中華民國第　屆中小學科學展覽會

作品說明書

科別:數學系

組別:

作品名稱:編碼與解碼的戰役

關鍵詞: 編碼、矩陣

編號:

組員:李國豪 張世昕 郭俊毅

編碼與解碼的戰役

摘要:

書籍編號 ISBN 能利用驗證碼來判定傳輸訊息的正確性，但是在傳輸過程中若是經過竄改，就無法完整確保資料傳達目的完整性。我們以傳輸信息列遊戲規則：甲生將代數碼和驗證碼傳輸給乙生，而丙生在途中將訊息竄改一碼，並且傳給乙生，而乙生要如何辨別甲生傳輸的信息是否正確無誤?

我們利用傳輸訊息列計算式比對，更深入探討其中資料的正確性，改正訊息並且探討其錯誤訊息被修改的途徑。以質數列為運算式，將 a_1~a_8 分別乘上 3、5、7、11、13、17、19、23 的和 mod29：

a_i =0、1、2…. 、9 當 i=1、2、3…8 前八碼為編碼，後 2 碼 R_1 作為驗證用

$$I\begin{pmatrix} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \end{pmatrix}\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \text{mod29} = \begin{pmatrix} R_1 \end{pmatrix}$$

首先，我們發現此組編碼運算式在一般情形下，若前 8 碼有誤，可利用 R 檢驗知其編碼出錯，並不能找出在 a_1~a_8 是錯於哪一個碼，所以我們試著想出另一列驗證公式並嘗試辨別竄改的位置，討論過後，我們嘗試將第二式的①~⑧編碼運算式用[1,1,1,1,1,1,1,1]，因為用 1 列運算式，可以把前面 a_1~a_8 相加的總和表露出來，使竄改的 code 與[1,1,1,1,1,1,1,1]運算出來的值(設 K)和原來驗證碼(設 R_2)的差(K－R_2)，數據比對，以便找出錯誤位置。

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \text{mod29} = \begin{pmatrix} R_2 \end{pmatrix}$$

接著擴大討論範圍，驗證碼列入考慮，辨別是驗證碼錯誤或前面 code 錯誤，我們融入互補的概念，列三式：

$$\begin{pmatrix} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \\ 26 & 24 & 22 & 18 & 16 & 12 & 10 & 6 \\ 27 & 25 & 23 & 19 & 17 & 13 & 11 & 7 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \text{mod}29= \begin{pmatrix} A_1 \\ B_1 \\ C_1 \end{pmatrix}$$

B_1、C_1 為正確傳輸訊息，A_1 為獨立出的式子(以便比對 B_1 和 C_1 差值)

先將竄改過後的 code 運算後得 B_2 和 C_2，A_2 可以由 B_1 推知，比對 $\begin{pmatrix} B_1 \\ B_2 \end{pmatrix}$ 與 $\begin{pmatrix} C_1 \\ C_2 \end{pmatrix}$，若兩組有一碼對應碼不相符，則為驗證碼錯誤，若兩組中有兩碼對應碼不相符，則為 code 錯誤，並觀察(C_2-B_2)、(C_1-B_1)的值，用【$(C_2-B_2)-(C_1-B_1)$】值判定 code 被修改的變動量值，由變動的量值與 A_1、A_2 的對應關係，可得知編碼原來的 code，將之校正回正確編碼。

壹、 研究動機:

以前曾看過「終極密碼戰」、「劍魚」等電影裡，資訊隱藏與破解的遊戲，在資訊普及的現代。人類對於編碼的應用也相當廣泛，包括資訊傳輸、機密文件、卡號編碼、書籍編號．．．諸如此類，於是密碼學(cryptography)的概念也日漸重要。無非是為了使人類在資訊科技的生活中，增添一道安全的防護鎖。

例如每本書都有其不同的條碼，就像是書本的身分證一樣。一般人對於書籍編碼的認知往往局限於每本書皆有不同的編號而已，但我們小組認為其蘊藏在背後的意義極深，這引發我們的興趣。這些書籍上的條碼編碼都有一套存在的規則，通常是以 ISBN 書籍編碼，以辨識訊息的正確性質。更深入探討其中，我們假設遊戲規則：甲將 code a_1~a_8 和驗證碼傳輸給乙，而丙在途中將訊息竄改一碼，並且傳給乙，而乙要如何辨別甲傳輸的信息是否正確無誤?

