи Гамильтонов цикл?'")
        print("Алиса отсылает Бобу Гамильтонов цикл")
        print(f"Гамильтонов цикл: {Path}")
        print("Боб проходит по Матрице H-штрих, преобразуя соответствующий элемент цикла")
        print("Если элементы Матрицы F равны этим преобразованным, то Боб пытается по данному циклу пройти
свой граф\n" +
              "Если ему это удается, то Все впорядке. Если нет, то возникла ошибка в алгоритме")

        Bob_Check = RSA_Matrix_Encode(New_Graph_H, d, N_encode, N)

        flag = False
        check = 0
        for i in range(len(Graph_F)):
            for j in range(len(Graph_F[i])):
                if Bob_Check[i][j] == Graph_F[i][j]:
                    check += 1
        if check == pow(N, 2):
            flag = True
        if flag:
            print(f"\nМатрицы идентичны, т.к. flag = {flag}")
            print(f"Гамильтонов цикл Алисы: {Path}")
            print(f"Гамильтонов цикл Боба: {Path}")
        else:
            print(f"Матрицы разные flag = {flag}")

    if answer == 2:
        print("-'Докажи изоморфизм, Алиса?'")
        print("\nАлиса отсылает Бобу матрицу, которая еще не преобразования в RSA\n" +
              "И рандом столбцов, который она использовала после получения Гамильтонова цикла")
        print("\nБоб проверяет матрицы: сравнивает матрицы F и H-штрих путем\n" +
              "повторного шифрования и сравнения матриц")

        Bob_Check = RSA_Matrix_Encode(New_Graph_H, d, N_encode, N)

        flag = False
```

```
    check = 0
    for i in range(len(Graph_F)):
        for j in range(len(Graph_F[i])):
            if Bob_Check[i][j] == Graph_F[i][j]:
                check += 1
    if check == pow(N, 2):
        flag = True
    if flag:
        print(f"\nМатрицы идентичны, т.к. flag = {flag}")
    else:
        print(f"\nМатрицы разные flag = {flag}")

    print("Далее Боб отбрасывает все разряды кроме единичного от каждого элемента матрицы и получает матрицу,\n" +
          "изоморфную стартовой\n")

    print(firststr)
    for i in range(len(New_Graph_H)):
        print(" " + str(i + 1), end="\t")
        for j in range(len(New_Graph_H[i])):
            Numstr = str(New_Graph_H[i][j])
            New_Graph_H[i][j] = int(Numstr[-1])
            print(New_Graph_H[i][j], end="\t")
        print("\n", end="")

    print("\nЗатем Боб переставляет столбцы в соответствии с полученной нумерацией от Алисы")
    print(f"\nРандом Алисы вершин графа: {NewList_str}")

    k = 0
    z = 0
    for i in NewList:
        for j in NewList:
            New_Graph_G[int(i)][int(j)] = New_Graph_H[k][z]
            z += 1
        z = 0
        k += 1

    print("\nЗатем Алиса проверяет граф H,преобразуя его по ряду Алисы, и исходный граф Алисы\n" +
          "Если они идентичны, то из графа H мы получили граф G")

    flag = False
    check = 0
    for i in range(len(New_Graph_G)):
        for j in range(len(New_Graph_G[i])):
            if Graph_H[i][j] == New_Graph_G[i][j]:
                check += 1
    if check == pow(N, 2):
        flag = True
    if flag:
        print()
        print(f"Матрицы идентичны, т.к. flag = {flag}")
    else:
        print(f"Матрицы разные flag = {flag}")

if __name__ == '__main__':
    Gamilton_Cycle()
```