Лабораторная работа №7

Дискретное логарифмирование в конечном поле

Доборщук Владимир Владимирович, НФИмд-02-22

Содержание

Список иллюстраций

Список таблиц

1 Цель и задачи работы

Цель — Изучить алгоритмы для задач дискретного логарифмирования.

Задачи:

- Реализовать алгоритм для задач дискретного логарифмирования через p-метод Полларда

2 Теоретическая информация

Все теоретическое описание дано в описании лабораторной работы.

3 Выполнение лабораторной работы

При выполнении лабораторной работы мы строго следовали алгоритмике, представленной в описании.

3.1 Реалиазация и тестирование

Программный код выглядит следующим образом:

```
# Laboratory Work
# Theme: Discrete logarthmification
# Author: Vladimir Doborschuk

# --- Modules ---

import numpy as np

# --- Functions ---

# --- mod(a, b) ---

def mod(a ,b):
    return a % b

# --- find mod order ---
```

```
def order(a, p):
    x = 1
    while mod(a**x - 1, p) != 0:
        x += 1

    return x

# --- Pollard's P-method for Log ---

'''
a - основание
b - значение остатка
p - простое число
'''
def po_method(a: int, b: int, p: int):
    print(f"\n{a}^(x) = {b} mod {p}")

    ↪ print("-----------------------------------------------------------------
    ↪ ")
    print('|\tc\t|\tlog c\t|\td\t|\tlog d\t|')

    ↪ print("-----------------------------------------------------------------
    ↪ ")

    u = np.random.randint(4)
    v = np.random.randint(4)
    r = order(a, p)
```

```
c = mod(np.power(a, u) * np.power(b, v), p)
d = c

u_c, u_d = u, u
v_c, v_d = v, v

print(f'|\t{c}\t|\t{u_c}+{v_c}x\t|\t{d}\t|\t{u_d}+{v_d}x\t|')

def f(x, u_x, v_x):
    if x < r:
        return mod(a*x, p), u_x + 1, v_x
    else:
        return mod(b*x, p), u_x, v_x + 1

c, u_c, v_c = f(c, u_c, v_c)
tmp_d = f(d, u_d, v_d)
d, u_d, v_d = f(tmp_d[0], tmp_d[1], tmp_d[2])

while mod(c, p) != mod(d, p):

    print(f'|\t{c}\t|\t{u_c}+{v_c}x\t|\t{d}\t|\t{u_d}+{v_d}x\t|')
    c, u_c, v_c = f(c, u_c, v_c)
    tmp_d = f(d, u_d, v_d)
    d, u_d, v_d = f(tmp_d[0], tmp_d[1], tmp_d[2])

print(f'|\t{c}\t|\t{u_c}+{v_c}x\t|\t{d}\t|\t{u_d}+{v_d}x\t|')

print("----------------------------------------------------------------
")
```

```
    x = 1
    # print(v_c - v_d, u_d - u_c)
    while mod((v_c - v_d)*x, r) != mod(u_d - u_c, r):
        x += 1

    print(f"x = {x}")
    print(f"\n{a}^({x}) = {b} mod {p}")

    print("-----------------------------------------------------------------
    ")
    return x

# --- Main ---

def main():
    po_method(10, 64, 107)
    po_method(2, 1, 15)


if __name__ == "__main__":
    main()
```

При запуске получаем следующие результаты:

```
10^(x) = 64 mod 107
-----------------------------------------------------------------
|       c       |       log c   |       d       |       log d   |
-----------------------------------------------------------------
|       101     |       0+3x    |       101     |       0+3x    |
```

44	0+4x	12	1+4x
12	1+4x	23	3+4x
13	2+4x	53	5+4x
23	3+4x	92	5+6x
16	4+4x	30	6+7x
53	5+4x	47	7+8x
75	5+5x	99	9+8x
92	5+6x	16	10+9x
3	5+7x	75	11+10x
30	6+7x	3	11+12x
86	7+7x	86	13+12x

---

x = 20

10^(20) = 64 mod 107

2^(x) = 1 mod 15

---

c	log c	d	log d
1	0+2x	1	0+2x
2	1+2x	4	2+2x
4	2+2x	4	2+4x

---

x = 2

2^(2) = 1 mod 15

---

4 Выводы

В рамках выполненной лабораторной работы мы изучили и реализовали p-метод Полларда для задач дискретного логарифмирования.

Список литературы