Лабораторная работа 7

Дискретное логарифмирование в конечном поле

Пологов Владислав Александрович

Содержание

List of Figures

1 Цель работы

1.1 Цель работы

Реализовать алгоритм, реализующий p-метод Полларда для задач дискретного логарифмирования

2 Описание реализации

2.1 Описание реализации

Для реализации алгоритмов использовались средства языка Python.

3 Реализация

3.1 Алгоритм, реализующий p-метод Полларда для задач дискретного логарифмирования

На вход будет подаваться 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. На выходе должны получить показатель x, для которого a ^ x = b (mod p), если такой показатель существует. Алгоритм представлен на рисунке 1. (рис. -fig. 3.1)

3.2 Алгоритм, реализующий p-метод Полларда

1. Выбрать произвольные целые числа u, v и положить $c \leftarrow a^u b^v \ (mod\ p), d \leftarrow c$.
2. Выполнять $c \leftarrow f(c)(mod\ p), d \leftarrow f\big(f(d)\big)(mod\ p)$, вычисляя при этом логарифмы для c и d как линейные функции от x по модулю r, до получения равенства $c \equiv d\ (mod\ p)$.
3. Приравняв логарифмы для c и d, вычислить логарифм x решением сравнения по модулю r. Результат: x или "Решений нет".

Figure 3.1: Алгоритм, реализующий p-метод Полларда для дискретного логарифмирования

3.3 Код, реализующий алгоритм

В начале использовалась функция, реализующая расширенный алгоритм Евклида, представленный на рисунке 2.(рис. -fig. 3.2) Также применена функция, для нахождения логарифма методом перебора, испульзуема для проверки.(рис. -fig. 3.3) Код, реализующий p-метод Полларда для задач дискретного логарифмирования представлен на рисунках 2, 3, 4. (рис. -fig. 3.4)

3.4 Код, реализующий алгоритм ч.1

```
def ext_euclid(a, b):
    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
```

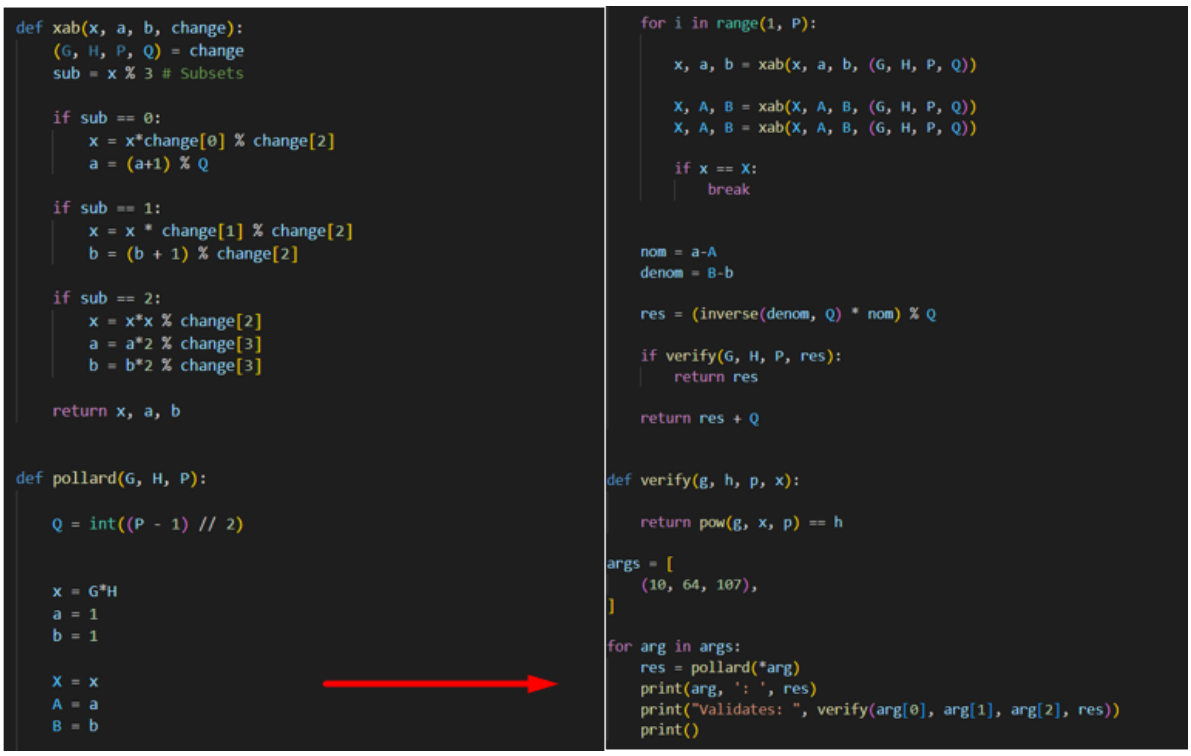
Figure 3.2: Код, реализующий p-метод Полларда

3.5 Код, реализующий алгоритм ч.2

```python
def premutive_log(g, a, b):
    '''
        Поиск дискретного логарифма перебором
        использовалась для тестирования
    '''
    x = 0
    while(x != b):
        if((g**x - a )%b == 0):
            '''если разность (g^x - a) делится на b '''
            return x
        x += 1
    return None

def test():
    g = 10
    a = 64
    m = 107
    print("Путём перебора", premutive_log(g, a, m), end = '\n\n')

test()

def ext_euclid(a, b):
    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):
    return ext_euclid(a, n)[1]
```

Figure 3.3: Код, реализующий p-метод Полларда

3.6 Код, реализующий алгоритм ч.3



```python
def xab(x, a, b, change):
    (G, H, P, Q) = change
    sub = x % 3 # Subsets

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

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

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

    return x, a, b


def pollard(G, H, P):

    Q = int((P - 1) // 2)


    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

    res = (inverse(denom, Q) * nom) % Q

    if verify(G, H, P, res):
        return res

    return res + Q


def verify(g, h, p, x):

    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: Код, реализующий p-метод Полларда

4 Вывод

- Реализован программно p-метод Полларда для задач дискретного логарифмирования. Проведена проверка методом перебора.