

區塊鏈作業一報告

利用 Pollard' s Rho 方法解離散對數問題 (DLP)

姓名：胡樂麒

學號：B1228005

長庚大學人工智慧學系大三

2025 年 10 月

摘要

本報告以兩種方式完成離散對數問題 (Discrete Logarithm Problem, DLP): (1) 使用 Python 自行實作 Floyd-Pollard Rho 演算法; (2) 以 PARI/GP 套件進行快速驗證。最終兩種方法所得結果一致，成功求得正確的離散對數值。

1 作業目標

本作業的使用 Pollard' s Rho 方法解離散對數問題，其數學形式如下：

$$h \equiv g^x \pmod{p}$$

其中給定質數模數 p 、生成元 g 及元素 h ，要求出未知整數 x 。

2 方法一：使用 Python 實作 Floyd-Pollard Rho 演算法

2.1 原理概述

Pollard' s Rho 方法藉由定義一個擬隨機函數 $f(x)$ ：

$$x_{i+1} = f(x_i),$$

不斷運算直到出現「碰撞」(collision)： $x_i = x_j$ ，此時有：

$$g^{a_i} h^{b_i} = g^{a_j} h^{b_j} \Rightarrow x = -\frac{a_i - a_j}{b_i - b_j} \pmod{n}.$$

演算法 average-case time complexity $O(\sqrt{n})$ 。

本程式運用 **Floyd cycle detection** (烏龜與兔子法) 尋找循環。

2.2 Python 程式碼

Listing 1: Floyd–Pollard Rho 演算法實作 (Python)

```
1 import random
2 from math import gcd
3
4 def pollards_rho_floyd(g, h, p, n, max_attempts=5):
5
6     def partition(x):
7         if x % 3 == 0:
8             return 0
9         elif x % 3 == 1:
10            return 1
11        else:
12            return 2
13
14    def f(x, a, b):
15        part = partition(x)
16        if part == 0:
17            x = (h * x) % p
18            b = (b + 1) % n
19        elif part == 1:
20            x = (x * x) % p
21            a = (2 * a) % n
22            b = (2 * b) % n
23        else:
24            x = (g * x) % p
25            a = (a + 1) % n
26        return x, a, b
27
28    for attempt in range(max_attempts):
29        x_t, a_t, b_t = g, 1, 0
30        x_h, a_h, b_h = g, 1, 0
31
32        for i in range(1, 2 * n):
33            x_t, a_t, b_t = f(x_t, a_t, b_t)
34            x_h, a_h, b_h = f(*f(x_h, a_h, b_h))
35
36            if x_t == x_h:
37                r = (a_t - a_h) % n
38                s = (b_h - b_t) % n
39                if s == 0 or gcd(s, n) != 1:
40                    print(f"[嘗試{attempt+1}] s 不可逆, 重試")
41                break
```

```

42         try:
43             s_inv = pow(s, -1, n)
44             x_result = (r * s_inv) % n
45             return x_result
46         except ValueError:
47             print(f"[嘗試_{attempt+1}]_模反元素錯誤，重試")
48             break
49     raise ValueError("超過最大嘗試次數，仍未找到解")
50
51 if __name__ == "__main__":
52     import time
53
54     p = 58002118547
55     n = 29001059273
56     g = 13513236970
57     h = 39818986975
58
59     print("開始執行_Floyd-Pollard_rho_DLP")
60
61     try:
62         x = pollards_rho_floyd(g, h, p, n)
63         print(f"找到的_x=_{x}")
64         print(f"驗證: {g}^{x}_mod_{p}=_{pow(g, x, p)}")
65     except ValueError as e:
66         print(e)