我們希望能找出一套更完善的編碼公式，並依據這套編碼公式設計出能輕易檢驗密碼的方法。

貳、研究目的:

利用傳輸編碼過程中現有的條件，編輯出不只能**判斷錯誤**，並且可以**辨別**修

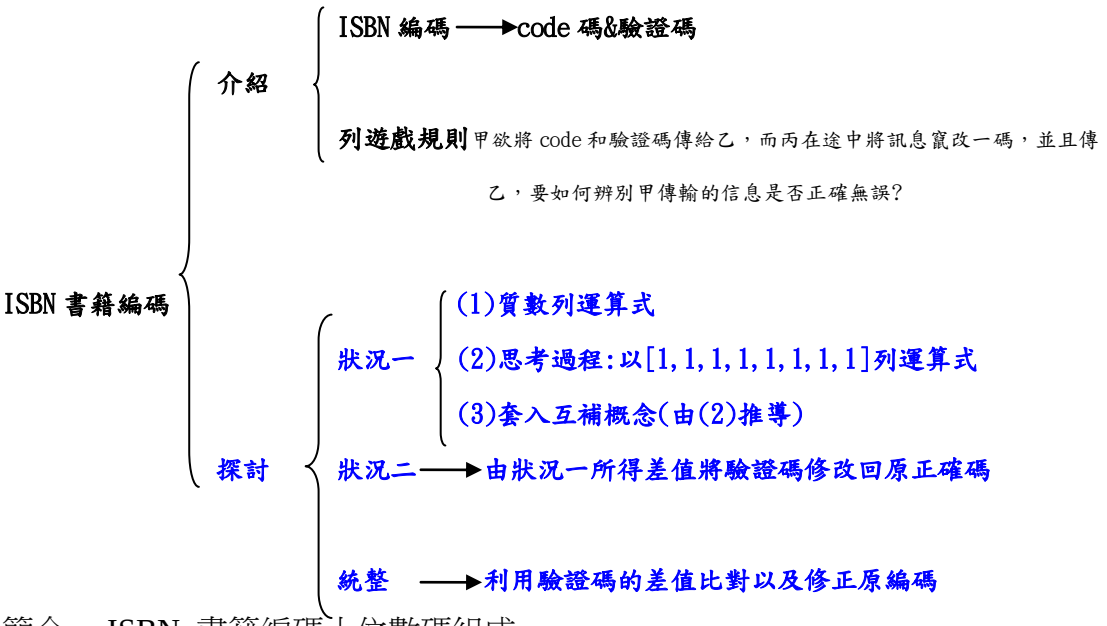
改過一碼的 code 或是驗證碼，更能辨識被修改的編碼，是**以何種途徑修改**。

參、研究設備及器材：

1. 紙
2. 筆
3. 電腦
4. C++程式語言

肆、研究過程和方法：

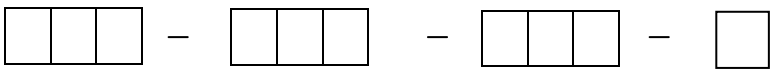
歸納研究過程&重點整理



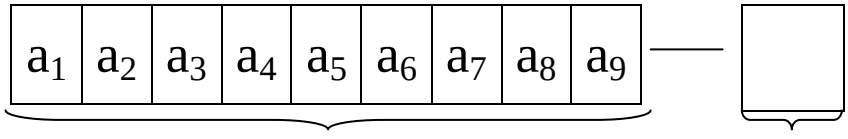
簡介: ISBN 書籍編碼十位數碼組成

結構 ⟹

群體識別號 － 出版者識別號 － 書名識別號 － 驗證碼



我們可以依序將 code 碼依序列為



code 碼　　R (驗證碼)

為了使運算過程系統化，我們可以使用矩陣乘法運算
code 碼會經過一組運算式，運算式子為：

將阿拉伯數字 1、2、…、9 依序成為一「列」，將 code 依序成為一「行」，再利用矩陣乘法運算後可得數字，假設其為 A，再以 10 為模數將 A 取成 R，R｛0、1、2、…9、｝

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \end{pmatrix} \bmod 11 = \begin{pmatrix} R \end{pmatrix}$$

