Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"

профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

**Расчетно-графическая работа по дисциплине  
Криптографические методы защиты информации**

Вариант 2

Выполнил:

Cтудент гр. ИВ-821 Овчинников М.А.

ФИО студента

Проверил:

Старший преподаватель кафедры ВС Петухова Я. В.

ФИО преподавателя

Новосибирск 2021 г.

**Оглавление**

[Постановка задачи 3](#_Toc89929584)

[Теоретические сведения 4](#_Toc89929585)

[Результат работы программы](#_Toc89929587) 6

[Список литературы](#_Toc89929588) 9

[Листинг 1](#_Toc89929589)0

# Постановка задачи

Это задание выполняется по вариантам, в зависимости от номера студента в журнале. Для определения номера варианта необходимо взять номер студента в журнале, вычислить его остаток от деления на и прибавить .

, где - номер варианта - номер в журнале.

Необходимо написать программу, реализующую протокол доказательства с нулевым знанием для задачи «Гамильтонов цикл».

Рассматриваемая задача является NP-полной и не имеет быстрых методов для решения, поэтому для тестирования необходимо генерировать правильные решения при помощи дополнительно разработанных программ.

Необходимо информацию о графах считывать из файла. В файле описание графа будет определяться следующим образом:

1. в первой строке файла содержатся два числаи , количество вершин графа и количество ребер соответственно;
2. в последующих m строках содержится информация о ребрах графа, каждое из которых описывается с помощью двух чисел (номера вершин, соединяемых этим ребром);
3. указывается последовательность вершин, задающая гамильтонов цикл (этот пункт можно вынести в отдельный файл).

# 1 Теоретические сведения

Доказательства с нулевым знанием – это интерактивный протокол, позволяющий одной из сторон убедиться в достоверности какого-либо утверждения, не получив при этом никакой другой информации от второй стороны.

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

Рассмотрим протокол, в котором абонент будет доказывать абоненту , что он знает гамильтонов цикл в некотором графе так, чтобы абонент не получил никаких знаний об этом цикле (доказательство с нулевым разглашением). Пусть абонент знает гамильтонов цикл в графе из вершин. Он может доказывать это абоненту (и всем, кто имеет этот граф) с помощью описываемого ниже протокола.

Протокол доказательства состоит из следующих шагов:

1. Абонент случайно выбирает перестановку и применяет ее к номерам вершин графа , получив при этом граф . Понятно, что графы и изоморфны. Зная гамильтонов цикл в графе , абонент знает гамильтонов цикл и в графе . Граф передается проверяющему .
2. Абонент , получив граф , случайным образом выбирает и передает абоненту .
3. Если , то абонент доказывает абоненту изоморфизм графов и Если , то абонент предоставляет проверяющему гамильтонов цикл графа .
4. Проверяющий проверяет, что в случае предъявленная перестановка действительно переводит граф в граф , а в случае проверяет гамильтонов цикл графа .

Эти четыре шага повторяются независимо раз. Вероятность того, что абонент не обманывает проверяющего увеличивается с числом итераций .

# Результат работы программы

После запуска программы необходимо ввести нужное число итераций.



Рисунок 1. Запуск программы.

После ввода числа итераций в первую очередь генерируется граф и гамильтонов цикл

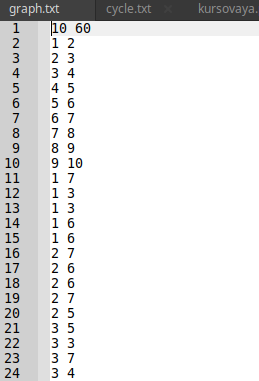


Рисунок 2. Сгенерированный граф.

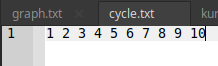


Рисунок 3. Сгенерированный гамильтонов цикл.

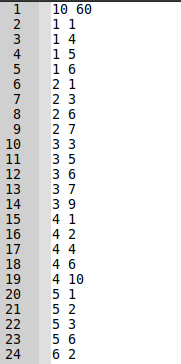


Рисунок 4. Изоморфный гамильтонов граф.

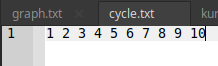


Рисунок 5. Гамильтонов цикл изоморфного графа.

