МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Цепные дроби и квадратные сравнения

ОТЧЁТ
ПО ДИСЦИПЛИНЕ
«ТЕОРЕТИКО-ЧИСЛОВЫЕ МЕТОДЫ В КРИПТОГРАФИИ»

студента 5 курса 531 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Мызникова Сергея Анатольевича

Преподаватель, профессор ___________ В.А. Молчанов

подпись, дата

Саратов 2024

СОДЕРЖАНИЕ

1 Постановка задачи

Цель работы — изучение основных свойств цепных дробей квадратных сравнений.

Порядок выполнения работы:

1. Разобрать алгоритм разложения чисел в цепную дробь и привести их программную реализацию.
2. Рассмотреть алгоритмы приложений цепных дробей и привести их программную реализацию.
3. Разобрать алгоритмы вычисления символов Лежандра и Якоби и привести их программную реализацию.
4. Рассмотреть алгоритмы извлечения квадратного корня в кольце вычетов.

2 Теоретические сведения

2.1 Алгоритм разложения чисел в цепную дробь

С помощью алгоритма Евклида любое рациональное число можно представить в виде специальной конструкции, которая называется цепной дробью.

Рассмотрим рациональное число r, представленное в виде несократимой дроби $r = \frac{a_0}{a_1}$. Так как НОД$(a_0, a_1) = 1$, то результат вычисления этого наибольшего общего делителя по алгоритму Евклида имеет вид:

$$a_0 = a_1 q_1 + a_2, 0 \leq a_2 < a_1,$$

$$a_1 = a_2 q_2 + a_3, 0 \leq a_3 < a_2,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..$$

$$a_{k-2} = a_{k-1} q_{k-1} + a_k, 0 \leq a_k < a_{k-1},$$

$$a_{k-1} = a_k q_k, a_k = \text{НОД}(a_0, a_1) = 1$$

Эти равенства можно переписать в виде:

$$\frac{a_0}{a_1} = q_1 + \frac{a_2}{a_1}; \ \frac{a_1}{a_2} = q_2 + \frac{a_3}{a_2}, \ldots, \frac{a_{k-2}}{a_{k-1}} = q_{k-1} + \frac{a_k}{a_{k-1}} = q_{k-1} + \frac{1}{q_k}$$

Тогда рациональное число r можно представить следующим образом:

$$r = \frac{a_0}{a_1} = q_1 + \frac{a_2}{a_1} = q_1 + \frac{1}{\frac{a_1}{a_2}} = q_1 + \frac{1}{q_2 + \frac{a_3}{a_2}} = q_1 + \cfrac{1}{q_2 + \cfrac{1}{\ldots + \cfrac{1}{q_{k-1} + \cfrac{1}{q_k}}}}$$

где q_1– целое число и $q_2, \ldots, q_k$ – целые положительные числа.

Определение. Выражение вида

$$q_1 + \cfrac{1}{q_2 + \cfrac{1}{\dots + \cfrac{1}{q_{k-1} + \cfrac{1}{q_k}}}}$$

называют цепной (или непрерывной) дробью с неполными частными q_1, q_2, …, q_k и обозначать символом (q_1; q_2, ..., q_k).

Определение. Для цепной дроби $\frac{a_0}{a_1}$ = (q_1; q_2, ..., q_k) выражения

$$\delta_1 = q_1, \delta_2 = q_1 + \frac{1}{q_2}, \delta_3 = q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3}}, \dots, \delta_k = q_1 + \cfrac{1}{q_2 + \cfrac{1}{\dots + \cfrac{1}{q_{k-1} + \cfrac{1}{q_k}}}}$$

называются подходящими дробями конечной цепной дроби (q_1; q_2, ..., q_k) и обозначаются символами δ_i = (q_1; q_2, ..., q_i) где $1 \le i \le k$.

Аналогично определяются подходящие дроби δ_i = (q_1; q_2, ..., q_i) для бесконечной цепной дроби (q_1; q_2, ..., q_k, ...).

