東海大學應用數學系

碩士論文

指導教授:沈淵源

離散對數問題的探討

研究生:洪郁程

中華民國一百零八年六月

東海大學
TUNGHAI UNIVERSITY
1955

東海大學應用數學系

碩士論文

指導教授:沈淵源

離散對數問題的探討

研究生:洪郁程

中華民國一百零八年六月

東 海 大 學

應用數學系

碩士學位口試委員審定書

本系碩士班 洪郁程 君

所提論文 On the Discrete Logarithm Problems
(離散對數問題的探討)

合於碩士班資格水準，業經本委員會評審通過，特此證明。

口試委員：劉康海

黃皇男

指導教授：陸洪湧

系 主 任：胡馨云

中 華 民 國 一〇八 年 六 月 三 日

摘　要

此篇論文主要在講述離散對數問題 (discrete logarithm problem) 的解。在密碼學中，離散對數問題是一個擁有廣泛應用的研究主題，有許多系統，安全性便是建立在解離散對數問題的難題上。文中首先介紹離散對數問題與密碼學中關係，進一步說明嬰步巨步演算法、波立格-赫爾曼演算法以及指數計算法等三種解離散對數問題的方法，最後呈現使用 MATLAB 語言實現這些算法程式的計算結果並比較計算時間的多寡。

關鍵詞： 密碼學、離散對數問題

Abstract

In this thesis, we are concerned to calculate discrete logarithm problem. In cryptography, the discrete logarithm problem is a research topic with a wide range of applications. There are many systems that security is built on the difficulty of solving discrete logarithm problems.

First, we introduce the relationship between discrete logarithm problem and cryptography. And explain three methods for solving discrete logarithm problems. Baby-Step Giant-step methods, Pohlig-Hellman Algorithm and Index calculus methods. Finally, the calculation results and the calculation time of these algorithm are implemented using the MATLAB to compared.

誌謝

此篇論文能夠在一年內完成真的要感謝我的指導教授，沈淵源教授的指導，一開始決定論文的方向跟論文章節的排序跟撰寫，都給我許多指引，讓我這個懵懂的研究生有學習的機會，對研究所的相關規定或是助教工作上有不懂的地方，老師也總是會熱心的幫助我，由衷的感謝老師。

此外，也很感謝口試委員黃皇男教授跟劉康滿教授的指導與指正，還有在程式方面的提點，讓我知道論文不夠好的地方，特別感謝黃皇男教授在我的指導老師不在的時候給予我程式上的幫助，尤其是因為我對於撰寫程式這方面較為不熟悉，接著也要感謝系上的許多老師在研究所課程中的指導，在我論文與課業雙頭並進的時候提供幫助，也使我在其中受益良多，我相信在研究所期間學到的東西都會對未來的我有所幫助。

最後感謝在研究所一起努力的朋友們，在我需要的時候都會幫助我，還有感謝我的家人的支持，讓我在學習上能無後顧之憂。

目錄

第一章　緒論與預備知識.....1

第一節 密碼學的進展.....1

第二節 公鑰密碼與離散對數.....3

第三節 預備知識.....4

第二章　離散對數問題解法.....9

第一節 演算法簡介.....9

a. 窮舉法:.....9

b. 嬰步巨步法:.....10

c. 離散對數 波赫法.....12

d. 指數計算法.....15

第三章　演算法實例與討論.....17

第一節:計算例子.....17

a. 嬰步巨步法:.....17

b. 波赫法:.....18

c. 指數計算法:.....22

第二節:演算法.....23

第三節: 結果與討論.....26

附錄.....28

參考文獻 33

表圖目次

表 1……27

表 2……27

第一章　緒論與預備知識

第一節 密碼學的進展

密碼學的發展也已經有數千年，從古至今的君王或是將領，都需要一種有效率而且安全的方法來治理國家或指揮軍隊，若是讓機密訊息不小心傳到不當人士手中，將可能產生非常嚴重的後果，所以就藉由密碼的技術來防備，而為了掌握對手的情報也是花了大量人力在解碼上，所以戰爭與外交就是使密碼學精進的推手，二戰時密碼學更是大幅進步。

最早開始所使用的就是位移密碼來加密，也就是將文章中的字母做位移，位移後的文章就是密文，這也是種簡單的數學運算加密，歷史上著名的凱薩大帝也曾使用過。而後也發展出像是仿射密碼跟維吉內爾密碼的對稱式加密法，也就是加密與解密使用相同鑰匙的密碼系統，不過都是由替代和位移法作為基礎，一旦鑰匙被破解很容易就被解出密文。在 1970 年代，密碼學有了重大的突破，Diffie 和 Hellman 兩位學者發表了公鑰密碼系統，這個演算法主要是以數學函數為基礎，並非替代和位移，利用模指數運算來協定雙方的鑰匙，讓密文在傳送時即使被發現也因為指數模數的不容易逆運算無法馬上算出鑰匙。

隨著現今時代進步科技進展快速，人跟人之間的交流溝通不再

僅限於書信，也不再有距離的限制，藉由網路的幫助能夠傳送到世界各地，但是也因為網路的發達，資料傳遞越來越透明，使得內容很容易有外洩的危機。所以在傳遞前先經由加密已經是不可或缺的事情。數學的發展也造就了密碼學的演進，近代密碼學也隨著數學中的數論這門研究出現了許多複雜的加密與解密法，但是現今計算方法與電腦的發展，使得較為簡單的運算或是演算法已經可以輕易地破解，所以也需要更嚴謹不容易破解的密碼系統。

第二節 公鑰密碼與離散對數

公鑰密碼系統大都建立在計算複雜度較高的數學難題中，使之較有安全性，而離散對數問題在公鑰密碼學就是極受重視並且廣泛應用，其中許多密碼系統其安全性便是建立在解離散對數的難題上，根據它通常沒有非常有效率的方法來計算的特性，來假設尋找離散對數的問題解作為密碼系統的基礎，現在除了少數特殊情況下可以快速計算，『是否存在離散對數問題的多項式時間經典算法？』這件事是科學家們還沒解答的問題。

ElGamal 在 1985 年提出了一個密碼系統是與利用離散對數問題來設計密碼相關，演算法如下:

假如三毛要傳遞數位信息x給四郎

1. 四郎選取一大質數p及整數$\alpha(\text{mod } p)$，然後選取一秘密整數a計算$\beta \equiv \alpha^a(\text{mod } p)$。
2. 四郎將p, α, β三數公開，但將a保持私密。
3. 三毛根據p, α, β，選取一隨機整數k，算出密碼文(y_1, y_2)，$y_1 \equiv \alpha^k, y_2 \equiv x\beta^k \pmod{p}$，將$(y_1, y_2)$傳給四郎。
4. 因為四郎知道a為多少所以可以輕鬆算出原本的明文

$$x \equiv y_2 {y_1}^{-a} \pmod{p}.$$

第三節 預備知識

在這我們會對離散對數以及它所需要的數論知識做些介紹

1. 質數: 除了 1 和本身外，無法被其他自然數整除的數，在許多密碼學演算法中選取大質數是必須的，在離散對數問題中也時常用到。

2. 同餘:若n整除$(a-b)$我們說a與b在模n下為同餘， 一般寫為

$$a \equiv b \pmod{n}$$

例 : $10 \equiv 1 \pmod{3}$。

3. 反元素:給一整數$a < n$，且$\gcd(a,n) = 1$，存在唯一一個整數x，使得$ax \equiv 1 \pmod{n}$，x即為a在模n下的反元素。

例: $a = 7$，$n = 11$，$7x \bmod 11 = 1$，則x=8 為a的反元素。

4. 費馬小定理 (Fermat's Little Theorem) : 假如a為整數p為正質數，且$\gcd(a,p) = 1$，則$a^{p-1} \equiv 1 \pmod{p}, \forall a \in \mathbb{Z}$.