矩陣乘法運算後：

$$1\times a_1+2\times a_2+3\times a_3+4\times a_4+5\times a_5+6\times a_6+7\times a_7+8\times a_8+9\times a_9 = A$$

驗證碼即為: $A \bmod 11 = \mathbf{R}$

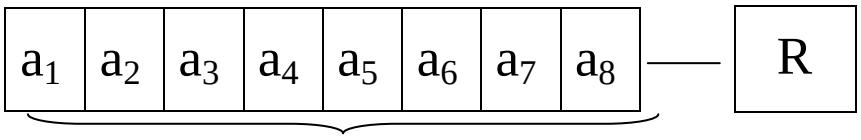
利用驗證碼來驗證 a_1~a_8 的碼是否正確無誤，並且辨別書籍的屬性、出版者、書名。若是驗證碼與 $1\times a_1+2\times a_2+3\times a_3+4\times a_4+5\times a_5+6\times a_6+7\times a_7+8\times a_8+9\times a_9$ 運算結果不符合，即可得知 a_1~a_8 為錯誤編碼，ISBN 即是利用這種方式來辨識條碼編輯的訊息。

為了解決通訊密碼的問題，我們將應用 ISBN 編碼的研究並設計一組 8 個位數+2 個位數的編碼，利用質數 3、5、7、11、13、17、19、23 依序成為一列，以便於和 code 作運算，算出的數字為 A，再以 29 為模數將 A 取成 R，R｛0、1、2、…28｝再將 R 作為驗證碼，合併成為一組 10 個位數的碼。

①~⑧為對應數字之位置

$$\begin{pmatrix} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \\ ① & ② & ③ & ④ & ⑤ & ⑥ & ⑦ & ⑧ \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \text{mod29} = \begin{pmatrix} R \end{pmatrix}$$

編碼形式為：



code 碼　　(驗證碼)

在敘述處於理想狀況下，可用於以上的例子，但在傳輸過程的過程或保存的不當造成編碼錯誤，應當如何補救與修正，以下分三種狀況做處理。

傳輸訊息的遊戲規則：甲將 code a_1~a_8 和驗證碼傳輸給乙，而丙在途中將訊息竄改一碼，並且傳給乙，而乙要如何辨別甲傳輸的信息是否正確無誤?

狀況一：在驗證碼為正確的前提下，a_1~a_8 若其中有"一個"產生錯誤如何找出 a_1~a_8 之 code 被修改過的正確位置？

驗證碼為正確的前提，只有少數驗證碼量值變動的情況下，修改一碼能迅速判定錯誤位置與數字。

為了更方便思考與運算，我們將 a_1~a_8 之 code 設為 0

並且以更改一碼為前提，將更改一碼的數字與對應項相乘之和 mod29

，並將 mod29 的值列出(詳見附件一)可以得知變更一碼得到驗證碼變動的量值相同的狀況：

單一狀況：3 項

二個相同：9 項

三個相同：9 項

四個相同：6 項

無此狀況：1 項

值只有 14 or 24 or 25 可以迅速由驗證碼得知錯誤 code 碼

由此可知，在列出一式的情況下，更改一碼，僅利用一式的驗證碼來判定錯誤碼是非常困難的，不可能完全判定錯誤的位置與數字碼，只能縮小由驗證碼得知錯誤碼的可能範圍。

必須再列第二式盡可能地縮小範圍。我們發現，在①~⑧皆為 1 的情形下，將 code 代入運算後的值會與 code 全部總合相同，因此列出第二式：

$$\begin{matrix} I \\ II \end{matrix}\begin{pmatrix} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \text{ mod}29 = \begin{pmatrix} R_1 \\ F_1 \end{pmatrix}$$

此時必須增加一點規則，因為以上討論得知，找出錯誤 code 碼的條件不足，因此在列出第二式時，必須增加線索以便找尋，因此在傳輸訊息中增加 [1,1,1,1,1,1,1,1]所運算出來的 F_1：