Каждая подходящая дробь δ_i = (q_1; q_2, ..., q_i) является несократимой рациональной дробью $\delta_i = \frac{P_i}{Q_i}$ с числителем P_i и знаменателем Q_i, которые вычисляются по следующим рекуррентным формулам:

$$P_i = q_i P_{i-1} + P_{i-2}, Q_i = q_i Q_{i-1} + Q_{i-2}$$

с начальными условиями $P_{-1} = 0, P_0 = 1, Q_{-1} = 1, Q_0 = 0$

Важные свойства подходящих дробей:

1. Числители и знаменатели двух последовательных подходящих дробей удовлетворяют равенству $P_i Q_{i-1} - P_{i-1} Q_i = (-1)^i$ для всех $i = 1, \dots, k$
2. P_i, O_i взаимно просты

3. $\frac{P_i}{Q_i}$-$\frac{P_{i-1}}{Q_{i-1}} = \frac{(-1)^i}{Q_i Q_{i-1}}$

4. $\delta_{2n} > \delta_{2n-1}$
5. $P_i Q_{i-2} - P_{i-2} Q_i = q_i (-1)^{i-1}$
6. $\frac{P_i}{Q_i}$ - $\frac{P_{i-1}}{Q_{i-1}} = \frac{q_i(-1)^{i-1}}{Q_i Q_{i-2}}$
7. $\delta_{2n} = \frac{P_{2n}}{Q_{2n}}$ – убывающая последовательность
8. $\delta_{2n-1} = \frac{P_{2n-1}}{Q_{2n-1}}$ – возрастающая последовательность

$$\delta_1 < \delta_3 < \cdots < \delta_{2n-1} \ldots \leq \frac{a}{b} \leq \cdots \delta_{2n} \ldots . < \delta_4 < \delta_2$$

9. $\frac{P_n}{Q_n} = q_1 + \sum_{i=1}^{n} \frac{(-1)^i}{Q_i Q_{i-1}}$ сходится по признаку Лейбница
10. $Q_n = q_n Q_{n-1} + Q_{n-2} \geq Q_{n-1} + Q_{n-2} \geq 2Q_{n-2} \geq 2^{\frac{n-2}{2}}$
11. $|a - \delta_i| \leq |\delta_i - \delta_{i-1}| = \frac{1}{Q_i Q_{i-1}}$
12. Если все $q_i > 0$, то $\lim_{n \to \infty} \delta_n = a$, в частности, любое число $a \in R_+$ представляется цепной дробью.

Вычисление числителей и знаменателей подходящих дробей с учетом

$$P_i = q_i P_{i-1} + P_{i-2}, Q_i = q_i Q_{i-1} + Q_{i-2}$$

оформляется в виде следующей таблицы:

i	-1	0	1	2	3	…	k-1	k
q_i			q_1	q_2	q_3		q_{k-1}	q_k
P_i	0	1	P_1	P_2	P_3		P_{k-1}	P_k
Q_i	1	0	Q_1	Q_2	Q_3		Q_{k-1}	Q_k

2.2 Решение Диофантовых уравнений

Определение: Диофантовыми уравнениями называются алгебраические уравнения с целочисленными коэффициентами, решение которых отыскивается в целых числах.

Например, диофантовым уравнением является уравнение вида

$$ax - by = 1$$

с целыми неотрицательными коэффициентами a, b.

Предположим, что $\frac{P_k}{Q_k}$ – последняя подходящая дробь в представлении непрерывной дробью рационального числа $\frac{a}{b}$, где НОД(a, b) = 1. Тогда $a = P_k$, $b = Q_k$. Перепишем уравнение $P_k Q_{k-1} - P_{k-1} Q_k = (-1)^{k-1}$ для соседних подходящих дробей: $a(-1)^{k-1} Q_{k-1} - b(-1)^{k-1} P_{k-1} = 1$. Получаем одно решение диофантового уравнения $ax - by = 1$: $x_0 = (-1)^{k-1} Q_{k-1}$, $y_0 = (-1)^{k-1} P_{k-1}$.

Остальные имеют вид:

$$x = (-1)^{k-1} Q_{k-1} + b$$

$$y = (-1)^{k-1} P_{k-1} + a$$

В общем случае диофантово уравнение разрешимо, если число с делится на НОД(a, b):

$$x = (-1)^{k-1} \frac{c}{\text{НОД}(a, b)} Q_{k-1} + b$$

$$y = (-1)^{k-1} \frac{c}{\text{НОД}(a, b)} P_{k-1} + a$$

2.3 Решение линейных сравнений

Аналогично диофантовым уравнениям решаются уравнения первой степени вида: $ax \equiv b (mod\, m)$. Для решения нужно взять обе части диофантового уравнения в виде: $ax - my = b$ по модулю m. Это сравнение будет разрешимо тогда, когда b делится на НОД(a, m). Решение будет иметь вид:

$$x = (-1)^{k-1} \frac{b}{\text{НОД}(a, m)} Q_{k-1} \ (mod\ m)$$

2.4 Поиск обратного элемента в кольце вычетов Z_m

Обратным к числу a по модулю m называется такое число b, что: ab ≡ 1 (mod m), и его нередко обозначают как a^{-1}.