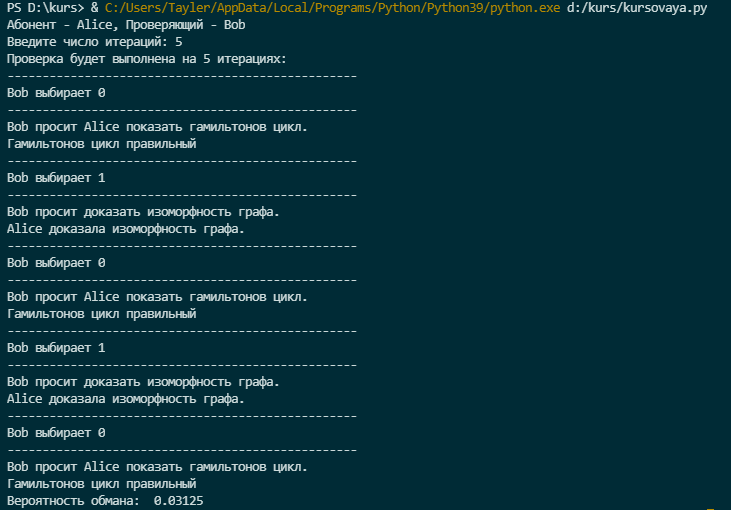


Рисунок 5. Результат работы программы.

Также была продемонстрирована работа программы с заведомо неправильным гамильтоновым циклом.

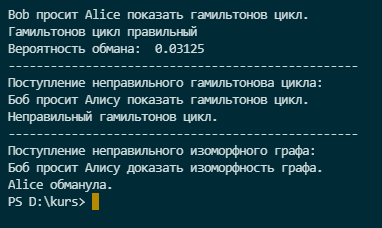


Рисунок 6. Результат работы при неправильном гамильтоновом цикле.

# Список литературы

1. Рябко Б. Я., Фионов А. Н. Р 98 Криптографические методы защиты информации: Учебное пособие для вузов. - 2-е издание, стереотип. - М.: Горячая линия-Телеком, 2012. - 229 c.
2. Саммерфилд М. Программирование на Python 3. Подробное руководство. - Пер. с англ. - СПб.: Символ‑Плюс, 2009. - 608 с.
3. Колисниченко Д.Н., Аллен Питер В. LINUX: полное руководство. - СПб: Наука и Техника, 2006. - 784 с.

# Листинг

#!/usr/bin/python3

import random as rnd

debug = False

# N <= 1000 M <= N^2

# N - количество вершин

# M - количество ребер

def check\_errors(n, m):

if n > 1000:

print("n должно быть <= 1000.")

exit()

if m > n\*\*2:

print("m должно быть <= n^2")

exit()

def check\_cycle(matrix, cycle, n):

if len(set(cycle)) != n:

print("Неправильный гамильтонов цикл.")

return False

for i in range(n - 1):

if cycle[i + 1] not in matrix[cycle[i] - 1]:

print(cycle[i + 1], matrix[cycle[i] - 1])

print("Неправильный гамильтонов цикл.")

return False

print("Гамильтонов цикл правильный")

return True

def get\_p\_matrix(matrix, cycle, n, cycle\_n=1000):

new\_index = [(i + 1) for i in range(n)]

rnd.shuffle(new\_index)

new\_index\_file = open("new\_index.txt", "w")

for i in range(n):

det = ''

if i != n:

det = ' '

new\_index\_file.write(str(new\_index[i]) + det)

new\_index\_file.close()

if debug:

print("---NEW INDEX---")

print([i for i in range(1, n + 1)])

print(new\_index)

print("------")

p\_matrix = [set() for i in range(n)]

p\_cycle = []

for i in range(n):

for v in range(1, n + 1):

if v in matrix[i]:

p\_matrix[new\_index[i] - 1].add(new\_index[v - 1])

if i < cycle\_n:

p\_cycle.append(new\_index[cycle[i] - 1])

return p\_matrix, p\_cycle

def bob\_check\_p\_matrix(matrix, p\_matrix, p\_index, n):

for i in range(n):

for v in range(1, n + 1):

if v in p\_matrix[i]:

if (p\_index.index(v) + 1) not in matrix[p\_index.index(i + 1)]:

print("Алиса обманула.")

return False

print("Алиса доказала изоморфность графа.")