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	——	R_1	F_1

設原 code 經過 II 式運算為 R_2，修改後的 code 經過實際運算後為 K，由此可知被修改 code 碼總和的值比原本的 code 碼總和的值多 $K-F_1$，「$K-F_1$」也就是錯誤數字的變動量值，由此可以準確辨別錯誤數字的變動量值，在兩式比對的過程中，可利用反證法，若結論為不能完全判定數字與位置，則代表 II 式所得到的值在 I 式的行格中(詳見附件一)會找到相同的 R 值，因為在 II 式所得知 $K-R_2$ 的差在代入 I 式的值必須有異同，以便判定被竄改的位置，因此 I 式中 [3, 5, 7,11,13,17,19,23]在代入的 code 運算若是有相同的值，就表示會有 1 個以上可能的位置，即辨別數字而不能辨別位置，但事實不符，因此在列兩式 [3, 5, 7,11,13,17,19,23]和[1, 1, 1, 1, 1, 1, 1, 1]的情形下可得知 code 修改訊息。

由此可知，列兩個式子，可以得知錯誤數字碼與錯誤位置的情況：

單一狀況：27 項

二個相同：0 項

三個相同：0 項

四個相同：0 項

無此狀況：1 項

可以符合狀況一論及的問題。

列第二式的推導：

設原編碼訊息 a_1~a_8 為

1	1	1	1	1	1	3	1	——	20	10

假設前 8 碼其中某一碼

被改過後為

1	1	1	1	1	1	**1**	1	——	20	10

被改過的 code 碼經 II 式運算後為 8 可得知被竄改，

且竄改某一碼變動的量值為 8－10＝－2

檢驗原式與被改過的式子，比較後得知，前面某一碼變動量值確實為－2

接著，我們試著把 I 式與 II 式產生關聯性，並且，由餘數定理得知：

$(26\times n+3\times n) \bmod 29= 0$　且 $\mathbf{n \in Z}$

$(3\times a_1+26\times a_1+5\times a_2+24\times a_2\ldots\ldots+23\times a_8+6\times a_8) \bmod 29=0$

$\mathbf{0 \leqq a_1 \sim a_8 \leqq 9}$　　且 $\mathbf{a \in Z}$

由以上的結論可以知道[3, 5, 7,11,13,17,19,23]跟 [26,24,22,18,16,12,10, 6]算出來的驗證碼相加 mod29 會等於零，因此可能產生關聯性：

$$\begin{pmatrix} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \\ 26 & 24 & 22 & 18 & 16 & 12 & 10 & 6 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \bmod 29 = \begin{pmatrix} R_1 \\ R_2 \end{pmatrix}$$

接著把[26,24,22,18,16,12,10,6] 的①~⑧都加 1 得[27,25,23,19,17,13,11,7]式子

因 $(27\times5+3\times5)\bmod29=5$ 故可以得知他的關聯性存在，也可以來比較，

設原式 code 碼經[3,5,7,11,13,17,19,23]與 [27,25,23,19,17,13,11,7]運算後的和

mod29 的值為 M_1

也就是【R_1＋(R_2＋8) 】mod29＝ M_1

⇒ 8 mod29＝ M_1 ＝F_1

8 會因 code 碼數不同而有變動

與前八碼其中某碼被改過的 code 碼總合為 K_1，可以得知被改過某 code 碼的值比原本的 code 碼的值多 $K_1－M_1$

列第二式實際運算：

假設原編碼

1	1	1	1	1	1	1	1	11	26

前 8 碼某碼被改過之後為

1	1	1	1	1	1	1	**3**	11	26

經過驗算後，得知某一碼變動的量值為 10－(11+26)mod29=2

經原式跟被改過的式子比較後，會發現變動的量值確實為 2

整合：

接著，我們將以上運算式整合起來，以便驗證起來更加方便

被改過 code 碼後的運算式：

$$\begin{matrix} I \\ II \\ III \end{matrix}\begin{pmatrix} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \\ 26 & 24 & 22 & 18 & 16 & 12 & 10 & 6 \\ 27 & 25 & 23 & 19 & 17 & 13 & 11 & 7 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \text{mod } 29 = \begin{pmatrix} A_1 \\ B_1 \\ C_1 \end{pmatrix}$$

字面上顯示為 a_1、a_2、a_3、a_4、a_5、a_6、a_7、a_8、B_1、C_1 其中 a_1~a_8 為 code 碼

這樣可以準確找出前八碼中一碼錯的的位置，並且改回正確的位置。且把 I 式當作驗證用，並且得知 $C_1－B_1$ 為某碼修改過的值，且(A_1+B_1)mod29=0