Понятно, что для нуля обратного элемента не существует никогда; для остальных же элементов обратный может как существовать, так и нет. Утверждается, что обратный существует только для тех элементов, которые взаимно просты с модулем m.

Для решения задачи по поиску обратного элемента в кольце необходимо решить следующее линейное сравнение: ab ≡ 1 (mod m). Для этого можно воспользоваться диофантовыми уравнениями и получить выражение следующего вида: ax + my = 1.

2.5 Символ Лежандра

Число $a \in Z_p$ называется квадратичным вычетом по модулю p, если

$$(\exists x \in Z)\ x^2 \equiv a \ (mod\ p).$$

В противном случае число a называется квадратичным невычетом по модулю p.

Для нечетного простого числа p символом Лежандра числа $a \in Z$ называется выражение:

$$\left(\frac{a}{p}\right) = \begin{cases} 1, a - \text{вычет} \\ 0, a \equiv 0 \ (mod\ p) \\ -1, \ \ a - \text{невычет} \end{cases}$$

1. $a \equiv b \ (mod\ p) \rightarrow \left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$

2. $\left(\frac{ac^2}{p}\right) = \left(\frac{a}{p}\right)$ для любого с $\in Z$, НОД(с, р) = 1

3. $\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \ (mod\ p)$ для НОД(а, р) = 1

4. $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$

5. $\left(\frac{1}{p}\right) = 1, \left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}} = \begin{cases} 1, \text{если } p \equiv 1 \ (mod\ 8) \\ -1, \text{если } p \equiv 3 \ (mod\ 8) \end{cases}$

6. $\left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}} = \begin{cases} 1, \text{если } p \equiv \ \pm 1 \ (mod\ 8) \\ -1, \text{если } p \equiv \ \pm 3 \ (mod\ 8) \end{cases}$

7. Квадратичный закон взаимности Гаусса

$$\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)(-1)^{\frac{(p-1)}{2}\frac{q-1}{2}}$$

Для любых нечетных простых чисел p, q.

2.6 Символ Якоби

Определение. Пусть натуральное число $n = \ p_1^{\alpha_1} \cdot \ldots \cdot p_k^{\alpha_k}$

Символом Якоби числа $a \in Z$ называется выражение:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha_1} \cdot \ldots \cdot \left(\frac{a}{p_k}\right)^{\alpha_k}$$

Символ Якоби для простого числа n совпадает с символом Лежандра и удовлетворяет почти всем свойствам символа Лежандра (хотя в общем случае символ Якоби не связан с квадратичными вычетами).

Символ Якоби позволяет упростить вычисление символа Лежандра (без разложения числа a на множители).

2.7 Алгоритм извлечения квадратного корня в кольце вычетов

Для решения сравнения $x^2 = n\ mod\ p$, где n – квадратичный вычет, p – нечетное простое число можно использовать вероятностный алгоритм извлечения квадратного корня в поле Z_p с арифметикой только этого поля. Вероятность успеха $P_i = \ \frac{1}{2}$.

3 Результаты работы

3.1 Алгоритмы: описание и временная сложность

3.1.1 Алгоритм разложения чисел в цепную дробь

Вход: два целых числа a, b.

Выход: таблица разложения подходящих дробей

Шаг 1: положить в массив P числа 0 и 1

Шаг 2: положить в массив Q числа 1 и 0

Шаг 3: пока b не равно 0 выполнить следующее

Шаг 3.1: положить c = целая часть от a / b

Шаг 3.2: добавить c в массив

Шаг 3.3: по формуле $P_i = q_i P_{i-1} + P_{i-2}$ вычислить новый элемент и добавить в массив P

Шаг 3.4: по формуле $Q_i = q_i Q_{i-1} + Q_{i-2}$ вычислить новый элемент и добавить в массив Q

Шаг 3.5: присвоить a целую часть от a / b, a b присвоить остаток и вернуться к шагу 1

Шаг 2: вывести массив с разложением, P, Q

Временная сложность алгоритма. O(L(a)*L(b))

3.1.2 Решение диофантовых уравнений

Вход: три целых числа a, b, c.