return True

def matrix\_to\_file(filename, w\_matrix, n, m):

p\_graph\_file = open(filename, 'w')

p\_graph\_file.write(str(n) + " " + str(m) + "\n")

for i in range(n):

str\_v = ''

for v in w\_matrix[i]:

det = '\n'

if i == n:

det = ''

str\_v += str(i + 1) + " " + str(v) + det

p\_graph\_file.write(str\_v)

p\_graph\_file.close()

def cycle\_to\_file(filename, p\_cycle):

p\_cycle\_file = open(filename, 'w')

for v in p\_cycle:

p\_cycle\_file.write(str(v) + " ")

p\_cycle\_file.close()

def gen\_graph(n = 1000, m = 100000):

file = open("graph.txt", 'w')

cycle = open("cycle.txt", 'w')

file.write(str(n) + ' ' + str(m) + "\n")

count = (m // n) - 1

if count == 0:

count = 1

for i in range(n - 1):

file.write(str(i + 1) + ' ' + str(i + 2) + "\n")

if i != n - 1:

cycle.write(str(i + 1) + ' ')

for j in range(count):

v = rnd.randint(1, n - 1)

while v == i:

v = rnd.randint(1, n - 1)

file.write(str(i + 1) + ' ' + str(v) + "\n")

for j in range(count):

v = rnd.randint(1, n - 1)

while v == i:

v = rnd.randint(1, n - 1)

file.write(str(n) + ' ' + str(v) + "\n")

file.write(str(n) + ' ' + str(1))

cycle.write(str(n))

file.close()

cycle.close()

def file\_len(filename="graph.txt"):

with open(filename) as f:

return sum(1 for \_ in f)

def get\_matrix(filename = "graph.txt"):

file = open(filename, 'r')

n, m = file.readline().split(' ')

matrix = [set() for i in range(int(n))]

line = file.readline()

while line:

v1, v2 = line.split(' ')

matrix[int(v1) - 1].add(int(v2))

line = file.readline()

file.close()

return matrix, int(n), file\_len(filename) - 1

def get\_cycle(n = 1000, filename = "cycle.txt"):

cycle\_file = open(filename, 'r')

cycle = cycle\_file.readline().split(' ')

for i in range(n):

if cycle[i] == '':

break

cycle[i] = int(cycle[i])

cycle\_file.close()

return cycle

# START

if debug:

print("@@@@@ DEBUG MODE ON @@@@@")

gen\_graph()

cycle = get\_cycle()

matrix, n, m = get\_matrix()

check\_errors(n, m)

if debug:

print("---ORIGINAL MATRIX---")

print(matrix)

print("---CYCLE---")

print(cycle)

print("---N---")

print(n)

count = 5

print("Проверка Алисы на", count, "итерациях:")

for i in range(count):

p\_matrix, p\_cycle = get\_p\_matrix(matrix, cycle, n)

p\_matrix\_filename = "p\_graph.txt"

matrix\_to\_file(p\_matrix\_filename, p\_matrix, n, m)

p\_index\_filename = "new\_index.txt"

bob\_choise = rnd.randint(0, 1)

p\_index = get\_cycle(n, p\_index\_filename)

if bob\_choise:

print("Боб просит доказать изоморфность графа.")

if not bob\_check\_p\_matrix(matrix, p\_matrix, p\_index, n):

exit()

continue

print("Боб просит Алису показать гамильтонов цикл.")

p\_cycle\_filename = "p\_cycle.txt"

cycle\_to\_file(p\_cycle\_filename, p\_cycle)

check = check\_cycle(p\_matrix, p\_cycle, n)

if not check:

print("Неправильный путь.")

exit()

print("Вероятность, что Алиса обманула: ", 1 / (2 \*\* count))

wrong\_cycle\_n = 999

wrong\_cycle = get\_cycle(wrong\_cycle\_n, "wrong\_cycle.txt")

print("Поступление неправильного гамильтонова цикла:")

p\_matrix, p\_cycle = get\_p\_matrix(matrix, wrong\_cycle, n, wrong\_cycle\_n)

p\_matrix\_filename = "p\_graph.txt"

matrix\_to\_file(p\_matrix\_filename, p\_matrix, n, m)

print("Боб просит Алису показать гамильтонов цикл.")

p\_cycle\_filename = "p\_cycle.txt"

cycle\_to\_file(p\_cycle\_filename, p\_cycle)

check\_cycle(p\_matrix, p\_cycle, n)

print("Поступление неправильного изоморфного графа:")

p\_matrix\_filename = "wrong\_graph.txt"

p\_matrix, n, m = get\_matrix(p\_matrix\_filename)

p\_index\_filename = "new\_index.txt"

p\_index = get\_cycle(n, p\_index\_filename)

print("Боб просит Алису доказать изоморфность графа.")

bob\_check\_p\_matrix(matrix, p\_matrix, p\_index, n)