例: $3^{10} \equiv 1 \pmod{11}$。

5. 歐拉函數 (Euler's Phi Function) : 對整數n，所有小於或等於n且與n互質的正整數數量，以函數$\varphi(n)$表示

$$\varphi(n) = n\prod\left(1-\frac{1}{p_i}\right)$$

p_i為n的所有質因數。

例: $\varphi(36) = 36 \times \left(1-\frac{1}{2}\right) \times \left(1-\frac{1}{3}\right) = 12$。

6. 歐拉定理 :為費馬小定理與歐拉函數的推廣，若$\gcd(a,n)=1$，也就是a、n互質，則$a^{\varphi(n)} \equiv 1 \pmod{n}, \forall a \in \mathbb{Z}$。

例: $a=3; n=10; \varphi(n)=4$

$a^{\varphi(n)} = 3^4 = 81 \equiv 1 \ mod(10)$。

7. 原根 (Primitive Roots) :也稱「乘法生成元素」，在離散對數問題的計算有其重要性。對兩正整數 a、n，由歐拉定理可知，若$\gcd(a,n)=1$，則存在正整數$\varphi(n) \leq n-1$，使得

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

定義x為使$a^{\varphi(n)} \equiv 1 \pmod{n}$成立的最小$\varphi(n)$，由前面得知$x$一定小於$\varphi(n)$，所以若$x=\varphi(n)$，我們則稱$a$是模$n$的原根。

例: 模 11 之下 7 的次冪，a=7，n=11

j	1	2	3	4	5	6	7	8	9	10
7^j	7	5	2	3	10	4	6	9	8	1

$7^{10} \equiv 1 \pmod{11}, \quad \varphi(11)=10$

所以 7 是模 11 的一個原根。

一般而言，當n為一質數，模n的一個原根就是其中一個非零元素a使得每一模n的非零元素都是a的一個次冪，也就是說 1, a, $a^2, \ldots, a^{n-1}$ 模 n 皆不同餘。

原根的性質:

1. 若n為一整數，則$g^n \equiv 1 \pmod{p} \Leftrightarrow n \equiv 0 \pmod{p-1}$

證明:$(\Longrightarrow)$ 若$g^n \equiv 1 \pmod{p}$，$p-1 \mid x$。假設不對，那

$x = (\mathrm{p}-1)\mathrm{q}+\mathrm{r}$，$0 < \mathrm{r} < \mathrm{p}-1$。所以得到

$$1 \equiv g^n \equiv (g^q)^{p-1} g^r \equiv 1 * g^r \equiv g^r \pmod{p}$$

(⇐)由費馬小定理得證。

2. 若 i、j 為整數，$g^i \equiv g^j (\mathrm{mod}\ p) \Leftrightarrow \mathrm{i} \equiv j (\mathrm{mod}\ p-1)$。

   證明:假如$\mathrm{i} > \mathrm{j}$，

   $$g^i \equiv g^j \Leftrightarrow g^{i-j} \equiv 1 \Leftrightarrow j-k \equiv 0 \Leftrightarrow j \equiv k \pmod{p-1}$$

3. 若m為$2,4,p^k,2p^k$(p為奇質數)其中之一時，原根才存在。設模m有原根，那會有$\varphi(\varphi(m))$個對模m不相同的原根。只要找到了其中一個原根，就很容易找到其他的原根。

驗證一個數是質數 p 的原根的方法:

在模p之下，計算每一個次冪

$$\{g^{(p-1)/q_1}, g^{(p-1)/q_2}, \ldots, g^{(p-1)/q_n}\}$$

q_n為$p-1$的質因數，若都不為 1，g就是質數p的一個原根。

例:找質數 23 的最小原根

$p = 23,\ p-1 = 2 * 11$

$\boldsymbol{g}$	2	3	5
$\boldsymbol{g^{\frac{p-1}{q_1}} (\mathrm{mod}\ p) = g^{11} (\mathrm{mod}\ 23)}$	1	1	22
$\boldsymbol{g^{\frac{p-1}{q_2}} (\mathrm{mod}\ p) = g^{2} (\mathrm{mod}\ 23)}$	4	9	2

所以 5 就是質數 23 的最小原根。

8. 中國餘式定理(Chinese Remainder Theorem):

令n_1、n_2、…、n_k為兩兩互質的正整數，令n=$n_1 \times n_2 \cdots \times n_k$。

則同餘聯立組

$$\begin{cases} x \equiv x_1 \pmod{n_1} \\ x \equiv x_2 \pmod{n_2} \\ \vdots \\ x \equiv x_k \pmod{n_k} \end{cases}$$

其解為

$$x = \sum_{i=1}^{k} x_i M_i s_i \pmod{n}$$

其中$M_i = n/n_i$ 而$s_i \equiv M_i^{-1} \pmod{n_i}$ $(i = 0,1,\cdots,k)$ 。

例:

$$\begin{cases} x \equiv 2 \pmod{3} \\ x \equiv 3 \pmod{5} \\ x \equiv 2 \pmod{7} \end{cases}$$

三個模數$n_1 = 3, n_2 = 5, n_3 = 7$的乘積是$n = 105$，所以$M_1 = 35, M_2 = 21, M_3 = 15$，$s_1 = 2, s_2 = 1, s_3 = 1$

$$x \equiv x_1 M_1 s_1 + x_2 M_2 s_2 + x_3 M_3 s_3 \pmod{105}$$

$$x \equiv 2 \times 70 + 3 \times 21 + 2 \times 15 \pmod{105}$$

$$x \equiv 23 \pmod{105}$$

9. 離散對數(Discrete logarithm) : 是一種基於同餘運算和原根的一種對數運算。而解$a^x \equiv b \pmod{p}$中x的問題稱之為離散對數問題，在這我們以x=$L_a(b)$表示。通常選取a為模p的原根，若a不是原根，有些b值的離散對數會沒意義。