Выход: коэффициенты x, y

Шаг 1: найти НОД(a, b)

Шаг 2: если c делится на НОД, то продолжить

Шаг 3: вычислить разложение числа в цепную дробь a / b

Шаг 4: вычислить x в соответствии с формулой

$$x = (-1)^{k-1} \frac{c}{\text{НОД}(a,b)} Q_{k-1} + b$$

Шаг 5: вычислить y в соответствии с формулой

$$y = (-1)^{k-1} \frac{c}{\text{НОД}(a,b)} P_{k-1} + a$$

Шаг 6: вывести x, y

Временная сложность алгоритма. O(2*L(a)*L(b) + 2*L(c)*L(НОД(a, b)) *L(P_{k-1}) $* L(Q_{k-1})$)

3.1.3 Решение линейных сравнений

Вход: три целых числа a, b, m.

Выход: коэффициент x

Шаг 1: найти НОД(a, b)

Шаг 2: если c делится на НОД, то продолжить

Шаг 3: вычислить разложение числа в цепную дробь a / b

Шаг 4: вычислить x в соответствии с формулой

$$x = (-1)^{k-1} \frac{b}{\text{НОД}(a,m)} Q_{k-1} \ (mod\ m)$$

Шаг 5: вернуть x

Временная сложность алгоритма. O(2*L(a)*L(b) + L(b)*L(НОД(a, m)) $*L(Q_{k-1})$)

3.1.4 Поиск обратного элемента

Вход: два целых числа a, m

Выход: обратный элемент

Шаг 1: найти НОД(a, 1)

Шаг 2: если c делится на НОД, то продолжить

Шаг 3: вычислить разложение числа в цепную дробь a / 1

Шаг 4: вычислить x в соответствии с формулой

$$x = (-1)^{k-1} \frac{1}{\text{НОД}(a, m)} Q_{k-1} \ (mod\ m)$$

Шаг 5: вернуть x

Временная сложность алгоритма. O(2*L(a) + L(НОД(a, m)) *$L(Q_{k-1})$)

3.1.5 Символ Лежандра

Вход: $a \in Z, p -$ нечетное простое

Выход: 1 или -1 или 0

Шаг 1: если a < 0, то по свойству 4 выделить множитель $(\frac{-1}{p})$

Шаг 2: заменяем a на остаток от деления на p

Шаг 3: представляем a = $p_1^{\alpha_1} \cdot ... \cdot p_k^{\alpha_k}$ и при этом опускаем множители с четными степенями, а у множителей с нечетными убираем степени и вычисляем $(\frac{p_i}{p})$

Шаг 4: если $p_i = 2$, то вычисляем по свойству 6 $\left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}$

Шаг 5: к остальным символам $(\frac{p_i}{p})$ применяем квадратичный закон взаимности Гаусса.

Временная сложность алгоритма. O($\log^2 p$ + $\sqrt{a} + \sqrt{p}$)

3.1.6 Символ Якоби

Вход: $a \in Z, p$

Выход: 1 или -1 или 0

Шаг 1: заменить a на такое b, что $a = b \ (mod\ p)$ и $|b| < \frac{p}{2}$

Шаг 2: если $b < 0$, то по свойству 2 рассмотренном при описании символа Лежандра выделяем множитель $\left(\frac{-1}{p}\right)$

Шаг 3: если b – четное, то представляем $b = 2^t a_1$ и при нечетном t вычисляем $\left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}$

Шаг 4: к символу $\left(\frac{a_1}{p}\right)$ применяется квадратичный закон взаимности Гаусса

Временная сложность алгоритма. $O(\log^2 p)$

3.1.7 Алгоритм извлечения квадратного корня в кольце вычетов

Вход: $a \ \in Z$, нечетное простое p

Выход: квадратный корень

Шаг 1: вычисляем разложение $p - 1 = 2^m q$

Шаг 2: случайным образом выбрать b такое, что $\left(\frac{b}{p}\right) = \ -1$

Шаг 3: вычислить последовательность $a_1, \dots, a_n$ элементов поля Z_p и последовательность $k_1, \dots, k_n$ по правилу:

1. $a_1 = a, a_{i+1} = \ a_i b^{2^{m-k_i}} (mod\ p), i > 0$

2. k_i – наименьшее $k \geq 0$, при котором $a_i^{2^k q} \equiv 1 (mod\ p)$

Выполнение шага 2 заканчивается в тот момент, когда выполняется равенство $k_n = 0$.

