Правительство Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
(НИУ ВШЭ)

Московский институт электроники и математики им. А.Н. Тихонова

ОТЧЕТ
О ПРАКТИЧЕСКОЙ РАБОТЕ № _1_
по дисциплине «Математические основы защиты информации»
Подстановочные шифры

Студент гр. БИБ191
_____ И. Г. Тюрин
«___» _____ 2021 г.

Руководитель
Заведующий кафедрой информационной безопасности киберфизических систем
канд. техн. наук, доцент
_____О.О. Евсютин
«___» _____ 2021 г.

Москва 2021

СОДЕРЖАНИЕ

1 Задание на практическую работу

Целью работы является приобретение навыков программной реализации и криптоанализа применительно к простым подстановочным шифрам.

В рамках практической работы необходимо выполнить следующее:

1) написать программную реализацию следующих шифров:

- шифр простой замены:
- аффинный шифр;
- аффинный рекуррентный шифр;

2) изучить методы криптоанализа моноалфавитных подстановочных шифров с использование дополнительных источников;

3) провести криптоанализ данных шифров; 4) подготовить отчет о выполнении работы.

Программа должна обладать следующей функциональностью для каждого из реализованных в ней шифров:

1) принимать на вход произвольную последовательность символов, вводимую пользователем в качестве открытого текста или шифртекста;

2) принимать на вход секретный ключ вида, соответствующего конкретному шифру;

3) осуществлять зашифрование или расшифрование введенного текста по выбору пользователя.

2 Краткая теоретическая часть

2.1 Описание шифров

Шифр простой замены

Очередной шифр, относящийся к группе одноалфавитных шифров подстановки. Ключом шифра служит перемешанный произвольным образом алфавит. Например, ключом может быть следующая последовательность букв: XFQABOLYWJGPMRVIHUSDZKNTEC.

При шифровании каждая буква в тексте заменяется по следующему правилу. Первая буква алфавита замещается первой буквой ключа, вторая буква алфавита — второй буквой ключа и так далее. В нашем примере буква A будет заменена на X, буква B на F.

При расшифровке буква сперва ищется в ключе и затем заменяется буквой стоящей в алфавите на той же позиции.

Афинный рекуррентный шифр

Рассмотрим немного более интересный одноалфавитный шифр подстановки под названием аффиный шифр. Он тоже реализует простую подстановку, но обеспечивает немного большее пространство ключей по сравнению с шифром Цезаря. В аффинном шифре каждой букве алфавита размера m ставится в соответствие число из диапазона 0… m-1. Затем при помощи специальной формулы, вычисляется новое число, которое заменит старое в шифртексте.

Процесс шифрования можно описать следующей формулой:

$$E(x) = (ax + b) mod\ m$$

Где x — номер шифруемой буквы в алфавите; m — размер алфавита; a, b — ключ шифрования.

Для расшифровки вычисляется другая функция:

$$D(x) = a^{-1}(x - b) mod\ m$$

Где a-1 — число обратное a по модулю m. Это значит, что для корректной расшифровки число a должно быть взаимно простым с m.

С учетом этого ограничения вычислим пространство ключей аффиного шифра на примере английского алфавита. Так как английский алфавит содержит 26 букв, то в качестве a может быть выбрано только взаимно простое с 26 число. Таких чисел всего двенадцать: 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23 и 25. Число b в свою очередь может принимать любое значение в интервале от 0 до 25, что в итоге дает нам 12*26 = 312 вариантов возможных ключей.

2.2 Методы криптоанализа шифров

Криптоанализ шифра простой замены

Пространство ключей шифра простой замены огромно и равно количеству перестановок используемого алфавита. Так для английского языка это число составляет 26! = 288. Разумеется, наивный перебор всех возможных ключей дело безнадежное и для взлома потребуется более утонченная техника, такая как *поиск восхождением к вершине*:

1) Выбирается случайная последовательность букв — основной ключ. Шифртекст расшифровывается с помощью основного ключа. Для получившегося текста вычисляется коэффициент, характеризующий вероятность принадлежности к естественному языку.

2) Основной ключ подвергается небольшим изменениям (перестановка двух произвольно выбранных букв). Производится расшифровка и вычисляется коэффициент полученного текста.

3) Если коэффициент выше сохраненного значения, то основной ключ заменяется на модифицированный вариант.

4) Шаги 2-3 повторяются пока коэффициент не станет постоянным.

Для вычисления коэффициента используется еще одна характеристика естественного языка — *частота встречаемости триграмм.*

Чем ближе текст к английскому языку, тем чаще в нем будут встречаться такие триграммы как THE, AND, ING. Суммируя частоты появления в естественном языке всех триграмм, встреченных в тексте получим коэффициент, который с большой долей вероятности определит текст, написанный на естественном языке.

Криптоанализ аффиного шифра

Очевидно, что и в случае аффиного шифра простейшим способом взлома оказывается перебор всех возможных ключей. Но в результате перебора получится 312 различных текстов. Проанализировать такое количество сообщений можно и вручную, но лучше автоматизировать этот процесс, используя такую характеристику как частота появления букв.

Давно известно, что буквы в естественных языках распределены не равномерно. К примеру, частоты появления букв английского языка в текстах имеют следующие значения:

Буква	E	T	A	O	I	N	S	H	R	D	L	C	U	M	W	F	G	Y	P	B	V	K	X	J	Q	Z
Частота, %	12,7	9,06	8,17	7,51	6,97	6,75	6,33	6,09	5,99	4,25	4,03	2,78	2,76	2,41	2,36	2,23	2,02	1,97	1,93	1,49	0,98	0,77	0,15	0,15	0,1	0,05

Т.е. в английском тексте наиболее встречающимися буквами будут E, T, A. В то время как самыми редкими буквами являются J, Q, Z. Посчитав частоту появления каждой буквы в тексте, мы можем определить насколько частотная характеристика текста соответствует английскому языку.

Для этого необходимо вычислить значение:

$$x = \sum_{i=1}^{c} n_i f_i$$

Где ni — частота i-й буквы алфавита в естественном языке. И fi — частота i-й буквы в шифртексте. Чем больше значение x, тем больше вероятность того, что текст написан на естественном языке.

Таким образом, для взлома аффиного шифра достаточно перебрать 312 возможных ключей и вычислить значение χ для полученного в результате расшифровки текста. Текст, для которого значение χ окажется максимальным, с большой долей вероятности и является зашифрованным сообщением

Разумеется, следует учитывать, что метод не всегда работает с короткими сообщениями, в которых частотные характеристики могут сильно отличатся от характеристик естественного языка.

Криптоанализ афинного рекуррентного шифра

Аффинный рекуррентный шифр уже не получится взломать при помощи частотного анализа, в отличие от аффинного. Это невозможно, потому что одни и те же символы могут шифроваться разными буквами и наоборот. Это происходит из-за смены ключей. Однако, имея ключ, но не имея информации о нем, возможно получение исходного текста путем многократного повторного шифрования. Приведу пример: была зашифрована строка. У нас имеется шифртекст. После многократного повторения шифрования (на 11 итерации) шифртекст превратился в читаемую строку, что и было начальной строкой.

3 Примеры шифрования

Примеры «ручного» шифрования (зашифрование и расшифрование) с необходимыми пояснениями в части выбора параметров шифров.