將改過的 code 碼重新運算：

$$\begin{matrix} \\ I \\ II \\ III \end{matrix}\begin{pmatrix} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \\ 26 & 24 & 22 & 18 & 16 & 12 & 10 & 6 \\ 27 & 25 & 23 & 19 & 17 & 13 & 11 & 7 \end{pmatrix}\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \bmod 29 = \begin{pmatrix} A_2 \\ B_2 \\ C_2 \end{pmatrix}$$

利用 C_2-B_2 可以得知修改過後的數字總和量值，

而 C_1-B_1 可以知道原本編碼應有的數字總和量值

E∈N

由 $A_1+E【(C_2-B_2)-(C_1-B_1)】mod29=A_2$

$【(C_2-B_2)-(C_1-B_1)】$得知修改過後變動的量值，

E 必定為[3, 5, 7,11,13,17,19,23]的其中一個數字，

為了減少運算時間，可以參照附件一的表格，並且 E 可以得知①~⑧對應的 a_1 ~a_8 位置，知道被改過的位置，而知道量值和位置後，也代表可以從中發現錯誤的 code 碼為哪一個。

狀況一的實際運算：

設前八碼某碼被改過為

3	1	1	1	1	1	1	1	——	18 | 26

得知某碼被修改多 2

得知 C_1 應為 11，可是經上面 code 碼運算之後發現為 17

E∈ {3, 5, 7,11,13,17,19,23**}**

並且得知(17－E×2)mod29=11

(可以列式子運算或是比對附件一的表格)

E＝3(得知錯誤位置)

故得知原式為

1	1	1	1	1	1	1	1	——	18 | 26

狀況一的結論：

利用 C_2-B_2 的差得知修改過後的量值：

E∈N

由 $A_1+E【(C_2-B_2)-(C_1-B_1)】mod29=A_2$

$【(C_2-B_2)-(C_1-B_1)】$得知修改過後變動的量值，

E 必定為[3, 5, 7,11,13,17,19,23]的其中一個數字，並且 E 可以得知①~⑧對應的 a_1 ~a_8 位置，知道被改過的位置。

狀況二:如果擴大範圍將驗證碼列入考慮，應如何辨別是驗證碼錯誤或前面 code 錯誤?

在上述的運算式中，我們先假設驗證碼皆為正確，因此擴大範圍將驗證碼列入考慮，因此，我們深入研究發現，在改一碼的前提下，也可以順利找出其錯誤碼，因為，如果算過之後，只有其中一個驗證碼不符合，可以得知後面的驗證碼錯誤，如果後面兩碼驗證碼都不符合，表示是前面 8 碼錯誤。

實際運算：

1	1	1	1	1	1	1	1	——	18	26

為原本正確的編碼
我們試圖去修改其中一碼使其錯誤

(1)後面其中一碼的狀況：

1	1	1	1	1	1	1	1	——	19	26

經過 1×3+1×5+1×7…算過後得驗證用碼為 11
得知 19 碼的位置是錯的，正確情況下應是 18
因為 19+11 mod 29≠0　18+11 mod 29＝0
又因 26 碼這的位置是正確的又只有一碼被修改的可能
1+1+1+1+1+1+1+1 mod 29= 8
故得知驗證碼 19 的部分是錯的正確得應該是 18

(2)前面其中一碼的狀況：

1	1	1	1	1	1	**3**	1	——	18	26

經過 1×3＋1×5＋1×7…算過後得驗證碼為 20
得知 18 可能是錯的，但 26 也是錯的，因此可以推知前面其中一碼錯誤，
經過推算得知，前面全部 code 碼相加應為 8＋n×29　　且 **n∈ Z**
前面碼相加得 10 故推知其中一碼減 2 就是原編碼
因

1	1	1	1	1	1	**3**	1	——	18	26

前面 code 碼只有一碼為少 2、大於 0 的，故得知是 3 的位置錯誤，又得知應該

要少 2 故推知原碼為

1	1	1	1	1	1	1	1	——	18	26

每個都分別向下修 2 運算之後，得到符合原來驗證碼，就是正確的編碼。

E∈ {3, 5, 7,11,13,17,19,23**}**

➔20－2E　mod29＝11

(可以利用附件一表格比對)