Шаг 3: вычислить последовательность $r_n, \dots, r_1$ по правилу:

$r_n = \ a_n^{\frac{q+1}{2}} (mod\ p), r_i = r_{i+1} \left(b^{2^{m-k_i-1}}\right)^{-1} (mod\ p)$, $i > 0$

Шаг 4: положить $x_0 = r_1$- искомое решение

Временная сложность алгоритма. O($\log^4$p)

3.2 Тестирование алгоритмов

Разложение чисел в цепную дробь

```
Выберите функцию для тестирования:
1. Разложение чисел в цепную дробь
2. Решение диофантовых уравнений ax - by = c
3. Решение линейных сравнений ax = b (mod m)
4. Вычисление обратного элемента в кольце вычетов Z_m
5. Вычисление символа Якоби
6. Вычисление символа Лежандра
7. Вероятностный алгоритм извлечения квадратного корня в поле Z_p
Введите номер функции (1-7): 1
Введите числитель для разложения в цепную дробь: 19
Введите знаменатель: 15
Результат разложения: ([1, 3, 1, 3], [0, 1, 1, 4, 5, 19], [1, 0, 1, 3, 4, 15])
```

Рисунок 1. Тестирование разложения числа в цепную дробь

Решение диофантовых уравнений

```
Выберите функцию для тестирования:
1. Разложение чисел в цепную дробь
2. Решение диофантовых уравнений ax - by = c
3. Решение линейных сравнений ax = b (mod m)
4. Вычисление обратного элемента в кольце вычетов Z_m
5. Вычисление символа Якоби
6. Вычисление символа Лежандра
7. Вероятностный алгоритм извлечения квадратного корня в поле Z_p
Введите номер функции (1-7): 2
Введите a: 3
Введите b: 5
Введите c: 5
Решение уравнения: x = 15, y = 8
```

Рисунок 2. Тестирование алгоритма решения диофантовых уравнений

Решение линейных сравнений

```
Выберите функцию для тестирования:
1. Разложение чисел в цепную дробь
2. Решение диофантовых уравнений ax - by = c
3. Решение линейных сравнений ax = b (mod m)
4. Вычисление обратного элемента в кольце вычетов Z_m
5. Вычисление символа Якоби
6. Вычисление символа Лежандра
7. Вероятностный алгоритм извлечения квадратного корня в поле Z_p
Введите номер функции (1-7): 3
Введите a: 3
Введите b: 4
Введите m: 5
Решение сравнения: x = 3
```

Рисунок 3. Тестирование алгоритма решения линейных сравнений

Вычисление обратного элемента в кольце вычетов

```
Выберите функцию для тестирования:
1. Разложение чисел в цепную дробь
2. Решение диофантовых уравнений ax - by = c
3. Решение линейных сравнений ax = b (mod m)
4. Вычисление обратного элемента в кольце вычетов Z_m
5. Вычисление символа Якоби
6. Вычисление символа Лежандра
7. Вероятностный алгоритм извлечения квадратного корня в поле Z_p
Введите номер функции (1-7): 4
Введите a: 4
Введите m: 7
Обратный элемент: 2
```

Рисунок 4. Тестирование алгоритма вычисления обратного элемента в кольце вычетов

Вычисление символа Якоби

```
Выберите функцию для тестирования:
1. Разложение чисел в цепную дробь
2. Решение диофантовых уравнений ax - by = c
3. Решение линейных сравнений ax = b (mod m)
4. Вычисление обратного элемента в кольце вычетов Z_m
5. Вычисление символа Якоби
6. Вычисление символа Лежандра
7. Вероятностный алгоритм извлечения квадратного корня в поле Z_p
Введите номер функции (1-7): 5
Введите a: 10
Введите n: 13
Символ Якоби: 1
```

Рисунок 5. Тестирование алгоритма вычисления символа Якоби

Вычисления символа Лежандра

```
Выберите функцию для тестирования:
1. Разложение чисел в цепную дробь
2. Решение диофантовых уравнений ax - by = c
3. Решение линейных сравнений ax = b (mod m)
4. Вычисление обратного элемента в кольце вычетов Z_m
5. Вычисление символа Якоби
6. Вычисление символа Лежандра
7. Вероятностный алгоритм извлечения квадратного корня в поле Z_p
Введите номер функции (1-7): 6
Введите первое число: 10
Введите второе число (простое): 13
Символ Лежандра: 1
```