Построение шифра простой замены без программы:

1. Шифрование
   Ключ - 2 4 6 14 5 13 10 22 19 8 21 1 16 20 11 12 24 23 3 26 15 7 17 25 18 9

исходный текст	индекс исходной буквы в алфавите	новый индекс с учетом ключа	буква, соответствующая новому индексу
a	1	2	b
f	6	13	m
f	6	13	m

i	9	19	s
n	14	20	t
e	5	5	e

2. Расшифрование
   Ключ - 2 4 6 14 5 13 10 22 19 8 21 1 16 20 11 12 24 23 3 26 15 7 17 25 18 9

шифртекст	индекс буквы шифртекста в алфавите	новый индекс с учетом ключа	буква, соответствующая новому индексу
b	2	1	a
m	13	6	f
m	13	6	f
s	19	9	i
t	20	14	n
e	5	5	e

3. Шифрование
   Ключ - 2 4 6 14 5 13 10 22 19 8 21 1 16 20 11 12 24 23 3 26 15 7 17 25 18 9

исходный текст	индекс исходной буквы в алфавите	новый индекс с учетом ключа	буква, соответствующая новому индексу
c	3	6	f
i	9	19	s
p	16	12	l
h	8	22	v
e	5	5	3
r	18	23	w

4. Расшифрование
   Ключ - 2 4 6 14 5 13 10 22 19 8 21 1 16 20 11 12 24 23 3 26 15 7 17 25 18 9

шифртекст	индекс буквы шифртекста в алфавите	новый индекс с учетом ключа	буква, соответствующая новому индексу
f	6	3	c
s	19	9	i
l	12	16	p
v	22	8	h
3	5	5	e
w	23	18	r

Построение Аффинного шифра без программы:

1. Шифрование
   Alpha – 7, Beta – 3.

Новый индекс = (исходный индекс * alpha + beta) mod мощность алфавита

исходный текст	индекс исходной буквы в алфавите	новый индекс с учетом ключа	буква, соответствующая новому индексу
c	3	24	x
i	9	14	n
p	16	11	k
h	8	7	g
e	5	12	l
r	18	25	y

2. Расшифрование
   Alpha – 7, Beta – 3.

Новый индекс = (исходный индекс - beta) * alpha-1 mod мощность алфавита

шифртекст	индекс исходной буквы в алфавите	новый индекс с учетом ключа	буква, соответствующая новому индексу
x	24	3	c
n	14	9	i
k	11	16	p
g	7	8	h
l	12	5	e
y	25	18	r

3. Шифрование
   Alpha – 3, Beta – 1.

Новый индекс = (исходный индекс * alpha + beta) mod мощность алфавита

исходный текст	индекс исходной буквы в алфавите	новый индекс с учетом ключа	буква, соответствующая новому индексу
a	1	4	d
f	6	19	s
f	6	19	s
i	9	2	b
n	14	17	q
e	5	16	p

4. Расшифрование
   Alpha – 3, Beta – 1.

Новый индекс = (исходный индекс - beta) * alpha-1 mod мощность алфавита

шифртекст	индекс исходной буквы в алфавите	новый индекс с учетом ключа	буква, соответствующая новому индексу
d	4	1	a
s	19	6	f
s	19	6	f
b	2	9	i
q	17	14	n
p	16	5	e

Построение Аффинного рекуррентного шифра без программы:

1. Шифрование
   Alpha1 – 7, Beta1 – 3, Alpha2 – 9, Beta2 – 22.

Новый индекс = (исходный индекс * alpha + beta) mod мощность алфавита

исходный текст	индекс исходной буквы в алфавите	текущий ключ	новый индекс с учетом ключа	буква, соответствующая новому индексу
c	3	7 3	24	x
i	9	9 22	25	y
p	16	11 3	11	k
h	8	21 25	23	w
e	5	23 2	13	m
r	18	14 1	19	s

2. Расшифрование
   Alpha1 – 7, Beta1 – 3, Alpha2 – 9, Beta2 – 22.

Новый индекс = (исходный индекс - beta) * alpha-1 mod мощность алфавита

исходный текст	индекс исходной буквы в алфавите	текущий ключ	новый индекс с учетом ключа	буква, соответствующая новому индексу
x	24	7 3	3	c
y	25	9 22	9	i
k	11	11 3	16	p
w	23	21 25	8	h
m	13	23 2	5	e
s	19	14 1	18	r

3. Шифрование
   Alpha1 – 7, Beta1 – 3, Alpha2 – 9, Beta2 – 22.

Новый индекс = (исходный индекс * alpha + beta) mod мощность алфавита

исходный текст	индекс исходной буквы в алфавите	текущий ключ	новый индекс с учетом ключа	буква, соответствующая новому индексу
a	1	7 3	10	j
f	6	9 22	24	x
f	6	11 3	17	q
i	9	21 25	6	f
n	14	23 2	12	l
e	5	14 1	19	s

4. Расшифрование
   Alpha1 – 7, Beta1 – 3, Alpha2 – 9, Beta2 – 22.

Новый индекс = (исходный индекс - beta) * alpha-1 mod мощность алфавита

шифртекст	индекс исходной буквы в алфавите	текущий ключ	новый индекс с учетом ключа	буква, соответствующая новому индексу
j	10	7 3	1	a
x	24	9 22	6	f
q	17	11 3	6	f
f	6	21 25	9	i
l	12	23 2	14	n
s	19	14 1	5	e

4 Программная реализация шифров

Программный код указан в приложении А.

Результаты работы Алгоритма, реализующего шифр простой замены:

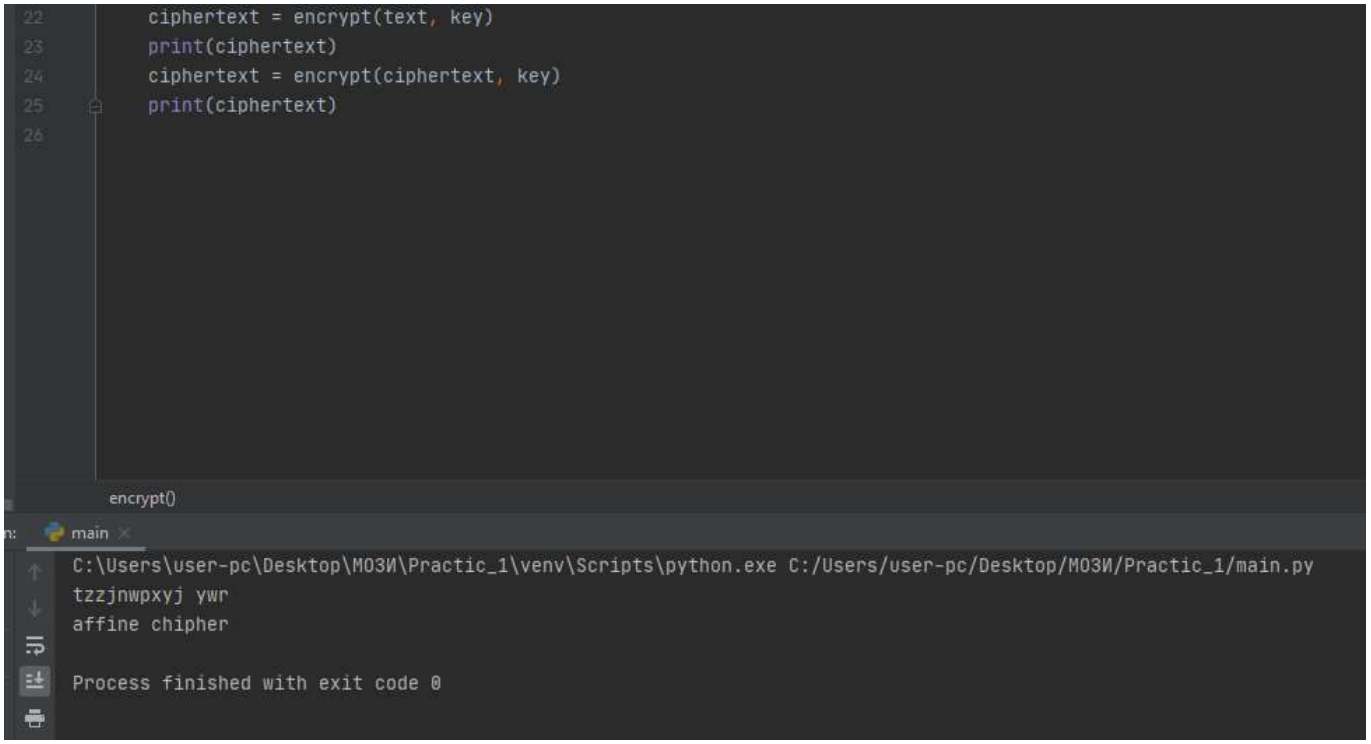

Рисунок 4.1 – Результаты работы Алгоритма, реализующего шифр простой замены

Результаты работы Алгоритма, реализующего шифр простой замены:

Рисунок 4.2 – Результаты работы Алгоритма, реализующего шифр простой замены

Результаты работы Алгоритма, реализующего аффинный шифр:

Рисунок 4.3 – Результаты работы Алгоритма, реализующего аффинный шифр

Результаты работы Алгоритма, реализующего аффинный шифр:

Рисунок 4.4 – Результаты работы Алгоритма, реализующего аффинный шифр

Результаты работы Алгоритма, реализующего аффинный рекуррентный шифр:

Рисунок 4.5 – Результаты работы Алгоритма, реализующего аффинный рекуррентный шифр

Результаты работы Алгоритма, реализующего аффинный рекуррентный шифр:

Рисунок 4.6 – Результаты работы Алгоритма, реализующего аффинный рекуррентный шифр

5 Примеры криптоанализа

Добрались до интересной части. Криптоанализ шифра можно, как обычно, разделить на две части: получение первичного ключа с помощью частотного анализа символов, и нахождение финального ключа с помощью частотного анализа биграмм. Получается, нам нужны идеальные частоты символов и биграмм английского текста. Где их добыть, решайте сами. Частоты одиночных символов я выгуглил, а частоты биграмм собрал самостоятельно по очень большому количеству литературных текстов. Так или иначе, с этими частотами криптоанализ работает.

*Первый этап***.** Выполняем частотный анализ символов шифротекста. После этого, выстраиваем символы алфавита в порядке убывания их частот в шифротексте, это и есть первичный ключ. Теперь выполняем расшифровку этим ключом, и получаем первичный открытый текст. На этом первый этап завершен.

Второй этап. В первичном открытом тексте считаем рейтинг биграмм. Теперь нужно провернуть хитрую штуковину. А именно, поменять местами, в первичном ключе, самый частый символ со вторым по частоте. После этого, расшифровать текст новым ключом, и снова посчитать рейтинг биграмм, и, *внимание*, если он улучшился(т.е. стал меньше по значению, чем предыдущий, до замены), то *оставить* в ключе эту замену, и перейти к следующей паре. Если же рейтинг не изменился или ухудшился, **отменить** перестановку символов, и поменять следующую пару. Смысл сего действия в том, чтобы *перебрать все возможные перестановки* на ключе, *которые улучшают рейтинг биграмм*. Финальным ключом к шифротексту станет такой ключ, любые перестановки в котором *уже никогда не улучшат* рейтинг биграмм.

Замечание 1. Для того, чтобы не расшифровывать текст по новой после каждой перестановки на ключе, мы просто меняем в словаре частоты соответствующих биграмм(строки 80-88).

Замечание 2. Вы можете выбрать другой алгоритм обхода ключа для выполнения перестановок на нем, нет предела совершенству.

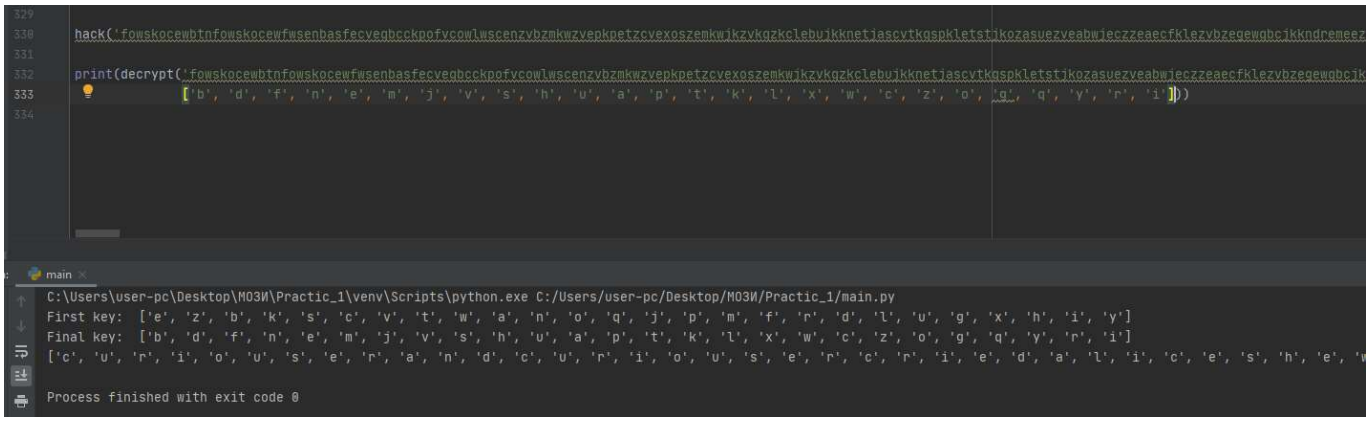

Рисунок 5.1 – Результаты работы Алгоритма, реализующего взлом подстановочных шифров

Текст, взятый для анализа:
‘Curiouser and curiouser!’ cried Alice (she was so much surprised, that for the moment she quit

e forgot how to speak good English); ‘now I’m opening out like the largest telescope that ever w as!...

6 Выводы о проделанной работе

В результате выполнения работы был реализован алгоритм аффинного и рекуррентного аффинного шифра на языке Python, а также было произведено ознакомлении с криптоанализом и получены навыки работы с данными шифрами.

7 Список использованных источников

1. Афинный рекуррентный шифр, Краткая теория, Реализация шифра, Криптоанализ - Аффинный и аффинный рекуррентный шифр // Афинный рекуррентный шифр, Краткая теория, Реализация шифра, Криптоанализ - Аффинный и аффинный рекуррентный шифр URL: https://studbooks.net/1786378/informatika/affinnyy_rekurrentnyy_shifr (дата обращения: 28.01.2021).

2. Классический криптоанализ // Классический криптоанализ URL: https://habr.com/ru/post/271257/ (дата обращения: 28.01.2021).

3. Affine cipher implementation using python // Affine cipher implementation using python URL: https://gist.github.com/mengelbart/3e170dad4b805cd71b29ae8572779a60 (дата обращения: 28.01.2021).

4. Mathematical cryptography algorithms implemented in Python URL: https://github.com/aschwenn/cryptomath (дата обращения: 28.01.2021)

5. Sweigart A.l. «Взламываем секретные шифры с Python» (Hacking Secret Ciphers with Python). Электронное издание, 2020.

6. Шифрование и криптография в Python // Шифрование и криптография в Python URL: https://python-scripts.com/encryption-cryptography (дата обращения: 27.01.2021).

ПРИЛОЖЕНИЕ А.

Шифр простой замены

```
alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
'o', 'p', 'q', 'r', 's', 't', 'u',
            'v', 'w', 'x', 'y', 'z', ' ']

''' Функция шифрования по ключу
    text - list символов текста
    key - list с перестановкой на алфавите'''


def encrypt(__text__, __key__):
    result = ''
    for i in range(len(__text__)):
        result += (__key__[alphabet.index(__text__[i])])

    return result


# Press the green button in the gutter to run the script.
if __name__ == 'replacement':
    text = 'my text'
    key = ['t', 'g', 'x', 'v', 'w', 'z', 'b', 'y', 'j', 'i', 'l', 'k', 'o', 'n', 'm',
' ', 'q', 'r', 's', 'a', 'u',
           'd', 'e', 'c', 'h', 'f', 'p']
    ciphertext = encrypt(text, key)
    print(ciphertext)
    ciphertext = encrypt(ciphertext, key)
    print(ciphertext)
```

Аффинный шифр

```
class Affine(object):
    DIE = 128
    KEY = (7, 3, 55)

    def __init__(self):
        pass

    def encryptChar(self, char):
        K1, K2, KI = self.KEY
        return chr((K1 * ord(char) + K2) % self.DIE)

    def encrypt(self, string):
        return "".join(map(self.encryptChar, string))

    def decryptChar(self, char):
        K1, K2, KI = self.KEY
        return chr(KI * (ord(char) - K2) % self.DIE)

    def decrypt(self, string):
        return "".join(map(self.decryptChar, string))


affine = Affine()
print(affine.encrypt('Affine Cipher'))
print(affine.decrypt('JMMbFcXb[F!'))
```