➔E＝19

將 E 重新代入

➔20－38 mod29＝11

➔－18 mod 29＝11

算出來的驗證碼，要是 0~28 不符合，必須利用餘數定理調整回來。

狀況二的結論：

比對

$$\begin{pmatrix} \mathbf{B_1} \\ \mathbf{C_1} \end{pmatrix} 、 \begin{pmatrix} \mathbf{B_2} \\ \mathbf{C_2} \end{pmatrix}$$

若是 B_1 或 C_1 只有一碼不等於對應位置的 B_2、C_2

表示錯誤的碼在後面，若是錯在前面的 code 碼，在後面的 B_1、C_1 一定不會和對應的 B_2、C_2 相同，若是相同，則此一編碼就沒有錯誤。(因為改前面的碼，相對來說後面並不會被修改，改後面一碼就能拿兩碼比對。) 若是不同，就能以 B_2、C_2 為基準，將更改的 B_1、C_1 修改回來。

附件一：

I 式 mod29

$$\begin{matrix} \\ \text{I} \\ \text{II} \\ \text{III} \end{matrix}\begin{pmatrix} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \\ 26 & 24 & 22 & 18 & 16 & 12 & 10 & 6 \\ 27 & 25 & 23 & 19 & 17 & 13 & 11 & 7 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \text{mod } 29=R$$

$(3\times a_1 +5\times a_2+\ldots\ldots+19\times a_7+23\times a_8)$ mod29=R

代入 code⇒	1	2	3	4	5	6	7	8	9
① 3 ⇒	3	6	9	12	15	18	21	24	27
② 5 ⇒	5	10	15	20	25	1	6	11	16
③ 7 ⇒	7	14	21	28	6	13	20	27	5
④ 11 ⇒	11	22	4	15	26	8	19	1	12
⑤ 13 ⇒	13	26	10	23	7	20	4	17	1
⑥ 17 ⇒	17	5	22	10	27	15	3	20	8
⑦ 19 ⇒	19	9	28	18	8	27	17	7	26
⑧ 23 ⇒	23	17	11	5	28	22	16	10	4

統計 R 值對應①~⑧出現的位置：

1：②④⑤	**11**：②④⑧	**21**：①③
2：無	**12**：①④	**22**：④⑥⑧
3：①⑥	**13**：③⑤	**23**：⑤⑧
4：④⑤⑧	**14**：③	**24**：①
5：②③⑥⑧	**15**：①②④⑥	**25**：②
6：①②③	**16**：②⑧	**26**：④⑤⑦
7：③⑤⑦	**17**：⑤⑥⑦⑧	**27**：①③⑥⑦
8：④⑥⑦	**18**：①⑦	**28**：③⑦⑧
9：①⑦	**19**：④⑦	
10：②⑤⑥⑧	**20**：②③⑤⑥	

附件二：

II 式 mod29

$$\begin{array}{c} \\ \text{I} \\ \text{II} \\ \text{III} \end{array}\begin{pmatrix} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \\ 26 & 24 & 22 & 18 & 16 & 12 & 10 & 6 \\ 27 & 25 & 23 & 19 & 17 & 13 & 11 & 7 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \text{mod } 29=R$$

$(26\times a_1 +24\times a_2+\ldots\ldots+10\times a_7+6\times a_8)$ mod29=R

代入 **code** ⟹	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
① **26** ⟹	26	23	20	17	14	11	8	5	2
② **24** ⟹	24	19	14	9	4	28	23	18	13
③ **22** ⟹	22	15	8	1	23	16	9	2	24
④ **18** ⟹	18	7	25	14	3	21	10	28	17
⑤ **16** ⟹	16	3	19	6	22	9	25	12	28
⑥ **12** ⟹	12	24	7	19	2	14	20	9	21
⑦ **10** ⟹	10	20	1	11	21	2	12	22	3
⑧ **6** ⟹	6	12	18	24	1	7	13	19	25

