Лабораторная работа №7

Дискретное логарифмирование в конечном поле

Доборщук Владимир Владимирович, НФИмд-02-22

Содержание

# 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