Расширенный алгоритм Евклида

```
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
```

```
        g, x, y = egcd(b % a, a)
        return (g, y - (b // a) * x, x)


# x = modinv(b) mod n, (x * b) % n == 1
def modinv(b, n):
    g, x, _ = egcd(b, n)
    #print(g, x, _)
    if g == 1:
        return x % n

print(modinv(7, 128))
```

Аффинный рекуррентный шифр

```
class Affine(object):
    DIE = 128
    KEY1 = (7, 3, 55)
    KEY2 = (9, 22, 57)

    def __init__(self):
        pass

    def encryptChar(self, char, iter):
        return chr((iter[0] * ord(char) + iter[1]) % self.DIE)

    def encrypt(self, string):
        dirDictA, dirDictB, invDictA = Affine.calcMas(string)
        res = ''
        for i in range(len(string)):
            iter = [dirDictA[i], dirDictB[i], invDictA[i]]
            res += self.encryptChar(string[i], iter)
        return res

    def decryptChar(self, char, iter):
        return chr(iter[2] * (ord(char) - iter[1]) % self.DIE)

    def decrypt(self, string):
        dirDictA, dirDictB, invDictA = Affine.calcMas(string)
        res = ''
        for i in range(len(string)):
            iter = [dirDictA[i], dirDictB[i], invDictA[i]]
            res += self.decryptChar(string[i], iter)
        return res

    @classmethod
    def calcMas(cls, string):
        K1, K2, KI1 = cls.KEY1
        K3, K4, KI3 = cls.KEY2
        dirDictA = [K1, K3]
        dirDictB = [K2, K4]
        invDictA = [KI1, KI3]
        for i in range(2, len(string)):
            dirDictA.append(dirDictA[i - 1] * dirDictA[i - 2])
            dirDictB.append(dirDictB[i - 1] + dirDictB[i - 2])
            invDictA.append(invDictA[i - 1] * invDictA[i - 2])
        return dirDictA, dirDictB, invDictA


affine = Affine()

print(affine.encrypt('Affine Cipher'))
print(affine.decrypt('J,3>&Bq;H'))
```

```
# Эталонные частоты символов английского алфавита
symbolsFreqsStandart = [8.17, 1.49, 2.78, 4.25, 12.7, 2.23, 2.02, 6.09, 6.97, 0.15,
0.77, 4.03, 2.41, 6.75, 7.51, 1.93,
                        0.1, 5.99, 6.33, 9.06, 2.76, 0.98, 2.36, 0.15, 1.97, 0.05]
# Эталонные частоты биграмм английского алфавита
bigrammsFreqsStandart = {'oc': 0.1333384, 'um': 0.0728041, 'vz': 0.0, 'ox':
0.0139953, 'jr': 0.0, 'bh': 0.0004793,
                         'ys': 0.0889562, 'pc': 0.0005751, 'ha': 1.7533862, 'ez':
0.002684, 'az': 0.0172544,
                         'xi': 0.0111675, 'bc': 0.0001438, 'qx': 0.0, 'hd':
0.0015817, 'kl': 0.0156249, 'yd': 0.0030195,
                         'xw': 0.0, 'ew': 0.1074089, 'id': 0.4131479, 'wl': 0.018932,
'mz': 0.0, 'wi': 0.5180165,
                         'qo': 0.0, 'pf': 0.0013899, 'hn': 0.0092503, 'on':
1.4637513, 'hx': 0.0, 'gd': 0.0012462,
                         'co': 0.6017964, 'gg': 0.0393018, 'cc': 0.0585213, 'hh':
0.0009107, 'pd': 0.0011024, 'wq': 0.0,
                         'bw': 0.0001438, 'ta': 0.4067254, 'kk': 0.0001438, 'qn':
0.0, 'vr': 0.0004793, 'pe': 0.3927781,
                         'ly': 0.436681, 'wv': 0.0, 'lq': 0.0, 'ep': 0.1284976, 'xt':
0.0313456, 'aa': 0.0001917,
                         'wc': 0.0011024, 'zi': 0.0061828, 'wa': 0.7742928, 'zn':
0.0, 'uj': 9.59e-05, 'kn': 0.1178095,
                         'gf': 0.0003834, 're': 1.7554471, 'ex': 0.1552899, 'ho':
0.7850289, 'xq': 0.0002876,
                         'fw': 0.0012941, 'ht': 0.2339893, 'wz': 0.0, 'hf':
0.0028757, 'yx': 4.79e-05, 'ia': 0.0999799,
                         'ob': 0.0785556, 'fq': 0.0, 'oh': 0.0224308, 'to':
1.1336164, 'qf': 0.0, 'gj': 0.0,
                         'ud': 0.0639373, 'iz': 0.0226704, 'ft': 0.0989734, 'by':
0.1159882, 'lv': 0.0322562,
                         'aw': 0.1056355, 'pi': 0.0983024, 'sj': 0.0005751, 'oi':
0.0861284, 'iv': 0.1765225, 'pv': 0.0,
                         'me': 1.0261117, 'as': 1.1904123, 'bs': 0.0404521, 'km':
0.0024923, 'bj': 0.0115509, 'vt': 0.0,
                         'nj': 0.0082917, 'gv': 9.59e-05, 'oy': 0.0270319, 'uk':
0.0023485, 'vd': 0.0, 'ss': 0.3641165,
                         'ke': 0.3602822, 'cn': 0.0001438, 'yg': 0.0008627, 'vs':
9.59e-05, 'ul': 0.4280059,
                         'lm': 0.159316, 'zw': 0.0, 'kh': 0.0035947, 'fh': 0.0010544,
'ei': 0.1148858, 'lw': 0.0184527,
                         'jg': 0.0, 'hq': 0.0, 'mn': 0.0075728, 'xr': 0.0002396,
'ci': 0.1234171, 'em': 0.282781,
                         'ge': 0.3587964, 'jm': 0.0, 'vp': 0.0, 'ir': 0.3141266,
'cb': 9.59e-05, 'of': 0.8852484,
                         'ib': 0.0658065, 'fv': 0.0, 'ua': 0.0644166, 'ds':
0.1273953, 'yn': 0.0037385, 'yh': 0.0043136,
                         'am': 0.3072248, 'qr': 0.0, 'zo': 0.0012941, 'iw':
0.0002876, 'xl': 0.0004314, 'ar': 0.9913153,
                         'ww': 0.0005272, 'wm': 0.0005272, 'rf': 0.0232935, 'uh':
4.79e-05, 'sg': 0.0048888,
                         'rc': 0.0551663, 'rk': 0.1053479, 'vq': 0.0, 'zv':
0.0001438, 'xx': 0.0, 'do': 0.3965645,
                         'es': 1.0640235, 'fd': 0.0009586, 'hj': 4.79e-05, 'wg':
4.79e-05, 'bf': 4.79e-05,
                         'lp': 0.0271757, 'ji': 0.0011024, 'zd': 0.0, 'mc':
0.0219994, 'pq': 0.0, 'nu': 0.0396852,
                         'vf': 0.0, 'ql': 0.0, 'tc': 0.0425609, 'gk': 0.0, 'qq': 0.0,
'ba': 0.1554337, 'kp': 0.0003834,
                         'dq': 0.0004793, 'wr': 0.0346047, 'dn': 0.0160083, 'yk':
0.0, 'ag': 0.1888881, 'wo': 0.2980224,
```

```
                        'xa': 0.0302432, 'mg': 0.0001438, 'ot': 0.4721485, 'nr':