例: $p = 11, a = 2, 2^{6+10k} \equiv 9 \pmod{11}$，通常取最小整數為離散對數，所以得到

$$L_2(9) = 6$$

也可以說是以 2 為底 6 的離散對數為 9。

第二章　離散對數問題解法

這邊將會提到四種方法來解離散對數問題，將會一一提到他們的優缺點還有演算步驟。

令a、b、p為非零整數會滿足

$$b \equiv a^x \pmod{p}$$

解$x = \log_a b$ $(x = L_a(b))$就是離散對數問題，底下我們會以這個式子來解。

第一節 演算法簡介

a. 窮舉法:

$0 \le x \le p-1$，所以將a取n次冪x=0,1,2,…，帶入$a^x \equiv b(\text{mod } p)$看是否吻合，也就是從$a^1 \equiv b(\text{mod p}), a^2 \equiv \text{b(mod p)} \dots a^x \equiv b(\text{mod p})$，每次需執行$x-1$個乘法以及比較是否吻合$x$次，會非常耗時間，而且在數字大時會非常難算，可能$a^2 \equiv b(\text{mod } p)$就是一個很大的問題了，更不用說是更高次方的解，但是計算時僅要儲存a,b 與a^x共三個數，在儲存空間上使用較少。

例:令解$7^x \equiv 3(\text{mod } 13)$

x	1	2	3	4	5	6	7	8	9	10	11	12	13
$7^x(\text{mod } 13)$	7	10	5	9	11	12	6	3	8	4	2	1	7

我們可以看到其實算到$x = 8$的時候就已經找到答案了，不過不是每

次都這麼好運，若p變大就算在算不到一半的次數就找到答案，還是得要算非常多運算，時間上也需要花非常多，如果算一題離散對數問題要花費一年以上的時間也太沒效率了。

b. 嬰步巨步法:

因窮舉法花費時間大都較長，無法快速有效的解決問題，所以得找方法改進，將步驟減少讓運算時間降低，Daniel Shanks 發現的嬰步巨步法就是用計算的儲存空間來減少計算時間，其步驟剩下$\sqrt{p-1}$，而其演算法運算方法如下:

令 $m = \lceil \sqrt{p-1} \rceil$，要解的離散對數$x = qm + r$,$0 \leq x < m$，所以

$$a^x = a^{qm+r} \equiv b(\text{mod } p) \Rightarrow (a^m)^q \equiv ba^{-r}(\text{mod } p)$$

接下來分為兩部分做計算。

嬰兒步:

令 B=$\{(ba^{-r}(\text{mod } p), r); 0 \leq r < m\}$，計算 B 中的所有數對，然後將其保留。

在 B 中找出$ba^{-r}(\text{mod } p)$=1 的數對。

若$ba^{-r}(\text{mod } p)$=1 $\Rightarrow$ $a^r \equiv b \ (\text{mod } p)$，所以$n = r$就是我們要的答案，如沒有等於 1 的那就進入巨人步。

巨人步:

算出$\delta \equiv a^m \ (\text{mod } p)$。

列出$\delta^q \pmod p, q = 1,2,3,\cdots$，然後檢驗是否出現在嬰兒步的 B 中的 $ba^{-r} \pmod p$。

如果有的話，$ba^{-r} \equiv \delta^q \equiv a^{qm} \pmod p$

$$\Longrightarrow \quad b \equiv a^{qm+r} \pmod p$$

所以$x = qm + r$就是我們要的答案。

例: 令解$7^x \equiv 3 \pmod{13}$

所以$p = 13, a = 7, b = 3$，而$m = \lceil \sqrt{p-1} \rceil = 4$，要求的離散對數 $x = qm + r$，$ba^{-r} \pmod p = 3 \times 7^{-r} \pmod{13}$

嬰兒步集合$\mathrm{B} = \{(ba^{-r} \pmod p, r); 0 \le x < m\}$

$$= \left\{ \begin{matrix} (3 \times 7^{-0} \pmod{13}, 0), (3 \times 7^{-1} \pmod{13}, 1), \\ (3 \times 7^{-2} \pmod{13}, 2), (3 \times 7^{-3} \pmod{13}, 3) \end{matrix} \right\}$$

$$= \{(3,0), (6,1), (12,2), (11,3)\}$$

沒有出現$(1, r)$所以我們將其保留。

令$\delta \equiv a^m \pmod p \equiv 7^4 \pmod{13} \equiv 9 \pmod{13}$

巨人步集合 $\{\delta^q \pmod p; 0 \le q \le m\} = \{9,3,1,9\}$

我們發現當$q = 2$時，$\delta^2 = 3 \pmod{13}$等於嬰兒步集合中$r = 0$，

$ba^{-r} \pmod p = 3 \times 7^0 \pmod{13} = 3 \pmod{13}$，因此，

$$ba^0 \equiv \delta^2 \equiv a^{2\times 3} \pmod{13}$$

$q = 2, m = 4, r = 0$，所以$x = 2 \times 4 + 0 = 8$。

$x = 3, 7^8 \equiv 3 \pmod{13}$跟我們用窮舉法算出來的結果相同，如果了

解此法的過程就能快速解出答案，比起窮舉法，嬰步巨步法的步驟少了一些，在小數字可能比較沒那麼有感。

c. 離散對數 波赫法

波利格-赫爾曼演算法(Pohlig-Hellman Algorithm)，與前面較為不同的是，能夠計算較大質數的離散對數問題，使得密文更加有保障，不過可惜的是這個方法也只適用在$p-1$的質因子較小的時候，質因子大時，就無法快速的算出答案，而其演算法運算方法如下:

1. 先將$p-1$做質因數分解，令$p-1=q_1^{r_1}\times q_2^{r_2}\times q_3^{r_3}\cdots q_n^{r_n}$，為所有質因子的次方乘積。
2. 計算每個$L_a(b)\ \mathrm{mod}(q^r)$
   a. 將所要解的x表示成

$$x\equiv x_0+x_1q+x_2q^2+\cdots+x_rq^{r-1}(\mathrm{mod}\ \ q^r)$$

   可得到

$$a^{x_0+x_1q+x_2q^2+\cdots+x_rq^{r-1}}\equiv b\ \ (\mathrm{mod}\ \ p)$$

   依序算出$x_0,x_1,x_2,\ldots,x_r$。

   b. 將$a^{x_0+x_1q+x_2q^2+\cdots+x_rq^{r-1}}\equiv b\ \ (\mathrm{mod}\ \ p)$兩邊同取$(p-1)/q$次方，會變為

$$a^{(x_0+x_1q+x_2q^2+\cdots+x_rq^{r-1})\frac{p-1}{q}}=a^{x_0\times\frac{p-1}{q}+(x_1+x_2q+\cdots+x_rq^{r-2})p-1}$$

$$\overset{\text{費馬小定理}}{=} \quad a^{x_0 \times \frac{p-1}{q}} \equiv b^{\frac{p-1}{q}} \pmod{p}$$

