Лабораторная работа 7

Попов Дмитрий Павлович, НФИмд-01-23

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ПРЕЗЕНТАЦИЯ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

дисциплина: Математические основы защиты информации и информационной безопасности

Преподователь: Кулябов Дмитрий Сергеевич
Студент: Попов Дмитрий Павлович
Группа: НФИмд-01-23

МОСКВА
2023 г.

Прагматика выполнения лабораторной работы

Прагматика выполнения лабораторной работы

Требуется реализовать:

1. Алгоритм, реализующий p-метод Полларда для задач дискретного логарифмирования

Цель работы

Цель работы

Освоить на практике дискретное логарифмирование в конечном поле.

Выполнение лабораторной работы

Для реализации p-метода Полларда:

Для реализации p-метода Полларда:

1. Функция, реализующая p-метод Полларда
2. Функция нахождения НОД
3. Расширенный алгоритм Евклида для вычисления модульного обратного элемента

1. Функция, реализующая p-метод Полларда

1. Функция, реализующая p-метод Полларда

```python
def pollard_p_method(p, a, b, f, r, u, v):
    # Выбор произвольных чисел u, v
    c = (a ** u * b ** v) % p
    d = c

    # Итерации метода Полларда
    while True:
        c = f(c) % p
        d = f(f(d)) % p

        # Если c = d, вычисляем логарифмы для c и d
        if c == d:
            # Вычисляем логарифм x решением сравнения по модулю r
            # Решения нет, если r и p не взаимно просты
            if gcd(r, p - 1) != 1:
                return "Решения нет"

            # Вычисляем логарифм x
            x = (u - v * modinv((u - v), r) * (c - a ** u) % r) % r
            return x
```

Figure 1: pollard

2. Функция нахождения НОД

2. Функция нахождения НОД

```python
# Функция вычисления наибольшего общего делителя (GCD)
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a
```

Figure 2: gcd

3. Расширенный алгоритм Евклида для вычисления модульного обратного элемента

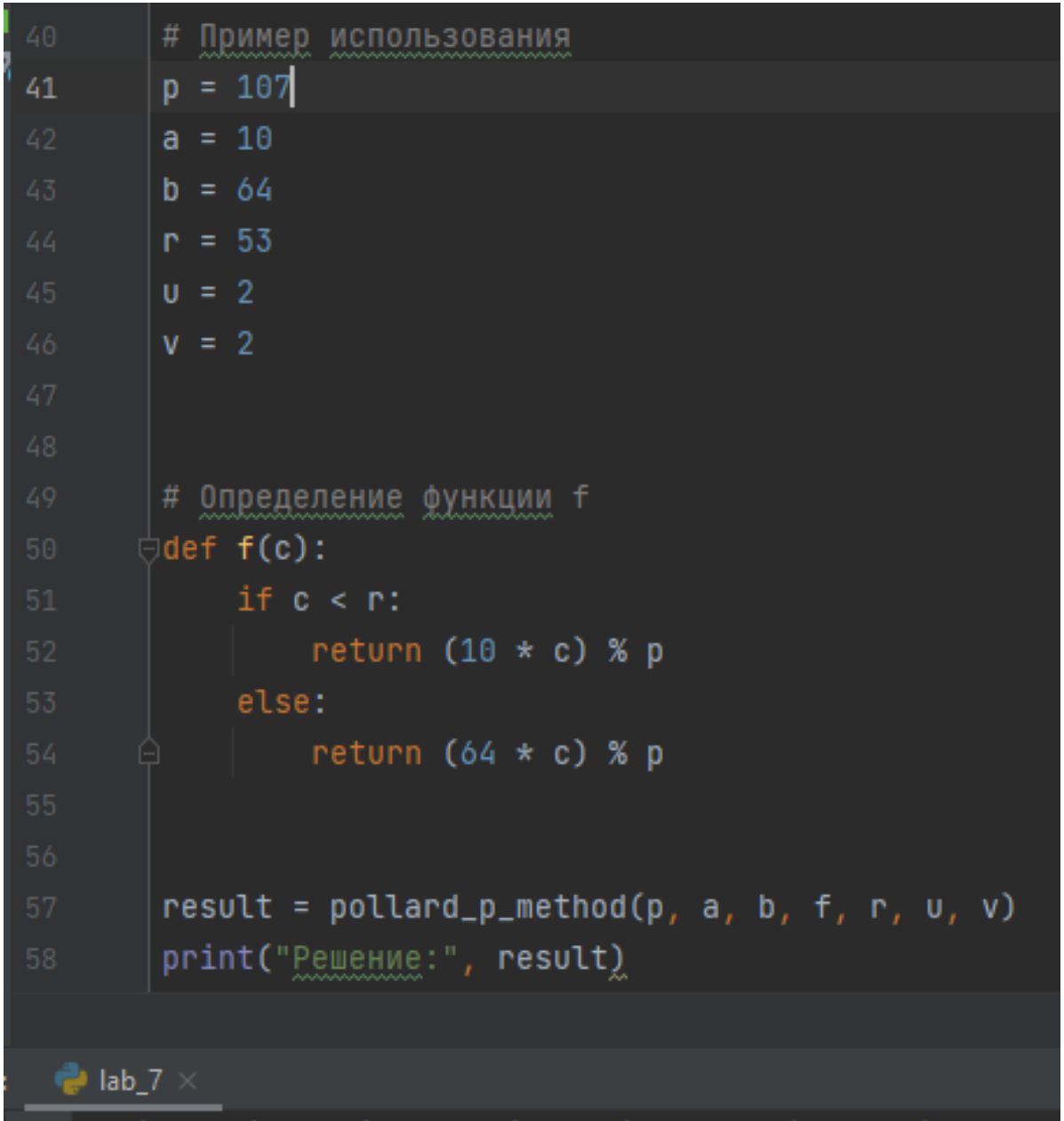
3. Расширенный алгоритм Евклида для вычисления модульного обратного элемента

```python
# Расширенный алгоритм Евклида для вычисления модульного обратного элемента
def modinv(a, m):
    m0, x0, x1 = m, 0, 1
    while a > 1:
        q = a // m
        m, a = a % m, m
        x0, x1 = x1 - q * x0, x0
    return x1 + m0 if x1 < 0 else x1
```

Figure 3: modinv

Основная фунция запуска где получаем входные значения

Основная фунция запуска где получаем входные значения



Выводы

Выводы

В результате выполнения работы я освоил на практике дискретное логарифмирование в конечном поле.