0.0122219, 'tq': 0.0, 'bn': 0.0007669,
                        'jf': 0.0, 'qc': 0.0, 'xs': 0.0006231, 'fr': 0.2388781,
'ev': 0.265239, 'qw': 0.0,
                        'di': 0.2823017, 'ov': 0.1414864, 'si': 0.4520183, 'jo':
0.0370491, 'mo': 0.3124491,
                        'wn': 0.1516473, 'ax': 0.0052722, 'ey': 0.1752284, 'ui':
0.0821024, 'ut': 0.5242952,
                        'hs': 0.0106882, 'bd': 0.0014858, 'jt': 0.0, 'vw': 0.0,
'sk': 0.0684426, 'ej': 0.0031154,
                        'ks': 0.0362343, 'uy': 0.0014379, 'lu': 0.0876621, 'os':
0.2433834, 'kd': 0.0003355,
                        'ro': 0.6651585, 'rv': 0.0612532, 'jb': 0.0, 'cl':
0.1495864, 'fp': 0.0023006, 'mt': 0.000671,
                        'ne': 0.757997, 'xf': 0.0010544, 'dr': 0.1308941, 'qj': 0.0,
'fm': 0.0004314, 'ug': 0.1809799,
                        'la': 0.4516828, 'gm': 0.004026, 'hz': 0.0, 'pb': 0.0013899,
'xy': 0.0006231, 'ko': 0.0017734,
                        'ms': 0.0702639, 'yt': 0.026984, 'ij': 4.79e-05, 'jl': 0.0,
'wp': 0.0004793, 'lj': 4.79e-05,
                        'ns': 0.3018088, 'jy': 0.0, 'zc': 0.0, 'cp': 0.0014379,
'ip': 0.0550704, 'fe': 0.2252662,
                        'fi': 0.2207609, 'oa': 0.0706953, 'kt': 0.0008148, 'yo':
0.6294035, 'tp': 0.0025402,
                        'hl': 0.0068538, 'cy': 0.0223349, 'lc': 0.0059911, 'pp':
0.1200142, 'db': 0.004745,
                        'te': 0.9009212, 'vk': 0.0, 'ux': 0.0025882, 'tv':
0.0002396, 'gp': 0.0030195, 'gw': 0.0007189,
                        'kb': 0.0016775, 'or': 1.0262555, 'dv': 0.0247314, 'xc':
0.0284219, 'uq': 0.0001917, 'mx': 0.0,
                        'ym': 0.0129888, 'hg': 0.0001438, 'iy': 0.0, 'xd': 0.0,
'ie': 0.2681148, 'ee': 0.5119775,
                        'va': 0.0685385, 'tl': 0.1617124, 'ts': 0.250381, 'mj':
4.79e-05, 'yr': 0.0028757,
                        'xp': 0.053345, 'uf': 0.0159604, 'bk': 0.0, 'nx': 0.0043615,
'ae': 0.0016775, 'dx': 0.0,
                        'cg': 0.0046012, 'dh': 0.0055118, 'mu': 0.1292166, 'kg':
0.0009107, 'jv': 0.0, 'go': 0.1641089,
                        'gt': 0.0115988, 'rg': 0.0594799, 'xz': 0.0, 'bg': 0.0,
'be': 0.6146893, 'jz': 0.0, 'qm': 0.0,
                        'ai': 0.490697, 'pz': 0.0, 'uc': 0.1287373, 'us': 0.4861917,
'dj': 0.0007669, 'nc': 0.2857047,
                        'th': 3.9283557, 'ju': 0.050613, 'qa': 0.0, 'zj': 0.0, 'jd':
0.0, 'qy': 0.0, 'ur': 0.6653502,
                        'et': 0.4489029, 'dk': 0.0014858, 'er': 2.1172151, 'rz':
0.0002876, 'ih': 0.0003834,
                        'tb': 0.0034509, 'ef': 0.1511201, 'fj': 4.79e-05, 'ja':
0.0190278, 'ix': 0.0167272,
                        'rx': 9.59e-05, 'tn': 0.0088189, 'tw': 0.0874704, 'in':
2.2636382, 'hr': 0.0903941,
                        'pl': 0.1756118, 'zh': 4.79e-05, 'lh': 0.0015337, 'zg': 0.0,
'vv': 0.0, 'sm': 0.0639852,
                        'mh': 0.0003355, 'yw': 0.0032112, 'vu': 0.0025402, 'bm':
0.002684, 'hb': 0.0065183,
                        'dl': 0.0628349, 'vj': 0.0, 'kq': 0.0, 'ct': 0.2344207,
'ue': 0.0961935, 'jk': 0.0,
                        'ig': 0.2644722, 'sa': 0.3689094, 'np': 0.0053201, 'fn':
0.0001917, 'cd': 0.0021089, 'xk': 0.0,
                        'vh': 0.0, 'lr': 0.0161041, 'uu': 0.0, 'sd': 0.004026, 'oq':
0.0004314, 'cs': 0.0035467,
                        'hk': 0.0001917, 'oj': 0.0012941, 'kc': 0.0014379, 'pr':
0.2487035, 'qi': 0.0, 'eg': 0.0598633,
                        'aj': 0.0039781, 'ao': 0.006087, 'sv': 0.0004793, 'ad':
```

```
0.5948467, 'pm': 0.0030675,
                        'bu': 0.2697443, 'eb': 0.0212325, 'td': 0.0005751, 'zu':
0.0002396, 'ol': 0.4180366,
                        'xh': 0.0030195, 'sq': 0.0101609, 'nk': 0.0962893, 'xe':
0.0104964, 'bp': 0.0, 'zy': 0.0015337,
                        'tx': 0.0, 'cw': 0.0, 'rm': 0.087758, 'sy': 0.0212805, 'ls':
0.0814793, 'qh': 0.0,
                        'ek': 0.0205136, 'eq': 0.0119822, 'om': 0.5926419, 'zq':
0.0, 'dd': 0.0545911, 'lk': 0.0383432,
                        'po': 0.3729834, 'ng': 1.0051188, 'dt': 0.002684, 'jq': 0.0,
'av': 0.3843426, 'ac': 0.3905733,
                        'su': 0.240939, 'xv': 0.0, 'kr': 0.0004314, 'pk': 0.0038822,
'og': 0.0469704, 'ye': 0.1527018,
                        'ka': 0.0132763, 'ck': 0.2258414, 'vb': 0.0, 'mq': 0.0,
'eu': 0.012174, 'ik': 0.0484083,
                        'rw': 0.0217598, 'cj': 0.0, 'na': 0.175468, 'oz': 0.0027799,
'rj': 0.0004314, 'du': 0.0549745,
                        'wj': 0.0, 'da': 0.1481485, 'wu': 0.0013899, 'cu':
0.1149337, 'nn': 0.0748172, 'iq': 0.0022047,
                        'nb': 0.0047929, 'tz': 0.0010065, 'yl': 0.0072852, 'aq':
0.0004793, 'dy': 0.0791787,
                        'xu': 0.0018692, 'pt': 0.0508047, 'ce': 0.5314366, 'ry':
0.2762627, 'fx': 0.0, 'tu': 0.1738864,
                        'dm': 0.012126, 'uz': 0.004074, 'ca': 0.4243633, 'is':
1.214904, 'tm': 0.016871,
                        'rp': 0.037145, 'ph': 0.0321604, 'im': 0.3939763, 'rq':
0.0004793, 'gq': 0.0, 'tj': 0.0008148,
                        'nz': 0.0014379, 'zf': 0.0, 'cf': 0.0012462, 'sn':
0.0138994, 'nv': 0.0359947, 'sw': 0.0508047,
                        'cm': 0.0100651, 'le': 0.8587437, 'ow': 0.5455277, 'sh':
0.4811591, 'if': 0.1947355,
                        'rs': 0.3518946, 'hy': 0.0433757, 'fk': 9.59e-05, 'dp':
0.0012941, 'ab': 0.2164952, 'jn': 0.0,