```

2.3 實驗結果

$p = 58002118547$, $n = 29001059273$, $g = 13513236970$, $h = 39818986975$

程式輸出結果如下：

開始執行 Floyd-Pollard rho DLP

找到的 $x = 10802198852$

驗證: $13513236970^{10802198852} \bmod 58002118547 = 39818986975$

3 方法二：使用 PARI/GP 套件

3.1 操作步驟

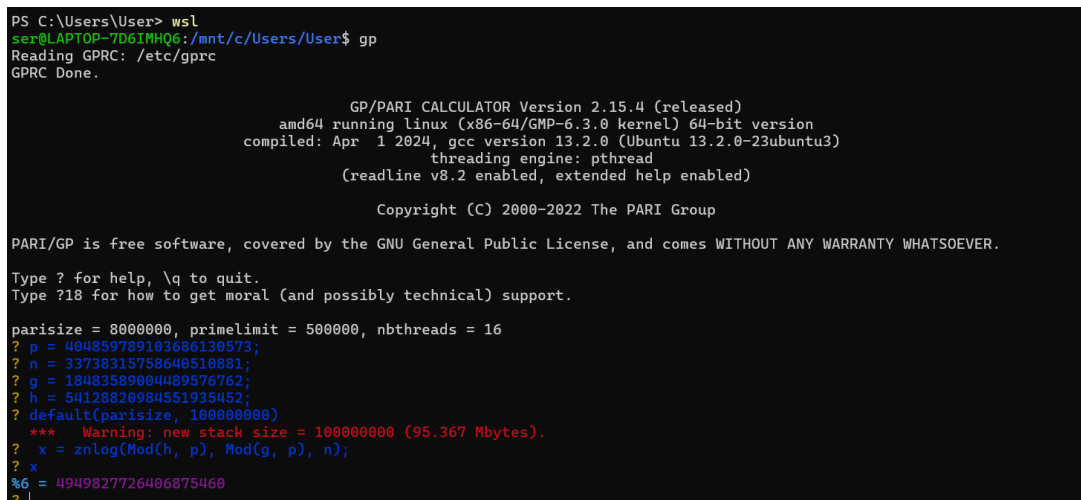
```

1 $ gp
2 p = 404859789103686130573;
3 n = 33738315758640510881;
4 g = 18483589004489576762;
5 h = 54128820984551935452;
6 default(parisize, 100000000)
7 x = znlog(Mod(h, p), Mod(g, p), n)
8 x

```

執行結果：

$$x = 4949827726406875460$$



```

PS C:\Users\User> wsl
ser@LAPTOP-7D6IMH06:/mnt/c/Users/User$ gp
Reading GPRC: /etc/gprc
GPRC Done.

      GP/PARI CALCULATOR Version 2.15.4 (released)
      amd64 running linux (x86_64/GMP-6.3.0 kernel) 64-bit version
      compiled: Apr 1 2024, gcc version 13.2.0 (Ubuntu 13.2.0-23ubuntu3)
      threading engine: pthread
      (readline v8.2 enabled, extended help enabled)

      Copyright (C) 2000-2022 The PARI Group

PARI/GP is free software, covered by the GNU General Public License, and comes WITHOUT ANY WARRANTY WHATSOEVER.
Type ? for help, \q to quit.
Type ?18 for how to get moral (and possibly technical) support.

parisize = 8000000, primelimit = 500000, nbthreads = 16
? p = 404859789103686130573;
? n = 33738315758640510881;
? g = 18483589004489576762;
? h = 54128820984551935452;
? default(parisize, 100000000)
*** Warning: new stack size = 100000000 (95.367 Mbytes).
? x = znlog(Mod(h, p), Mod(g, p), n);
? x
%6 = 4949827726406875460
?

```

圖 1: 於 WSL 下使用 PARI/GP 計算離散對數的執行畫面

4 比較與結論

方法	工具	優點	缺點
Python Pollard' s Rho	自行實作	可以看到運算流程	速度較慢
PARI/GP	電腦代數系統	運算快速且精確	黑箱運算

我在本作業以這兩種方式求解 DLP，且兩種方法結果一致。我覺得 PARI/GP 比 Python 實作方便且迅速，就算大數字也能很快得出結果，只是要先在終端機下載 wsl 環境與套件。