Рисунок 6. Тестирование алгоритма вычисления символа Лежандра

Вероятностный алгоритм извлечения квадратного корня в поле

```
Выберите функцию для тестирования:
1. Разложение чисел в цепную дробь
2. Решение диофантовых уравнений ax - by = c
3. Решение линейных сравнений ax = b (mod m)
4. Вычисление обратного элемента в кольце вычетов Z_m
5. Вычисление символа Якоби
6. Вычисление символа Лежандра
7. Вероятностный алгоритм извлечения квадратного корня в поле Z_p
Введите номер функции (1-7): 7
Введите a: 10
Введите p (простое): 13
Квадратный корень: 7
```

Рисунок 7. Тестирование алгоритма извлечения квадратного корня в поле

ЗАКЛЮЧЕНИЕ

В ходе работы были рассмотрены алгоритм разложения чисел в цепную дробь, приложения к алгоритму разложения чисел в цепную дробь, а именно, решение диофантовых уравнений, решение линейных сравнений и поиск обратного элемента в кольце вычетов. Также были рассмотрены алгоритмы вычисления символов Лежандра и Якоби и вероятностный алгоритм извлечения квадратного корня в поле.

Для проверки работоспособности этих алгоритмов была реализована программа на языке Python. Программы были протестированы и была подтверждена правильность их работы. Также была произведена оценка временной сложности алгоритмов.