                        'wk': 0.0009586, 'ru': 0.1182408, 'mb': 0.0555976, 'wh':
0.6630976, 'vo': 0.0497024, 'qe': 0.0,
                        'fl': 0.051284, 'mm': 0.0417941, 'ed': 1.3248052, 'nm':
0.0046491, 'xn': 0.0, 'nq': 0.010832,
                        'at': 1.5224164, 'ky': 0.0069018, 'au': 0.0806645, 'lg':
0.0022047, 'gi': 0.1091822,
                        'mi': 0.2701757, 'ec': 0.2608296, 'jw': 0.0, 'ap':
0.1791106, 'lb': 0.0041219, 'fy': 0.0044574,
                        'gs': 0.0416982, 'xm': 0.0001438, 'gb': 0.0011503, 'lo':
0.4294917, 'zx': 0.0, 'qu': 0.1142627,
                        'hm': 0.0100651, 'pw': 0.0016296, 'ah': 0.0156728, 'yu':
0.0, 'zm': 0.0, 'px': 0.0,
                        'ri': 0.543323, 'll': 0.7411739, 'sc': 0.1055876, 'sl':
0.054016, 'jp': 0.0002876, 'vn': 0.0,
                        'vm': 0.0, 'ws': 0.0362822, 'uw': 0.0, 'bi': 0.0537763,
'it': 1.2222851, 'ff': 0.1097574,
                        'gh': 0.3432195, 'sz': 0.0, 'vy': 0.008771, 'zz': 0.0056077,
'bb': 0.0178296, 'dg': 0.0395414,
                        'jc': 0.0, 'bl': 0.2468343, 'up': 0.2493745, 'fz': 0.0,
'fs': 0.0049367, 'tr': 0.3306142,
                        'oo': 0.4246988, 'kf': 0.0055118, 'qb': 0.0, 'pu':
0.0705515, 'an': 2.0177146, 'ok': 0.1354953,
                        'rh': 0.0187402, 'se': 0.8834271, 'hv': 0.0, 'ic':
0.4850414, 'lf': 0.0901544, 'mp': 0.1502574,
                        'ii': 0.0, 'zb': 0.0, 'dw': 0.0088669, 'ze': 0.0351319,
'eh': 0.0273675, 'ak': 0.1435952,
                        'yp': 0.0070456, 'hi': 1.2958561, 'za': 0.003355, 'fa':
0.2160639, 'zr': 0.0005751,
                        'gc': 0.0022527, 'bv': 0.0065663, 'he': 3.6071357, 'op':
0.1350639, 'lz': 4.79e-05,
```

```
                        'nh': 0.0082438, 'kx': 0.0, 'hp': 0.0002876, 'dz': 0.0,
'yj': 0.0001438, 'nt': 0.7810987,
                        'ub': 0.060726, 'yc': 0.0098734, 'lx': 0.0, 'ki': 0.095187,
'mk': 0.0, 'af': 0.0785556,
                        'jj': 0.0, 'ch': 0.5389135, 'ya': 0.0155769, 'hc':
0.0014379, 'qk': 0.0, 'qg': 0.0,
                        'kw': 0.0032592, 'gl': 0.0934615, 'ea': 0.7690206, 'iu':
0.0064225, 'py': 0.010113, 'gx': 0.0,
                        'il': 0.3936408, 'cx': 0.0, 'ay': 0.3080875, 'vc': 0.0,
'yz': 0.0007189, 'un': 0.3875538,
                        'br': 0.1393775, 'ga': 0.1517432, 'ty': 0.1117704, 'de':
0.6216869, 'bo': 0.1853414,
                        'fg': 0.0009107, 'bt': 0.0278467, 'nd': 1.4753501, 'mv':
0.0, 'gz': 0.0002396, 'ny': 0.1018491,
                        'bz': 0.0, 'wt': 0.0004793, 'zl': 0.0046491, 'je':
0.0247793, 'rt': 0.2767899, 'yv': 0.000671,
                        'vl': 0.0, 'so': 0.495442, 'ml': 0.004745, 'cr': 0.1618562,
've': 1.0398673, 'st': 1.0042082,
                        'qd': 0.0, 'qt': 0.0, 'yb': 0.0048408, 'cz': 0.0, 'pg':
4.79e-05, 'gr': 0.1434035,
                        'fo': 0.4685538, 'pj': 0.0, 'ni': 0.2512917, 'we':
0.5938881, 'ti': 0.6167982, 'li': 0.4602621,
                        'js': 0.0, 'ma': 0.5654183, 'xj': 0.0, 'gy': 0.0086751,
'cq': 0.0044095, 'uv': 0.0039302,
                        'pn': 0.0007189, 'yf': 0.0028278, 'fb': 0.0003834, 'fc':
0.0021089, 'rr': 0.1646361, 'zs': 0.0,
                        'df': 0.0115509, 'tt': 0.2426165, 'vi': 0.1706751, 'zk':
0.0, 'my': 0.2925106, 'xo': 0.0014858,
                        'el': 0.5384822, 'bq': 0.0, 'zt': 0.0, 'jx': 0.0, 'hu':
0.085026, 'gu': 0.0667172,
                        'en': 1.2625933, 'yq': 0.0, 'gn': 0.0320166, 'mr':
0.0999319, 'nw': 0.0081, 'no': 0.6392289,
                        'wd': 0.0053201, 'wx': 0.0, 'ln': 0.0033071, 'od':
0.1563444, 'pa': 0.234181, 'ra': 0.4437745,
                        'yi': 0.0260734, 'kz': 0.0, 'qp': 0.0, 'vg': 0.0, 'sf':
0.0103047, 'sx': 0.0, 'zp': 0.0,
                        'qs': 0.0, 'mf': 0.0036426, 'yy': 0.0, 'md': 0.0004314,
'cv': 0.0, 'rn': 0.1667929,
                        'sr': 0.0016296, 'hw': 0.0023006, 'dc': 0.0017734, 'wy':
0.0023485, 'ou': 1.7362276,
                        'oe': 0.0247314, 'rb': 0.0148101, 'vx': 0.0, 'al':
0.6719644, 'sp': 0.1451769, 'nf': 0.0441905,
                        'ku': 0.0012462, 'uo': 0.0038822, 'ld': 0.4214875, 'xb':
0.0002396, 'kj': 0.0, 'tf': 0.0070456,
                        'jh': 0.0001438, 'mw': 0.0001917, 'tk': 0.0001917, 'qz':
0.0, 'bx': 0.0, 'ps': 0.0514757,
                        'sb': 0.0118864, 'fu': 0.078316, 'nl': 0.099884, 'wf':
0.0014379, 'io': 0.3505526,
                        'eo': 0.0325438, 'kv': 0.0, 'rd': 0.2515793, 'qv': 0.0,
'lt': 0.0698805, 'tg': 0.0015817,
                        'xg': 0.0, 'rl': 0.101945, 'wb': 0.0011024}

# Словарь биграмм с частотами
bigramms = {'bf': 0, 'yh': 0, 'fq': 0, 'dg': 0, 'cv': 0, 'kp': 0, 'pk': 0, 'dl': 0,
'zv': 0, 'cx': 0, 'ry': 0, 'gv': 0,
            'yw': 0, 'ar': 0, 'jv': 0, 'mr': 0, 'fd': 0, 'bz': 0, 'xf': 0, 'ny': 0,
'pa': 0, 'po': 0, 'lf': 0, 'll': 0,
            'nb': 0, 'jq': 0, 'io': 0, 'ao': 0, 'zc': 0, 'xu': 0, 'th': 0, 'cu': 0,
'cc': 0, 'fy': 0, 'bi': 0, 'fc': 0,
            'ad': 0, 'rv': 0, 'ms': 0, 'pt': 0, 'xd': 0, 'hq': 0, 'un': 0, 'gh': 0,
'qa': 0, 'fx': 0, 'kr': 0, 'zk': 0,
            'nm': 0, 'fv': 0, 'vt': 0, 'wy': 0, 'zs': 0, 'ku': 0, 'eh': 0, 'or': 0,
'ae': 0, 'vc': 0, 'bh': 0, 'xx': 0,
```

```
            'ff': 0, 'dx': 0, 'bc': 0, 'ls': 0, 'ai': 0, 'tc': 0, 'ze': 0, 'eq': 0,
'aa': 0, 'si': 0, 'fn': 0, 'bv': 0,
            'ro': 0, 'yc': 0, 'tx': 0, 'dj': 0, 'hu': 0, 'dp': 0, 'cp': 0, 'sy': 0,