可以求出x_0，之後在兩邊同乘$(p-1)/q^2$ 以同樣方式解出x_1，後面以此類推解出$x_0, x_1, x_2, \ldots, x_r$，就可以把$x$解出來得到

$$x \equiv x_0 + x_1 q + x_2 q^2 + \cdots + x_r q^{r-1} \pmod{q^r}$$

c. 每一個質因子都按照 b.方式如此算出所有$L_a(b) \bmod(q^r)$接著用中國餘數定理合併所有$x \equiv x_0 + x_1 q + x_2 q^2 + \cdots + x_r q^{r-1} \pmod{q^r}$算出$x$，即為我們要的答案。

例: 令解$7^x \equiv 3 \pmod{13}$

1. $a = 7, b = 3, p = 13$，$p-1$的質因數分解為$2^2 * 3$，故有兩個質因子$q = 2, q = 3$，接著我們將其分別計算$L_7(3) \pmod 4$跟$L_7(3) \pmod 3$。

2. $q = 2$： 將x寫為$x \equiv x_0 + 2x_1 \pmod 4$，所以原方程式

$$7^x \equiv 3 \pmod{13} \Rightarrow 7^{x_0 + 2x_1} \equiv 3 \pmod{13},$$

在式子兩邊取$6 = (p-1)/2$次方，得到

$$(7^{x_0 + 2x_1})^6 \equiv 3^6 \pmod{13}$$

然後經由費馬小定理，可以把$7^{12x_1} \equiv 1 \pmod{13}$提出來，剩下

$$7^{6x_0} \equiv 3^6 = 1 \pmod{13}$$

得到$x_0 = 0$。

將$x_0 = 0$代回，原方程式變為$7^{2x_1} \equiv 3(\text{mod } 13)$，跟上面一樣，在式子兩邊取$3 = (p-1)/4$次方

$$\Rightarrow (7^{2x_1})^3 = (7^6)^{x_1} \equiv (1)^{x_1} \equiv 3^3 = 1(\text{mod } 13)$$

$$\Rightarrow (1)^{x_1} \equiv 1(\text{mod } 13)$$

得到$x_1 = 1$，所以x之質為

$$x \equiv x_0 + 2x_1 = 0 + 2 * 1 = 2 \ (\text{mod } 4)$$

$$x \equiv 2 \ (\text{mod } 4)$$

3. $q = 3$：在原同餘式兩邊取$4 = (p-1)/3$次方，得到

$$(7^4)^x \equiv 3^4 \ (\text{mod } 13)$$

$$\Rightarrow 9^x \equiv 3^2 \ (\text{mod } 13)$$

$$2x \equiv 1 \ (\text{mod } 3)$$

4. 現在我們有$x \equiv 2 \ (\text{mod } 4)$和$2x \equiv 1 \ (\text{mod } 3)$，再來根據中國餘數定理得到了$x \equiv 8 \ (\text{mod } 12)$，所以離散對數解$x = 8$，就如同上面的方法一樣。

在這方法有條理地整理跟慢慢推展的步驟下，解出答案的時間算是滿快的，只要質因數 q 不大的話，變大或是變多後，要計算個數

$$a^{k(p-1)/q}, k = 0,1,2,3 \ldots, q-1$$

快速的算完變的不太可行，這算法可能就變得不適合。

d. 指數計算法

1. 在$a^n \equiv b(\text{mod } p)$中$p$為一質數，$a$為模$p$的一個原根，令一個整數 B 且$p_1, p_2, \ldots, p_m$為小於 B 的所有質數集合，稱之為因數基底。

2. 列出幾個以a為底的$a^k(\text{mod } p)$的值，若可以寫成小於 B 之質數的乘積則留下，得到 k$\equiv \sum a_i L_a(p_i) (\text{mod } p-1)$，目標是解以$a$為底 B 中所有數的離散對數。

3. 對隨機數r，計算$ba^r(\text{mod } p)$之值，然後試著寫成小於 B 之質數的乘積，若可，我們就得到離散對數

$$x = L_a(b) \equiv -r + \sum b_i L_i(p_i) \ (\text{mod } p-1)$$

例: 令解$7^x \equiv 3(\text{mod } 13)$

$a = 7, b = 3, p = 13$，令 B=10，因數基底為{2,3,5,7}，

$$7^1 \equiv 7, 7^2 \equiv 2 * 5, 7^3 \equiv 5, 7^4 \equiv 3 * 3 \ (\text{mod } 13)$$

得到關係式

$$\begin{cases} L_7(7) \equiv 1 \ (\text{mod } 12) \\ L_7(2) + L_7(5) \equiv 2 \ (\text{mod } 12) \\ L_7(5) \equiv 3 \ (\text{mod } 12) \\ 2L_7(3) \equiv 4 \ (\text{mod } 12) \end{cases}$$

$$\Rightarrow L_7(2) \equiv 11, L_7(3) \equiv 8, L_7(5) \equiv 3, L_7(7) \equiv 1 \ (\text{mod } 12)$$

計算準備工作就完成了，在來隨機選擇一個次冪試試

$$3 * 7^1 \equiv 2^3 \ (\text{mod } 13)$$

我們要求的離散對數解就是

$$L_7(3) \equiv -1 + 3L_7(2) \equiv -1 + 33 \equiv 8 \pmod{12}$$

此方法在解得過程有些運氣成分在，最後的次冪選得好就有機會快速算出解答，不過前面準備工作完成之後，我們就可以藉由這些關係式來計算同質數同底不同數的離散對數，譬如說我們要算$L_7(10)$

$$10 * 7^1 \equiv 5 \pmod{13}$$

$$L_7(10) \equiv -1 + L_7(5) \equiv 2 \pmod{12}$$

或許在這例子中數字較小，多算幾次就跟窮舉法差不多了，不過因為能算同質數同底不同數的，或許在某些時候能夠發揮一些特殊作用。

第三章　演算法實例與討論

在這我們會取較大的數字做為範例，以及介紹演算法所用的程式

第一節:計算例子

a. 嬰步巨步法:

此法會先將要求的離散對數寫成$x = qm + r$

求 $2^x \equiv 10 \pmod{269}$

$p = 269, a = 2, b = 3$，$m = \lceil \sqrt{p-1} \rceil = 17$，$x = qm + r$，

$ba^{-r} \pmod{p} = 10 \times 2^{-r} \pmod{269}$

嬰兒步集合$\mathrm{B} = \{(ba^{-r} \pmod{p}, r); 0 \le x < m\}$

$$= \left\{ \begin{matrix} (10,0), (5,1), (137,2)(203,3)(236,4)(118,5) \\ (59,6), (164,7), (82,8), (41,9), (155,10), (212,11) \\ (106,12), (53,13), (161,14), (215,15), (242,16) \end{matrix} \right\}$$

