

```

def extended_gcd(a, b):
    """Расширенный алгоритм Евклида для нахождения НОД и коэффициентов Безу."""
    if a == 0:
        return b, 0, 1
    else:
        g, x, y = extended_gcd(b % a, a)
        return g, y - (b // a) * x, x

def mod_inverse(a, m):
    """Нахождение обратного элемента по модулю."""
    g, x, y = extended_gcd(a, m)
    if g != 1:
        return None # Обратный элемент не существует, если a и m не взаимно просты
    else:
        return x % m

def pollards_rho_discrete_log(a, b, p, order):
    """Реализация р-метода Полларда для дискретного логарифмирования."""
    def f(c):
        """Случайное отображение."""
        if c < p // 2:
            return (a * c) % p
        else:
            return (b * c) % p

    def log_f(c, u, v):
        """Вычисление логарифма для отображения f."""
        if c < p // 2:
            return (u + 1) % order, v % order
        else:
            return u % order, (v + 1) % order

    # Шаг 1: Инициализация
    u, v = 2, 2 # Произвольные начальные значения
    c = (pow(a, u, p) * pow(b, v, p)) % p
    d = c
    u_c, v_c = u, v
    u_d, v_d = u, v

    # Шаг 2: Поиск коллизии
    while True:
        # Обновляем c и его логарифм
        c = f(c)
        u_c, v_c = log_f(c, u_c, v_c)

        # Обновляем d и его логарифм (два шага)
        d = f(f(d))
        u_d, v_d = log_f(f(d), u_d, v_d)
        u_d, v_d = log_f(d, u_d, v_d)

        # Проверяем на коллизию
        if c == d:
            break

```


```
# Шаг 3: Решение сравнения
# Уравнение:  $(u_c + v_c * x) \equiv (u_d + v_d * x) \pmod{\text{order}}$ 
A = (v_c - v_d) % order
B = (u_d - u_c) % order

# Решаем уравнение  $A * x \equiv B \pmod{\text{order}}$ 
gcd, x, y = extended_gcd(A, order)
if B % gcd != 0:
    return None # Решение не существует

x = (x * (B // gcd)) % (order // gcd)
return x

# Пример использования
p = 107 # Простое число
a = 10  # Основание
b = 64  # Число, для которого ищем логарифм
order = 53 # Порядок числа a по модулю p

x = pollards_rho_discrete_log(a, b, p, order)
if x is not None:
    print(f"Дискретный логарифм x = {x}")
    # Проверка
    if pow(a, x, p) == b:
        print("Проверка пройдена:  $a^x \equiv b \pmod{p}$ ")
    else:
        print("Ошибка:  $a^x \not\equiv b \pmod{p}$ ")
else:
    print("Решения нет")
```

 Дискретный логарифм x = 20
Проверка пройдена: $a^x \equiv b \pmod{p}$