'zr': 0, 'vf': 0, 'dk': 0, 'tk': 0,
            'zi': 0, 'hi': 0, 'rm': 0, 'ac': 0, 'ax': 0, 'vm': 0, 'fe': 0, 'uk': 0,
'ah': 0, 'wa': 0, 'ak': 0, 're': 0,
            'nr': 0, 'uf': 0, 'hj': 0, 'pw': 0, 'ni': 0, 'ul': 0, 'kx': 0, 'ej': 0,
'sn': 0, 'pc': 0, 'jh': 0, 'fb': 0,
            'dw': 0, 'qw': 0, 'aq': 0, 'ba': 0, 'nf': 0, 'jx': 0, 'ha': 0, 'oo': 0,
'tg': 0, 'uu': 0, 'rl': 0, 'ot': 0,
            'os': 0, 'nx': 0, 'ko': 0, 'ig': 0, 'lh': 0, 'is': 0, 'ta': 0, 'sg': 0,
'hl': 0, 'fm': 0, 'wl': 0, 'ik': 0,
            'oc': 0, 'qh': 0, 'uy': 0, 'bt': 0, 'yo': 0, 'er': 0, 'px': 0, 'bw': 0,
'qz': 0, 'rz': 0, 'ml': 0, 'hn': 0,
            'nu': 0, 'ss': 0, 'ca': 0, 'zf': 0, 'xq': 0, 'gm': 0, 'sq': 0, 'py': 0,
'ne': 0, 'fl': 0, 'dr': 0, 'kh': 0,
            'es': 0, 'gd': 0, 'bx': 0, 'mv': 0, 'xo': 0, 'dh': 0, 'so': 0, 'ty': 0,
'uj': 0, 'mf': 0, 'ki': 0, 'se': 0,
            'tz': 0, 'nw': 0, 'jo': 0, 'mh': 0, 'dt': 0, 'mi': 0, 'pq': 0, 'lj': 0,
'ri': 0, 'qf': 0, 'km': 0, 'hd': 0,
            'iv': 0, 'kt': 0, 'gx': 0, 'zu': 0, 'in': 0, 'fk': 0, 'mk': 0, 'kc': 0,
'rc': 0, 'ew': 0, 'ux': 0, 'sl': 0,
            'vy': 0, 'ti': 0, 'kq': 0, 'xj': 0, 'xp': 0, 'uc': 0, 'qu': 0, 'rg': 0,
'aw': 0, 'ym': 0, 'cr': 0, 'di': 0,
            'rj': 0, 'gu': 0, 'pn': 0, 'ru': 0, 'gs': 0, 'wx': 0, 'sk': 0, 'pd': 0,
'xz': 0, 'ud': 0, 'ev': 0, 'lo': 0,
            'ov': 0, 'xs': 0, 'be': 0, 'yu': 0, 'yt': 0, 'ec': 0, 'bn': 0, 'js': 0,
'gr': 0, 'vb': 0, 'hh': 0, 'el': 0,
            'mp': 0, 'wn': 0, 'kl': 0, 'qj': 0, 'ja': 0, 'om': 0, 'us': 0, 'cg': 0,
'cj': 0, 'ye': 0, 'wk': 0, 'oq': 0,
            'dc': 0, 'dm': 0, 'up': 0, 'fs': 0, 'yq': 0, 'sf': 0, 'gn': 0, 'if': 0,
'ou': 0, 'qn': 0, 'lx': 0, 'rq': 0,
            'fu': 0, 'rk': 0, 'tr': 0, 'nq': 0, 'ur': 0, 'pm': 0, 'rw': 0, 'cn': 0,
'ge': 0, 'um': 0, 'gq': 0, 'tn': 0,
            'va': 0, 'qt': 0, 'wf': 0, 'wu': 0, 'le': 0, 'tm': 0, 'rr': 0, 'iu': 0,
'vz': 0, 'ei': 0, 'id': 0, 'mu': 0,
            'vo': 0, 'zg': 0, 'tp': 0, 'mb': 0, 'cy': 0, 'zx': 0, 've': 0, 'he': 0,
'lc': 0, 'qi': 0, 'ol': 0, 'ub': 0,
            'zz': 0, 'ip': 0, 'sv': 0, 'oj': 0, 'bu': 0, 'vw': 0, 'ef': 0, 'fh': 0,
'sm': 0, 'al': 0, 'uw': 0, 'xb': 0,
            'hp': 0, 'af': 0, 'pf': 0, 'wc': 0, 'zo': 0, 'ts': 0, 'qy': 0, 'nc': 0,
'lk': 0, 'dv': 0, 'vs': 0, 'jr': 0,
            'hx': 0, 'co': 0, 'ij': 0, 'pi': 0, 'pu': 0, 'ix': 0, 'jg': 0, 'zt': 0,
'hm': 0, 'hz': 0, 'gl': 0, 'jw': 0,
            'jz': 0, 'ui': 0, 'mz': 0, 'ma': 0, 'dy': 0, 'zj': 0, 'yl': 0, 'za': 0,
'wj': 0, 'dq': 0, 'hr': 0, 'xn': 0,
            'ay': 0, 'zh': 0, 'sr': 0, 'lg': 0, 'vk': 0, 'yk': 0, 'tb': 0, 'kb': 0,
'hs': 0, 'rd': 0, 'gk': 0, 'vi': 0,
            'kk': 0, 'fp': 0, 'qe': 0, 'kv': 0, 'hw': 0, 'dz': 0, 'yj': 0, 'su': 0,
'tv': 0, 'wr': 0, 'gz': 0, 'wd': 0,
            'wq': 0, 'do': 0, 'yg': 0, 'nn': 0, 'ct': 0, 'pp': 0, 'iw': 0, 'hy': 0,
'ra': 0, 'xm': 0, 'bq': 0, 'pj': 0,
            'mn': 0, 'bs': 0, 'vd': 0, 'qo': 0, 'wb': 0, 'tt': 0, 'rp': 0, 'sb': 0,
'gb': 0, 'ph': 0, 'wt': 0, 'ww': 0,
            'wg': 0, 'dn': 0, 'nk': 0, 'yz': 0, 'we': 0, 'df': 0, 'ek': 0, 'ag': 0,
'on': 0, 'zq': 0, 'sj': 0, 'pe': 0,
            'bd': 0, 'lp': 0, 'qv': 0, 'db': 0, 'jf': 0, 'md': 0, 'uv': 0, 'sc': 0,
'xa': 0, 'my': 0, 'ga': 0, 'jl': 0,
            'as': 0, 'gp': 0, 'vx': 0, 'gw': 0, 'mx': 0, 'nz': 0, 'ep': 0, 'ug': 0,
'fa': 0, 'xg': 0, 'tu': 0, 'xw': 0,
            'kz': 0, 'mo': 0, 'cm': 0, 'mc': 0, 'jj': 0, 'nt': 0, 'em': 0, 'rn': 0,
'ie': 0, 'mg': 0, 'jk': 0, 'xr': 0,
            'wv': 0, 'jt': 0, 'ap': 0, 'en': 0, 'wz': 0, 'bp': 0, 'ke': 0, 'de': 0,
```

```
'zw': 0, 'ez': 0, 'qs': 0, 'eg': 0,
            'na': 0, 'st': 0, 'oh': 0, 'oi': 0, 'ut': 0, 'ck': 0, 'ho': 0, 'yr': 0,
'wo': 0, 'yy': 0, 'ht': 0, 'eu': 0,
            'tq': 0, 'yx': 0, 'ds': 0, 'op': 0, 'ly': 0, 'sd': 0, 'jb': 0, 'jd': 0,
'gt': 0, 'pg': 0, 'ey': 0, 'pl': 0,
            'vq': 0, 'xl': 0, 'nd': 0, 'cb': 0, 'xy': 0, 'sw': 0, 'ee': 0, 'rx': 0,
'np': 0, 'wh': 0, 'fj': 0, 'xc': 0,
            'bj': 0, 'iq': 0, 'ia': 0, 'yb': 0, 'cq': 0, 'eb': 0, 'hv': 0, 'nl': 0,
'ir': 0, 'yf': 0, 'eo': 0, 'fr': 0,
            'qd': 0, 'nv': 0, 'jp': 0, 'xe': 0, 'ys': 0, 'jc': 0, 'zy': 0, 'fi': 0,
'tw': 0, 'qc': 0, 'lv': 0, 'gy': 0,
            'du': 0, 'vn': 0, 'cd': 0, 'vg': 0, 'nh': 0, 'nj': 0, 'at': 0, 'av': 0,
'hg': 0, 'ql': 0, 'gf': 0, 'da': 0,
            'kf': 0, 'kd': 0, 'sh': 0, 'pz': 0, 'ox': 0, 'kw': 0, 'il': 0, 'vl': 0,
'gg': 0, 'qr': 0, 'uo': 0, 'iz': 0,
            'tl': 0, 'ex': 0, 'gj': 0, 'qx': 0, 'ab': 0, 'oa': 0, 'mq': 0, 'qq': 0,
'ws': 0, 'lm': 0, 'ob': 0, 'te': 0,
            'pb': 0, 'ow': 0, 'xi': 0, 'xt': 0, 'mt': 0, 'uh': 0, 'oy': 0, 'an': 0,
'xv': 0, 'et': 0, 'to': 0, 'hf': 0,
            'ld': 0, 'ci': 0, 'ns': 0, 'hc': 0, 'lr': 0, 'bo': 0, 'az': 0, 'ue': 0,
'ih': 0, 'bb': 0, 'sz': 0, 'mm': 0,
            'rh': 0, 'lu': 0, 'tf': 0, 'ic': 0, 'oe': 0, 'je': 0, 'ln': 0, 'yp': 0,
'vv': 0, 'it': 0, 'cw': 0, 'iy': 0,
            'ib': 0, 'me': 0, 'ft': 0, 'vj': 0, 'lb': 0, 'zl': 0, 'zp': 0, 'dd': 0,
'uz': 0, 'ya': 0, 'br': 0, 'sa': 0,
            'lw': 0, 'ng': 0, 'rt': 0, 'ji': 0, 'rb': 0, 'of': 0, 'vu': 0, 'bm': 0,
'rf': 0, 'bl': 0, 'qk': 0, 'la': 0,
            'ce': 0, 'aj': 0, 'am': 0, 'fz': 0, 'rs': 0, 'ea': 0, 'yn': 0, 'fg': 0,
'oz': 0, 'ua': 0, 'sx': 0, 'gc': 0,
            'ju': 0, 'zb': 0, 'yi': 0, 'bg': 0, 'wm': 0, 'im': 0, 'au': 0, 'qb': 0,