沒有出現$(1, \mathrm{r})$

令$\delta \equiv a^m \pmod{p} \equiv 2^{17} \pmod{269} \equiv 69 \pmod{269}$

巨人步集合 $\{\delta^q \pmod{p}; 0 \le q \le m\}$

$$= \left\{ \begin{matrix} (69,1), (188,2)(60,3)(105,4)(251,5), (103,6) \\ (113,7), (265,8), (262,9), (55,10), (29,11) \\ (118,12), (72,13), (126,14), (86,15), (16,16)(28,17) \end{matrix} \right\}$$

我們發現當$q = 12$時，$\delta^{12} = 118 \pmod{269}$等於嬰兒步集合中

$r = 5$，$ba^{-r} \pmod{p} = 32^{-5} \pmod{269} = 118 \pmod{13}$，

因此，

$$ba^{-5} \equiv \delta^{12} \equiv a^{12*17} \pmod{13}$$

$q = 12, m = 17, r = 5$，所以$x = 12 * 17 + 5 = 209$。

b. 波赫法:

1. 求 $2^x \equiv 10 \pmod{269}$

$a = 2, b = 10, p = 269$，$p - 1 = 2^2 \times 67$

$\boldsymbol{q = 2}$：

$$2^x \equiv 10 (\text{mod } 269) \Rightarrow 2^{x_0 + 2x_1} \equiv 10 (\text{mod } 269)$$

$$(2^{x_0 + 2x_1})^{134} \equiv 10^{134} (\text{mod } 269)$$

$$\Rightarrow 2^{134x_0} = -1^{x_0} \equiv -1 \pmod{269}，x_0 = 1 \text{ 代回}$$

原方程式變為$2^{1+2x_1} \equiv 10 (\text{mod } 269)$

$$\Rightarrow 2^{2x_1} \equiv 10 * 2^{-1} = 5 (\text{mod } 269)$$

$$(2^{2x_1})^{67} \equiv 5^{67} (\text{mod } 269)$$

$$\Rightarrow 2^{134x_1} = -1^{x_1} \equiv 1 \pmod{269}，x_1 = 0 \text{ 代回}$$

$$x \equiv x_0 + 2x_1 = 1 + 2 * 0 = 1 \pmod 4 \quad q = 2, x \equiv 1 \pmod 4$$

$\boldsymbol{q = 67}$:

$$4 = (p - 1)/67$$

$$(2^x)^4 \equiv 10^4 \pmod{269}$$

$$\Rightarrow 16^x \equiv 47 \pmod{269}$$

得到$x = \frac{1}{42}$，$x \equiv \frac{1}{42} (\text{mod } 67) \Rightarrow x \equiv 8 (\text{mod } 67)$

所以我們現在有

$$\begin{cases} x \equiv 1 \pmod 4 \\ x \equiv 8 (\text{mod } 67) \end{cases}$$

根據中國餘數定理得到了$x \equiv 209 \pmod{269}$。

2. 求$3^x \equiv 5 \pmod{5807777}$

$a = 3, b = 5, p = 5807777$，$p - 1 = 2^5 * 13 * 23 * 607,$

$$q = 2,13,23,607$$

$q = 2$：

令$x \equiv x_0 + 2x_1 + 4x_2 + 8x_3 + 16x_4 \pmod{32}$

原方程式:

