Отчёт по лабораторной работе 7

Дискретное логарифмирование в конечном поле

Гебриал Ибрам Есам Зекри НФИ-02-22

Содержание

List of Tables

List of Figures

1 Цель работы

Реализация алгоритма, реализующий р-метод Полларда для задач дискретного логарифмирования.

2 Теоретические сведения

Пусть в некоторой конечной мультипликативной абелевой группе G задано уравнение

$$g^x = a$$

Решение задачи дискретного логарифмирования состоит в нахождении некоторого целого неотрицательного числа x, удовлетворяющего уравнению. Если оно разрешимо, у него должно быть хотя бы одно натуральное решение, не превышающее порядок группы. Это сразу даёт грубую оценку сложности алгоритма поиска решений сверху — алгоритм полного перебора нашёл бы решение за число шагов не выше порядка данной группы.

Чаще всего рассматривается случай, когда группа является циклической, порождённой элементом g. В этом случае уравнение всегда имеет решение. В случае же произвольной группы вопрос о разрешимости задачи дискретного логарифмирования, то есть вопрос о существовании решений уравнения , требует отдельного рассмотрения.

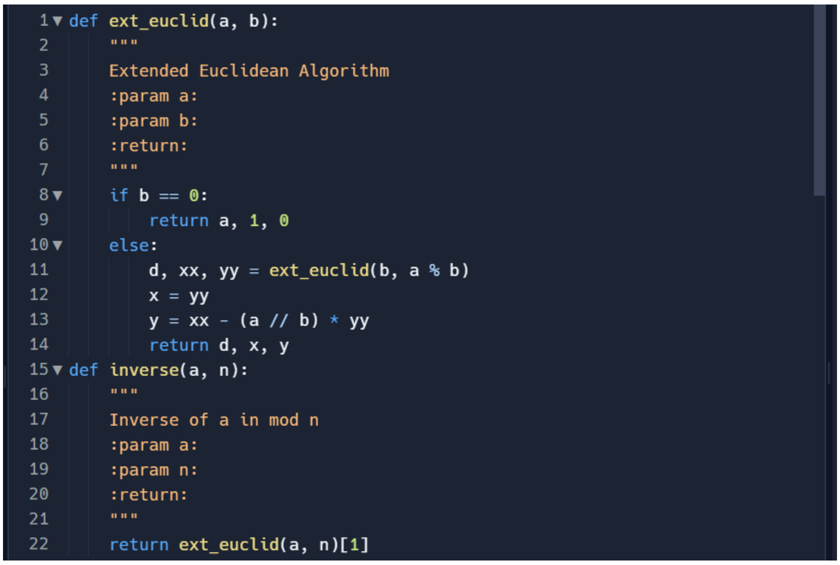
p-алгоритм Поллрада

- Вход. Простое число p, число a порядка r по модулю p, целое число b $1 < b < p$; отображение f, обладающее сжимающими свойствами и сохраняющее вычислимость логарифма.
- Выход. показатель x, для которого $a^x = b(modp)$, если такой показатель существует.

1. Выбрать произвольные целые числа u, v и положить $c = a^u b^v (mod p), d = c$
2. Выполнять $c = f(c)(mod p), d = f(f(d))(mod p)$, вычисляя при этом логарифмы для c и d как линейные функции от x по модулю r, до получения равенства $c = d(mod p)$
3. Приняв логарифмы для c и d, вычислить логарифм x решением сравнения по модулю r. Результат x или РЕШЕНИЯ НЕТ.

3 Выполнение лабораторной работы

1. Написал функцию ext_euclid и inverse (рис. 3.1)



```python
def ext_euclid(a, b):
    """
    Extended Euclidean Algorithm
    :param a:
    :param b:
    :return:
    """
    if b == 0:
        return a, 1, 0
    else:
        d, xx, yy = ext_euclid(b, a % b)
        x = yy
        y = xx - (a // b) * yy
        return d, x, y
def inverse(a, n):
    """
    Inverse of a in mod n
    :param a:
    :param n:
    :return:
    """
    return ext_euclid(a, n)[1]
```

Figure 3.1: Функция для расширенного алгоритма Евклида и обратного значения

2. Написал функцию xab (рис. 3.2)

```python
def xab(x, a, b, xxx_todo_changeme):
    """
    Pollard Step
    :param x:
    :param a:
    :param b:
    :return:
    """
    (G, H, P, Q) = xxx_todo_changeme
    sub = x % 3 # Subsets

    if sub == 0:
        x = x*xxx_todo_changeme[0] % xxx_todo_changeme[2]
        a = (a+1) % Q

    if sub == 1:
        x = x * xxx_todo_changeme[1] % xxx_todo_changeme[2]
        b = (b + 1) % xxx_todo_changeme[2]

    if sub == 2:
        x = x*x % xxx_todo_changeme[2]
        a = a*2 % xxx_todo_changeme[3]
        b = b*2 % xxx_todo_changeme[3]

    return x, a, b
```

Figure 3.2: Функция xab

3. Написал функцию pollard (рис. 3.3)

```python
def pollard(G, H, P):
    # P: prime
    # G: generator
    Q = int((P - 1) // 2)  # sub group
    x = G*H
    a = 1
    b = 1
    X = x
    A = a
    B = b
    for i in range(1, P):
        x, a, b = xab(x, a, b, (G, H, P, Q))
        X, A, B = xab(X, A, B, (G, H, P, Q))
        X, A, B = xab(X, A, B, (G, H, P, Q))

        if x == X:
            break
    nom = a-A
    denom = B-b
    # Необходимо вычислить обратное значение, чтобы правильно вычислить дробь по модулю q.
    res = (inverse(denom, Q) * nom) % Q
    if verify(G, H, P, res):
        return res

    return res + Q
```

Figure 3.3: Функция для алгоритма pollard

4. Написал функцию verify и блок работы программы(рис. 3.4)

```python
def verify(g, h, p, x):
    """
    Verifies a given set of g, h, p and x
    :param g: Generator
    :param h:
    :param p: Prime
    :param x: Computed X
    :return:
    """
    return  pow(g, x, p)== h

args = [
    (10, 64, 107),
]


for arg in args:
    res = pollard(*arg)
    print(arg, ': ', res)
    print("Validates: ", verify(arg[0], arg[1], arg[2], res))
    print()

```

Figure 3.4: Функция verify и блок работы программы

5. Получил результат (рис. 3.5)

```
>_ Console ×   Shell ×   +
(10, 64, 107) :  20
Validates:  True

>
```

Figure 3.5: Результат алгоритма

4 Выводы

Реализовал реализующий р-метод Полларда для задач дискретного логарифмирования.

5 Список литературы

1. Дискретное логарифмирование [Электронный ресурс] - Режим доступа: https://ru.wikipedia.org/wiki/Дискретное_логарифмирование