'fw': 0, 'ka': 0, 'td': 0, 'tj': 0,
            'ky': 0, 'lq': 0, 'kg': 0, 'jy': 0, 'hb': 0, 'cf': 0, 'by': 0, 'lz': 0,
'fo': 0, 'ks': 0, 'ch': 0, 'ok': 0,
            'jm': 0, 'vp': 0, 'og': 0, 'hk': 0, 'zn': 0, 'cl': 0, 'mw': 0, 'od': 0,
'kj': 0, 'wp': 0, 'sp': 0, 'qg': 0,
            'ps': 0, 'pr': 0, 'qp': 0, 'vr': 0, 'gi': 0, 'pv': 0, 'kn': 0, 'bk': 0,
'vh': 0, 'cz': 0, 'zm': 0, 'no': 0,
            'cs': 0, 'xh': 0, 'go': 0, 'yv': 0, 'yd': 0, 'ii': 0, 'li': 0, 'mj': 0,
'jn': 0, 'qm': 0, 'xk': 0, 'ed': 0,
            'zd': 0, 'wi': 0, 'uq': 0, 'lt': 0}
alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
'o', 'p', 'q', 'r', 's', 't', 'u',
            'v', 'w', 'x', 'y', 'z']


# Функция шифрования по ключу
# text - list символов текста
# key - list с перестановкой на алфавите
def encrypt(text, key):
    result = []
    for i in range(len(text)):
        result.append(key[alphabet.index(text[i])])

    return result


# Функция дешифрования по ключу
# code - list символов шифра
# key - list с перестановкой на алфавите
def decrypt(code, key):
    result = []
    for i in range(len(code)):
        result.append(alphabet[key.index(code[i])])
```

```
    return result


# Функция взламывает шифр замены
# code - list символов шифра
def hack(code):
    # С помощью частотного анализа найти первый ключ
    # ПЕРВЫЙ ЭТАП. Частотный анализ символов шифротекста
    freqs = [0 for i in range(26)]
    for i in range(len(code)):
        freqs[alphabet.index(code[i])] += 1

    for i in range(len(freqs)):
        freqs[i] = (freqs[i] / len(code)) * 100

    # Отсортировать полученные частоты
    freqsSorted = sorted(freqs)

    # Задача: выстроить символы алфавита по частотам в шифре
    key = []
    for i in range(len(freqsSorted)):
        symb = alphabet[freqs.index(freqsSorted[i])]
        if not (symb in key):
            key.append(symb)
        # Если попались символы с одинаковыми частотами
        else:
            # Взять другой(следующий) символ с такой частотой
            curFreq = freqsSorted[i]
            for j in range(len(freqs)):
                if freqs[j] == curFreq:
                    s = alphabet[j]
                    if not (s in key):
                        key.append(s)
                        break

    # Развернуть ключ, чтобы получилось в порядке убывания частот
    key = list(reversed(key))
    # Первичный ключ
    print('First key: ', key)

    # Выполняем дешифровку этим ключом
    text = decrypt(code, key)

    # ВТОРОЙ ЭТАП. Нахождение ключа частотным анализом биграмм
    # Посчитать частоту биграмм в полученном тексте
    for i in range(1, len(text)):
        bigramms[text[i - 1] + text[i]] += 1

    for k, v in bigramms.items():
        bigramms[k] = (bigramms[k] / (len(text) - 1)) * 100

    # Посчитать рейтинг биграмм для этого ключа
    bigrammsRating = 0
    for k, v in bigramms.items():
        bigrammsRating += (bigramms[k] - bigrammsFreqsStandart[k]) ** 2

    # Выполняем перестановки на ключе до тех пор, пока рейтинг улучшается
    prevBigrammsRating = bigrammsRating
    # Величина, через которую брать символы(соседи, через один, через два и так
далее)
    step = 1
    while True:
```

```
        findGoodBigrammFlag = False
        for i in range(0, 26):
            if i + step >= 26:
                break

            # Меняем местами два символа в ключе
            key[i], key[i + step] = key[i + step], key[i]

            # Копируем частоты биграмм
            copyBigrammsFreq = {k: v for k, v in bigramms.items()}

            # Меняем частоты биграмм в словаре, чтобы не расшифровывать текст по
новой
            for sym in alphabet:
                if sym == alphabet[i] or sym == alphabet[i + step]:
                    continue

                bigramms[sym + alphabet[i]], bigramms[sym + alphabet[i + step]] =
bigramms[sym + alphabet[i + step]], \

bigramms[sym + alphabet[i]]
                bigramms[alphabet[i] + sym], bigramms[alphabet[i + step] + sym] =
bigramms[alphabet[i + step] + sym], \

bigramms[alphabet[i] + sym]

            bigramms[alphabet[i] + alphabet[i + step]], bigramms[alphabet[i + step] +
alphabet[i]] = bigramms[alphabet[

i + step] +

alphabet[

i]], \

bigramms[

alphabet[i] +

alphabet[

i + step]]
            bigramms[alphabet[i] + alphabet[i]], bigramms[alphabet[i + step] +
alphabet[i + step]] = bigramms[alphabet[

i + step] +

alphabet[

i + step]], \

bigramms[

alphabet[i] +

alphabet[i]]

            # Считаем новый рейтинг биграмм
            bigrammsRating = 0
            for k, v in bigramms.items():
                bigrammsRating += (bigramms[k] - bigrammsFreqsStandart[k]) ** 2

                # Если рейтинг улучшился - оставляем перестановку в ключе
```

```
            # и ставим step в 1
            if bigrammsRating < prevBigrammsRating:
                findGoodBigrammFlag = True
                prevBigrammsRating = bigrammsRating
                step = 1
                break
            # Иначе - меняем символы обратно, возвращаем частоты
            else:
                key[i], key[i + step] = key[i + step], key[i]
                for k, v in copyBigrammsFreq.items():
                    bigramms[k] = v
        # Если за весь проход улучшающая перестановка не нашлась,
        # увеличиваем расстояние и идем еще раз
        if findGoodBigrammFlag == False:
            step += 1
            if step >= 25:
                break

    print('Final key: ', key)

    return key
```