$$3^x \equiv 5 \pmod{5807777}$$

$$\Rightarrow 3^{x_0+2x_1+4x_2+8x_3+16x_4} \equiv 5 \pmod{5807777}，$$

在式子兩邊取$2903888 = (p-1)/2$次方

$$(3^{x_0+2x_1+4x_2+8x_3+16x_4})^{2903888} \equiv 5^{2903888} \pmod{5807777}$$

$$\Rightarrow 3^{2903888x_0} = -1^{x_0} \equiv -1 \pmod{5807777}$$

得到$x_0 = 1$

將$x_0 = 1$ 代回，原方程式變為

$$3^{1+2x_1+4x_2+8x_3+16x_4} \equiv 5 \pmod{5807777}$$

$$\Rightarrow 3^{2x_1+4x_2+8x_3+16x_4} \equiv 5 * 1935926 \pmod{5807777}$$

$$\Rightarrow 3^{2x_1+4x_2+8x_3+16x_4} \equiv 3871853 \pmod{5807777}$$

在式子兩邊取$1451944 = (p-1)/4$次方

$$(3^{2x_1+4x_2+8x_3+16x_4})^{1451944} \equiv 3871853^{1451944} \pmod{5807777}$$

$$\Rightarrow 3^{2903888x_1} = -1^{x_1} \equiv -1 \pmod{5807777}$$

得到$x_1 = 1$

將$x_1 = 1$代回，原方程式變為

$$3^{2+4x_2+8x_3+16x_4} \equiv 3871853 \pmod{5807777}$$

$$\Rightarrow 3^{4x_2+8x_3+16x_4} \equiv 3871853 * 1935926^2 \pmod{5807777}$$

$$\Rightarrow 3^{4x_2+8x_3+16x_4} \equiv 1720823 \pmod{5807777}$$

在式子兩邊取$725972 = (p-1)/8$次方

$$(3^{4x_2+8x_3+16x_4})^{725972} \equiv 1720823^{725972} (\text{mod } 5807777)$$

$$\Rightarrow 3^{2903888x_2} = -1^{x_2} \equiv -1 \pmod{5807777}$$

得到$x_2 = 1$

將$x_2 = 1$代回，原方程式變為

$$3^{4+8x_3+16x_4} \equiv 1720823 (\text{mod } 5807777)$$

$$\Rightarrow 3^{8x_3+16x_4} \equiv 1720823 * 1935926^4 (\text{mod } 5807777)$$

$$\Rightarrow 3^{8x_3+16x_4} \equiv 2530778 \pmod{5807777}$$

在式子兩邊取$362986 = (p-1)/16$次方

$$(3^{8x_3+16x_4})^{362986} \equiv 2530778^{362986} (\text{mod } 5807777)$$

$$\Rightarrow 3^{2903888x_3} = -1^{x_3} \equiv 1 \pmod{5807777}$$

得到$x_3 = 0$

將$x_3 = 0$代回，原方程式變為

$$3^{16x_4} \equiv 2530778 (\text{mod } 5807777)$$

在式子兩邊取$181493 = (p-1)/32$次方

$$(3^{16x_4})^{181493} \equiv 2530778^{181493} (\text{mod } 5807777)$$

$$\Rightarrow 3^{2903888x_4} = -1^{x_4} \equiv 1 \pmod{5807777}$$

得到$x_4 = 0$

$$x \equiv x_0 + 2x_1 + 4x_2 + 8x_3 + 16x_4 \pmod{32}$$

$$= 1 + 2 * 1 + 4 * 1 + 8 * 0 + 16 * 0 = 7 \pmod{32}$$

$$x \equiv 7 \pmod{32}$$

$\boldsymbol{q = 13}$：原方程式$3^x \equiv 5 \pmod{5807777}$

在式子兩邊取$446752 = (p-1)/13$次方

$$(3^x)^{446752} \equiv 5^{446752} \pmod{5807777}$$

$$\Rightarrow 196907^x \equiv 5079650 \pmod{5807777}$$

得到$x = \frac{1}{2}$

$$x \equiv \frac{1}{2} \pmod{13}$$

$$\Rightarrow x \equiv 7 \pmod{13}$$

$q = 23$：原方程式$3^x \equiv 5 \pmod{5807777}$

在式子兩邊取$252512 = (p-1)/23$次方

$$(3^x)^{252512} \equiv 5^{252512} \pmod{5807777}$$

$$\Rightarrow 1958784^x \equiv 4223347 \pmod{5807777}$$

得到$x = \frac{1}{18}$

$$x \equiv \frac{1}{18} \pmod{23}$$

$$\Rightarrow x \equiv 9 \pmod{23}$$

$q = 607$：原方程式$3^x \equiv 5 \pmod{5807777}$

在式子兩邊取$9568 = (p-1)/23$次方

$$(3^x)^{9568} \equiv 5^{9568} \pmod{5807777}$$

$$\Rightarrow 5642297^x \equiv 5647073 \pmod{5807777}$$

得到$x = \frac{1}{524}$

$$x \equiv \frac{1}{524} \pmod{607}$$

$$\Rightarrow x \equiv 117 \pmod{607}$$

$$\begin{cases} x \equiv 7 \pmod{32} \\ x \equiv 7 \pmod{13} \\ x \equiv 9 \pmod{23} \\ x \equiv 117 \pmod{607} \end{cases}$$

根據中國餘數定理得到了$x \equiv 746727 \pmod{5807776}$。

c. 指數計算法:

求$2^x \equiv 10 \pmod{269}$

$a = 2, b = 10, p = 269$，令 B=12，因數基底為{2,3,5,7,11}

$$2^{40} \equiv 196 = 2 * 2 * 7 * 7 \pmod{269}$$

$$2^{71} \equiv 33 = 3 * 11 \pmod{269}$$

$$2^{109} \equiv 3 \pmod{269}$$

$$2^{113} \equiv 48 = 2 * 2 * 2 * 2 * 3 \pmod{269}$$

$$2^{130} \equiv 84 = 2 * 2 * 3 * 7 \pmod{269}$$

$$2^{246} \equiv 245 = 5 * 7 * 7 \pmod{269}$$

$$\Rightarrow \begin{cases} 2L_2(2) + 2L_2(7) \equiv 40 \pmod{268} \\ L_2(3) + L_2(11) \equiv 71 \pmod{268} \\ L_2(3) \equiv 109 \pmod{268} \\ L_2(3) + 4L_2(2) \equiv 113 \pmod{268} \\ 2L_2(2) + L_2(3) + L_2(7) \equiv 130 \pmod{268} \\ L_2(5) + 2L_2(7) \equiv 246 \pmod{268} \end{cases}$$

$$\Rightarrow L_2(3) \equiv 109, L_2(5) \equiv 322 \equiv 64 \pmod{268}$$

$$\Rightarrow L_2(7) \equiv 19, L_2(11) \equiv -38 \equiv 230$$

選取$10 * 2^{13} \equiv 144 = 2^4 * 3^2 \pmod{269}$，所以

$$L_2(10) + 13 \equiv 4L_2(2) + 2L_2(3) \pmod{268}$$

$$\Rightarrow L_2(10) \equiv 4 * 1 + 2 * 109 - 13 \equiv 209 \pmod{268}$$

第二節:演算法

窮舉法: 條列所有x，尋找$a^x(\text{mod } p)$的值與b相同的值。

Algorithm: Exhaustive method

INPUT $a,\ b,\ p$

OUTPUT x satisfying $a^x(\text{mod } p) = b$

Step 1 Set $x_1 = a$.

Step 2 For $i = 2, 3, \dots, p$ do Steps 3-4.

Step 3 Set $x_i = \text{mod } (x_{i-1} \cdot a,\ p)$.

Step 4 If $x_i = b$ then

OUTPUT (i);

STOP.

Step 5 OUTPUT('The exhaustive method fails');

STOP.

嬰步巨步法: 將x分為$qm + r$計算，分作嬰兒步與巨人步部分找出兩部分相同之處。

Algorithm : Baby-Step Giant-step

INPUT $a,\ b,\ p$

OUTPUT x satisfying $a^x(\text{mod } p) = b$ where $x = qm + r$

Step 1 Set $m = \text{Ceiling}(\sqrt{p})$.

Step 2 Set $[H, B, \sim] = \text{gcd(a, p)}$. (where $B = a^{-1} \pmod p$)

Step 3 Set $baby_1 = b$;

$delta = a$.

Step 4 For $i = 2, 3, \dots, m$ do Steps 5-7.

Step 5 Set $baby_i = \text{mod}(\ baby_{i-1} \cdot \text{B},\ p)$.

Step 6 If $baby_i = 1$ then

OUTPUT (i);

STOP.

Step 7 Set $delta = \mathrm{mod}(\ delta \cdot a,\ p)$.

Step 8 Set $giant_1 = delta$.

Step 9 For $i = 2, 3, \ldots, m$ do Step 10.

Step 10 Set $giant_i = \mathrm{mod}(\ giant_{i-1} \cdot giant_1,\ p)$.

Step 11 $[C, Q, K] = \mathrm{intersect}(giant, baby)$.

Step 12 Set $x = Qm + (K - 1)$.

Step 13 OUTPUT(x);

STOP.

Remark: the command intersect is referred to MATLAB intersect command:

`[C,ia,ib] = intersect(___)` also returns index vectors `ia` and `ib` using any of the previous syntaxes. Generally, `C = A(ia)` and `C = B(ib)`.

波赫法: 將$p-1$質因數分解，以每個質因數找出每一個x再由中國餘數定理合併算出離散對數x。

Algorithm 3 :Pohlig-Hellman Algorithm

INPUT a, b, p

OUTPUT x satisfying $a^x (\mathrm{mod}\ p) = b$

Step 1 Set $q_1, q_2, \ldots, q_n$ and $r_1, r_2, \ldots, r_n$ to be the prime factorization of $p-1$, i.e., $p - 1 = q_1^{r_1} \times q_2^{r_2} \times q_3^{r_3} \cdots \times q_n^{r_n}$.

Step 2 Set $n = number\ \ of\ \ prime\ \ factors$;

$r_i = exponent\ \ of\ \ the\ \ i-th\ \ prime\ \ factor$;

$Qn_i = q_i^{r_i}$; (i-th factor of $p-1$)

$right\_b = b$;

$[H, B, \sim] = \gcd(a,\ p)$. (where $B = a^{-1}\ (\mathrm{mod}\ p)$)

Step 3 For $i = 1, \ldots, n$ do Steps 4-16.