ПРИЛОЖЕНИЕ А

Программа с реализованными алгоритмами

```
import random

import sympy


# Разложение чисел в цепную дробь

def continued_fraction(numerator, denominator):

    result = []

    P = [0, 1]

    Q = [1, 0]

    i = 2

    while denominator != 0:

        integer_part = numerator // denominator

        result.append(integer_part)

        P.append(result[i - 2] * P[i - 1] + P[i - 2])

        Q.append(result[i - 2] * Q[i - 1] + Q[i - 2])

        remainder = numerator % denominator

        numerator, denominator = denominator, remainder

        i += 1

    return result, P, Q

def normal_Euclid(a, b):

    while a !=0 and b != 0:

        if a > b:

            a = a % b

        else:

            b = b % a

    return a + b
```

```
# Решение диофантовых уравнений ax - by = c
def linear_diofant(a, b, c):
    nod = normal_Euclid(a, b)
    if c % nod != 0:
        return (0, 0)
    else:
        res, P, Q = continued_fraction(a, b)
        x = pow(-1, len(res) - 2) * c / nod * Q[-2] + b
        y = pow(-1, len(res) - 2) * c / nod * P[-2] + a
        return (int(x), int(y))
# Решение линейных сравнений ax = b (mod m)
def compare(a, b, m):
    nod = normal_Euclid(a, m)
    if b % nod != 0:
        return 0
    else:
        res, P, Q = continued_fraction(a, m)
        x = pow(-1, len(res) - 2) * b / nod * Q[-2]
        return int(x % m)
# Вычисление обратного элемента в кольце вычетов Zm
def reverse(a, m):
    return compare(a, 1, m)
# Вычисление символа Якоби
def jacobi_symbol(a, n):
    if n <= 0 or n % 2 == 0:
        return 0
    result = 1
```

```
    while a != 0:
        while a % 2 == 0:
            a //= 2
            if n % 8 in [3, 5]:
                result = -result
        a, n = n, a
        if a % 4 == 3 and n % 4 == 3:
            result = -result
        a %= n
    return result if n == 1 else 0
def factorization_number(num):
    n = num
    i = 2
    result = []

    while i * i <= n:
        while n % i == 0:
            result.append(i)
            n //= i
        i += 1

    if n > 1:
        result.append(n)

    return result
# Вычисление символа Лежандра
def legendre_symbol(first, second):
```

```
a = first
factors = factorization_number(second)
if len(factors) == 1:
    p = second
else:
    return None
numerators = []
result = 1

if a < 0:
    numerators.append(-1)
    a = abs(a)
a %= p
arr = factorization_number(a)
count_dict = {}
for num in arr:
    if num in count_dict:
        count_dict[num] += 1
    else:
        count_dict[num] = 1
for key, count in count_dict.items():
    if count % 2 == 1:
        numerators.append(key)

for elem in numerators:
    if elem == -1:
        result *= int(pow(-1, (p - 1) // 2))
```

```
            continue

        if elem == 2:
            result *= int(pow(-1, (p * p - 1) // 8))
        else:
            result *= legendre_symbol(p, elem) * int(pow(-1, ((p - 1) //
2) * ((elem - 1) // 2)))

    return result
# Вероятностный алгоритм извлечения квадратного корня в поле Zp с арифметикой только этого поля.
def sq(a, p):
    if not sympy.isprime(p):
        return None
    if legendre_symbol(a, p) != 1:
        return None
    q = p - 1
    m = 0
    while q % 2 == 0:
        q //= 2
        m += 1
    while True:
        b = random.randint(0, p - 1)
        if legendre_symbol(b, p) == -1:
            break
    buff = [a]
    k_buff = []
    while int(pow(buff[-1], q, p)) != 1:
```

```
        for k in range(p):
            if int(pow(buff[-1], int(pow(2, k, p)) * q, p)) == 1:
                k_buff.append(k)
                buff.append(buff[-1] * int(pow(b, int(pow(2, m - k_buff[-1], p)), p)) % p)
                break
    k_buff.append(0)
    buff.append(buff[-1] * int(pow(b, int(pow(2, m - k_buff[-1], p)), p)) % p)
    r = int(pow(buff[-1], (q + 1) // 2, p))
    for i in range(len(k_buff) - 1, -1, -1):
        l = reverse(int(pow(b, int(pow(2, m - k_buff[i] - 1, p)), p)), p)
        r = (r * l) % p
    return r


def test_function_choice(choice):
    if choice == 1:
        numerator = int(input("Введите числитель для разложения в цепную дробь: "))
        denominator = int(input("Введите знаменатель: "))
        result = continued_fraction(numerator, denominator)
        print(f"Результат разложения: {result}")
    elif choice == 2:
        a = int(input("Введите a: "))
        b = int(input("Введите b: "))
        c = int(input("Введите c: "))
        result = linear_diofant(a, b, c)
        if result == (0, 0):
```

```
            print("Числа не подходят, уравнение не разрешимо.")
        else:
            print(f"Решение уравнения: x = {result[0]}, y = {result[1]}")
    elif choice == 3:
        a = int(input("Введите a: "))
        b = int(input("Введите b: "))
        m = int(input("Введите m: "))
        result = compare(a, b, m)
        if result == 0:
            print("Уравнение не разрешимо.")
        else:
            print(f"Решение сравнения: x = {result}")
    elif choice == 4:
        a = int(input("Введите a: "))
        m = int(input("Введите m: "))
        result = reverse(a, m)
        if result is None:
            print("Уравнение не разрешимо.")
        else:
            print(f"Обратный элемент: {result}")
    elif choice == 5:
        a = int(input("Введите a: "))
        n = int(input("Введите n: "))
        result = jacobi_symbol(a, n)
        print(f"Символ Якоби: {result}")
    elif choice == 6:
        first = int(input("Введите первое число: "))
```

```
        second = int(input("Введите второе число (простое): "))
        result = legendre_symbol(first, second)
        if result is None:
            print(f"Второе число должно быть простым")
        else:
            print(f"Символ Лежандра: {result}")
    elif choice == 7:
        a = int(input("Введите a: "))
        p = int(input("Введите p (простое): "))
        result = sq(a, p)
        if result is None:
            print("Числа не подходят.")
        else:
            print(f"Квадратный корень: {result}")


def main():
    print("Выберите функцию для тестирования:")
    print("1. Разложение чисел в цепную дробь")
    print("2. Решение диофантовых уравнений ax - by = c")
    print("3. Решение линейных сравнений ax = b (mod m)")
    print("4. Вычисление обратного элемента в кольце вычетов Z_m")
    print("5. Вычисление символа Якоби")
    print("6. Вычисление символа Лежандра")
    print("7. Вероятностный алгоритм извлечения квадратного корня в поле Z_p")

    choice = int(input("Введите номер функции (1-7): "))
```

```
    if 1 <= choice <= 7:
        test_function_choice(choice)
    else:
        print("Неверный выбор. Пожалуйста, введите число от 1 до 7.")


if __name__ == "__main__":
    main()
```