лабораторной работе №7
Дискретное логарифмирование
Студент: Хамза хуссен

---

1. Задачи исследования

Целью данной лабораторной работы является изучение и практическое исследование задачи дискретного логарифмирования, включая методы её решения в конечных циклических группах.

---

2. Основные теоретические положения

Рассматривается задача нахождения показателя степени x в уравнении:

$$g^x = a,$$

где g и a — элементы конечной мультипликативной абелевой группы G.

- **Разрешимость**: Если группа G является циклической и порождается элементом g, решение существует для любого $a \in G$.
- **Сложность**: В общем случае верхняя оценка сложности поиска решения методом полного перебора равна порядку группы $|G|$.

2.1. Алгоритм Полларда (ρ-метод)

Входные параметры:

- Простое число p;
- Элемент a порядка r по модулю p;
- Целое число b, где $1 < b < p$;
- Сжимающее отображение f, сохраняющее возможность вычисления логарифма.

Выходные данные:
Показатель x, удовлетворяющий сравнению $a^x \equiv b (\mathrm{mod} p)$, либо сообщение об отсутствии решения.

Этапы алгоритма:

1. **Инициализация**: Выбрать случайные целые u, v и вычислить:

$$c \equiv a^u b^v (\mathrm{mod} p), d \equiv c.$$

2. **Итерационный процесс**:
    - Обновлять значения:

$$c \leftarrow f(c)(\mathrm{mod} p), d \leftarrow f(f(d))(\mathrm{mod} p),$$

      одновременно выражая логарифмы c и d как линейные функции от x по модулю r.
    - Повторять до выполнения условия $c \equiv d(\mathrm{mod} p)$.
3. **Завершение**: Используя полученные линейные соотношения, решить сравнение для нахождения x по модулю r. Если решение не существует, вернуть соответствующее сообщение.

---

3. Выводы

В ходе работы изучены теоретические основы задачи дискретного логарифмирования и рассмотрен один из вероятностных методов её решения — алгоритм Полларда. Практическое применение алгоритма позволяет эффективно находить логарифмы в циклических группах, что важно для криптографических приложений.

4.Выполнение работы

4.1.Реализация алгоритма на языке Python

```
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]

def xab(x, a, b, xxx):
    (G, H, P, Q) = xxx
```

```
    sub = x%3

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

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

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

    return x, a, b

def pollrad(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 = [(5, 64, 107)]

for arg in args:
```

```
res = pollrad(*arg)
print(arg, " : ", res)
print("Validates: ", verify(arg[0], arg[1], arg[2], res))
```

4.2.Контрольный пример

```
print("Validates: ", verify(arg[0], arg[1], arg[2], res))
```

```
(5, 64, 107)  :  52
Validates:  True
```