Step 4 Set $right\_b = b$;

$left = \mathrm{powermod}(a, \frac{p-1}{q_i}, \mathrm{p})$

Step 5 For $j = 1, \ldots, r_i$ do Steps 6-15.

Step 6 Set $Q_j = q_i^{j-1}$;

$y = (p-1)/q_i^{\mathrm{j}}$.

$right = \mathrm{powermod}(right\_b, y, \mathrm{p})$

Step 7 If $left \neq \mathrm{right}$ and $right = 1$ then

Set $X_{\mathrm{j}} = 0$.

elseif $left = right$ then

Set $X_{\mathrm{j}} = 1$.

else do Steps 8-11.

Step 8 Set $left2_1 = left$;

$right2_1 = right$.

Step 9 For $k = 2, \ldots, Qn_i$ do Step 10.

Step 10 Set $left2_k = \mathrm{mod}(\ left2_{k-1} \cdot left,\ p)$;

$right2_k = \mathrm{mod}(\ right2_{k-1} \cdot right,\ p)$.

Step 11 Set $is = ismember(left2, right2)$; (Check whether the element of $left2$ belongs to in $right2$ or not)

$g = k$ when the first index k satisfying $is_k = 1$;

$G = K$ when the first index K satisfying if $right2_K = left2_g$;

$[\sim, GG, \sim] = \gcd(G, Qn_i)$;

$X_{\ j} = \mathrm{mod}(g \cdot GG, Qn_i)$.

Step 12 If $X_{\mathrm{j}} \neq 0$ and $\mathrm{j} \neq r_i$ do Steps 13-15.

Step 13 Set $D = \mathrm{powermod}(\ B, Q * X_{\mathrm{j}},\ p)$.

Step 14 Set $right_{inv} = \mathrm{mod}(\ right\_b \cdot D, \mathrm{p})$.

Step 15 Set $right_b = right_{inv}$.

Step 16 Set $S_i = \mathrm{dot}(X, Q)$;

$S_i = \mathrm{mod}(S_i, Qn_i)$.

Step 17 Set $Mn = (p-1)/Qn$. (element-wise division)

Step 18 Set $[\sim, Bn, \sim] = \gcd(Mn, Qn)$. (element-wise gcd computation)

Step 19 Set $CRT = Mn \cdot Bn \cdot S$; (element-wise product)

$x = \mathrm{mod}(\sum_k CRT_k, p-1)$.

Step 20 OUTPUT(x);

STOP.

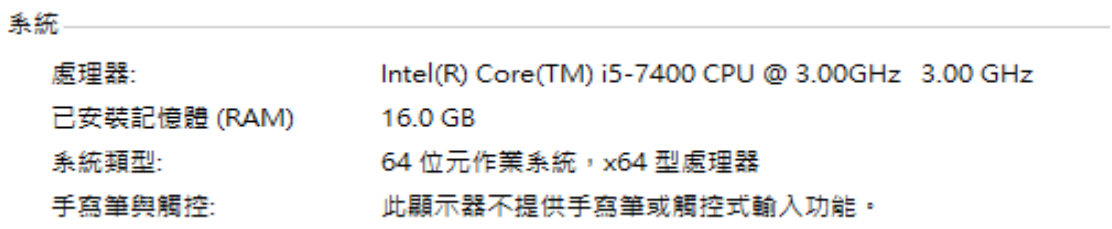
Remark: The operation `mod(a,p)` is in terms of the function $a - p \cdot \lfloor a/p \rfloor$.

Remark: The operation `powermod(a,b,p)` is referred to MATLAB powermod:

`powermod(a,b,m)` returns the modular power a^b mod `m`. Use `powermod` to compute the modular power without calculating a^b

第三節: 結果與討論

本節說明使用 MATLAB 語言 R2019a 版本，撰寫上述三種演算法的程式，程式列表在附錄內，使用的計算設備如下：



針對不同的數字，計算結果如下兩表所示。從此表來看，明顯可見嬰步巨步法的計算時間均較短，而當數字無須使用 int64 整數型態時窮舉法均較波赫法的時間快；反之，若需使用 int64 整數型態時，則波赫法遠快於窮舉法。至於為何從計算時間來看波赫法沒有嬰步巨步法有效率？本文認為是因為波赫法的計算時間會受到質因數的大小影響，如有一質因數過大計算時間便會加長。以及因為使用來計算的整數最大位數只有 10 位，但憑證認證機構應用的密碼有 2048 位元，亦即位數超過 616 位整數，實際應用上波赫法會更快。

表 1. 無須使用 int64 整數型態的離散對數，計算時間比較表

$a^x \equiv b \pmod{p}$	窮舉法	嬰步巨步	波赫法
$2^{209} \equiv 10 \pmod{269}$	0.00005s	0.00019s	0.0155s
$3^{33022} \equiv 10 \pmod{129281}$	0.0008s	0.0008s	0.0497s
$6^{532768} \equiv 10 \pmod{613231}$	0.0068s	0.0008s	0.0242s
$5^{856989} \equiv 10 \pmod{2456087}$	0.0106s	0.0012s	0.0186s
$2^{3525805} \equiv 10 \pmod{4257749}$	0.0426s	0.0011s	0.0323s
$3^{2890157} \equiv 10 \pmod{5807777}$	0.0350s	0.0012s	0.0396s
$2^{1582498} \equiv 10 \pmod{7368787}$	0.0192s	0.0012s	0.0327s
$7^{7662211} \equiv 10 \pmod{13012261}$	0.0915s	0.0013s	0.0315s
$6^{75042235} \equiv 10 \pmod{82921429}$	0.8939s	0.0019s	0.0315s
$200^{33489803} \equiv 10 \pmod{82921429}$	0.3997s	0.0019s	0.0367s
$82921100^{56200085} \equiv 10 \pmod{82921429}$	0.6695s	0.0020s	0.0360s
$6^{59283849} \equiv 1000 \pmod{82921429}$	0.7084s	0.0019s	0.0335s
$6^{7744183} \equiv 20000 \pmod{82921429}$	0.0936s	0.0028s	0.0361s
$6^{71427550} \equiv 5000000 \pmod{82921429}$	0.8496s	0.0028s	0.0335s

表 2. 需使用 int64 整數型態進行計算的離算對數，計算時間比較表

$a^x \equiv b \pmod{p}$	窮舉法	嬰步巨步	波赫法
$7^{106679392} \equiv 10 \pmod{122927041}$	558.6465s	0.1588s	0.5766s
$365^{29287712} \equiv 10 \pmod{122927041}$	153.4929s	0.1612s	0.5764s
$122921843^{49887136} \equiv 10 \pmod{122927041}$	260.9031s	0.1584s	0.5720s
$5^{291454187} \equiv 10 \pmod{295075367}$	1508.9448s	0.2494s	0.4243s
$2^{160388957} \equiv 10 \pmod{314606869}$	847.4935s	0.2560s	331.0533s
$3^{338245740} \equiv 10 \pmod{442110329}$	1712.5104s	0.3179s	0.5729s
$19^{56799910} \equiv 10 \pmod{776531401}$	292.7456s	0.3976s	0.0554s
$19^{227199640} \equiv 10000 \pmod{776531401}$	1187.7031s	0.3986s	0.0529s
$2^{1545365796} \equiv 10 \pmod{1624983827}$	7900.8790s	0.7240s	99.2586s
$2^{2224019923} \equiv 10 \pmod{2436343309}$	12309.8858s	0.8925s	21.9314s

附錄

```
clear
format rat
A = int64(3);
B = int64(10);
P = int64(442110329);

n = length(A);
x = zeros(n, 1);
time = zeros(n, 1);
N=10;
for i = 1 : n
    tic
    for j = 1 : N
        x(i) = easy(A(i),B(i),P(i));
    end
    time(i) = toc /N ;
    fprintf('Time=%10.7f for computing by Exhaustive method:
%i^%i (mod %i) = %i\n',time(i), ...
        A(i), x(i), P(i), B(i));
end
for i = 1 : n
    tic
    for j = 1 : N
        y(i) = Baby(A(i),B(i),P(i));
    end
    time(i) = toc /N ;
    fprintf('Time=%10.7f for computing by Baby-step giant-step
method: %i^%i (mod %i) = %i\n',time(i), ...
        A(i), y(i), P(i), B(i));
end
for i=1:n
    tic
    for j = 1 : N
        x(i) = polig(A(i),B(i),P(i));
    end
```

```
    time(i) = toc /N ;
    fprintf('Time=%10.7f for computing by Pohlig-Hellman algorithm:
%i^%i (mod %i) = %i\n',time(i), ...
        A(i), x(i), P(i), B(i));
end

%%
function x_ans = easy(a, b, p)
x=a;
for i=2:p
    x = x*a - p.*floor(x*a./p); % replace the mod(a,p) with a -
p.*floor(a./p)
  % x=mod( x*a, p);       % if the number of binary interger p*a <=
30 bits
  % x=mod( int64(x)*a, p);  % if the number of binary interger p*a <=
62 bits
  % x=mod( uint64(x)*a, p); % if the number of binary interger p*a <=
63 bits
  % x=mod( sym(x)*a, p);    % if the number of binary interger p*a >
64 bits
    if x==b
       x_ans = i;
       return;
    end
end
disp('The Exhaustive method fails');
end

%%
function x_ans = Baby(a, b, p)
if isinteger(p)
    m=isqrt(p);
else
    m=ceil(sqrt(p));
end
[~, B, ~] = gcd(a,p);
baby    = zeros(m,1);
baby(1) = b;
```

```
delta   = a;
for i = 2 : m
    baby(i) = baby(i-1) * B - p * floor(baby(i-1) * B /p);
    if baby(i) == 1
       x_ans   = i;
       return;
    end
    delta = delta * a - p * floor(delta * a /p);
end
giant    = zeros(m,1);
giant(1) = delta;
for i = 2 : m
    giant(i) = giant(i-1) * delta - p * floor(giant(i-1) * delta /p);
end
[~,q,r] = intersect(giant,baby);
if isempty(q)
    disp('Baby-step giant-step method fail!')
    x = 0;
else
    x = q * m +(r-1); % x=qm+r
end
x_ans = min(x);
end

%%
function x_ans = polig(a, b, p)

prime_factor = factor(p-1);
[q,d,~] = unique(prime_factor);
n = length(q);
 [~, B, ~] = gcd(a,p);
r(1:n-1) = d(2:n)-d(1:n-1);
r(n)     = length(prime_factor)-d(n)+1;
if isinteger(p)
   Qn = q .^ int64(r);
else
   Qn = q .^ r ;
end
```

```
if isinteger(p)
    left2   = int64(zeros(1, max(Qn)));
    right2  = int64(zeros(1, max(Qn)));
else
    left2   = zeros(1, max(Qn));
    right2  = zeros(1, max(Qn));
end
for i = 1 : n
    left = powermod(a,(p-1)/q(i),p);
    right_b = b;
    for j = 1 : r(i)
        Q(j)=(q(i))^(j-1);
        y=(p-1)/(q(i)^j);
        right = powermod(right_b,y,p);
        if left ~= right  &&  right == 1
            X(j) = 0;
        elseif left == right
            X(j) = 1;
        else
            left2(1)  = left;
            right2(1) = right;
            for k = 2 : Qn(i)
               left2(k)  = left   *left2(k-1)  - p * floor(left
*left2(k-1)  / p);
              right2(k) = right  *right2(k-1) - p * floor(right
*right2(k-1) / p);
            end
            is = ismember(left2,right2);
            g  = find(is==1,1);
            G  = find(right2==left2(g),1);
            [K,GG,~] = gcd(G,Qn(i));
            X(j) = g * GG - Qn(i) * floor(g* GG / Qn(i));
        end

        if X(j)~=0 && j~=r(i)
            if X(j)<0
              X(j)=X(j)+Qn(i);
```

```
            D = powermod(B,(Q(j)*X(j)),p);
        else
            D = powermod(B,(Q(j)*X(j)),p);
        end

        right_inv = right_b * D - p * floor( right_b * D /p);
        right_b  = right_inv;
    end
end
if isinteger(p)
    S(i) = sum( int64(X) .* int64(Q));
else
    S(i) = dot(X,Q);
end
S(i) = S(i) - Qn(i) * floor(S(i) / Qn(i));
clear X Q y right_b
end
Mn = (p-1)./ Qn;
[~,Bn,~] = gcd(Mn,Qn);
if isinteger(p)
    CRT = sum(int64(Mn) .* int64(Bn) .* int64(S));
else
    CRT = sum(Mn .* Bn .* S);
end
x_ans = CRT - (p-1) * floor(CRT / (p-1));
if x_ans<0
    x_ans=x_ans+p-1;
end
end
%%
function m=isqrt(p)
% square root of integer large p
m=1;
while m*m < p
    m=m+1;
end
end
```

參考文獻

[1] 沈淵源，不可能的任務 ─ 公鑰密碼傳奇，三民書局出版，2015年2月

[2] 沈淵源，密碼學之旅與MATHEMATICA同行，全華圖書出版，2006年2月

[3] 王旭正、柯宏叡、ICCL-資訊密碼暨建構實驗室，密碼學與網路安全 理論、應用與實務，博碩文化出版，2004 年 5 月

[4] William Stalling 著，賴榮樞 譯，密碼學與網路安全 第四版，台灣培生教育出版，2007 年 1 月

[5] 張智星，MATLAB 程式設計 入門篇，鈦思科技出版，2004 年 9 月