統計 R 值對應①~⑧出現的位置：

1：③⑦⑧	**11**：①⑦	**21**：④⑥⑦
2：①③⑥⑦	**12**：⑤⑥⑦⑧	**22**：③⑤⑦
3：④⑤⑦	**13**：②⑧	**23**：①②③
4：②	**14**：①②④⑥	**24**：②③⑥⑧
5：①	**15**：③	**25**：④⑤⑧
6：⑤⑧	**16**：③⑤	**26**：①⑥
7：④⑥⑧	**17**：①④	**27**：無
8：①③	**18**：②④⑧	**28**：②④⑤
9：②③⑤⑥	**19**：②⑤⑥⑧	
10：④⑦	**20**：①⑦	

附件三：

III 式 mod29

$$\begin{array}{c} \\ \text{I} \\ \text{II} \\ \text{III} \end{array}\begin{pmatrix} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \\ 26 & 24 & 22 & 18 & 16 & 12 & 10 & 6 \\ 27 & 25 & 23 & 19 & 17 & 13 & 11 & 7 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \text{mod } 29=R$$

$(27\times a_1 +25\times a_2+\ldots\ldots+11\times a_7+7\times a_8)$ mod29=R

代入 **code** ⟹	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
① **27** ⟹	27	25	23	21	19	17	15	13	11
② **25** ⟹	25	21	17	13	9	5	1	26	22
③ **23** ⟹	23	17	11	5	28	22	16	10	4
④ **19** ⟹	19	9	28	18	8	27	17	7	26
⑤ **17** ⟹	17	5	22	10	27	15	3	20	8
⑥ **13** ⟹	13	26	10	23	7	20	4	17	1
⑦ **11** ⟹	11	22	4	15	26	8	19	1	12
⑧ **7** ⟹	7	14	21	28	6	13	20	27	5

統計 R 值對應①~⑧出現的位置：

1：②⑥⑦	**11**：①③⑦	**21**：①②⑧
2：無	**12**：⑦	**22**：②③⑤⑦
3：⑤	**13**：①②⑥⑧	**23**：①③⑥
4：③⑥⑦	**14**：⑧	**24**：無
5：②③⑤⑧	**15**：①⑤⑦	**25**：①②
6：⑧	**16**：③	**26**：②④⑥⑦
7：④⑥⑧	**17**：①②③④⑤⑥	**27**：①④⑤⑧
8：④⑤⑦	**18**：④	**28**：③④⑧
9：②④	**19**：①④⑦	
10：③⑤⑥	**20**：⑤⑥⑧	

附件四:

```
#include<stdio.h>
main()
{
int
a[8],q,x1=0,x2=0,x3=0,x4=0,c1,c2,c3,c4,b=0,p1[8]={1,1,1,1,1,1,1,1},p2[8]={3,5,7,11,13,17,19,23},p3
[8]={26,24,22,18,16,12,10,6},p4[8]={27,25,23,19,17,13,11, 7};

printf("請輸入 a1~a8 的 code 碼,本程式會依序產生\n");
printf("[ 1, 1, 1, 1, 1, 1, 1, 1]運算式的驗證碼\n");
printf("[ 3, 5, 7,11,13,17,19,23]運算式的驗證碼\n");
printf("[26,24,22,18,16,12,10, 6]運算式的驗證碼\n");
printf("[27,25,23,19,17,13,11, 7]運算式的驗證碼\n");
for(q=0;q<8;q++)
{b=b+1;
printf("code(%d)=>",b);
scanf("%d",&a[q]);}

for(q=0;q<8;q++)
{x1+=a[q]*p1[q];
c1=x1 % 29;}
for(q=0;q<8;q++)
{x2+=a[q]*p2[q];
c2=x2 % 29;}
for(q=0;q<8;q++)
{x3+=a[q]*p3[q];
c3=x3 % 29;}
for(q=0;q<8;q++)
{x4+=a[q]*p4[q];
c4=x4 % 29;}
printf("以下為您輸入的 code 碼經過運算後得到的驗證碼\n",a[q]);
printf("[ 1, 1, 1, 1, 1, 1, 1, 1]運算式的驗證碼為%d\n",c1);
for(q=0;q<8;q++)
{printf("%d×%d",a[q],p1[q]);
if (q<7)
printf("+");
else
```

```
printf("=%d\n",c1);}
printf("[ 3, 5, 7,11,13,17,19,23]運算式的驗證碼為%d\n",c2);
for(q=0;q<8;q++)
{printf("%d×%d",a[q],p2[q]);
if (q<7)
printf("+");
else
printf("=%d\n",c2);}
printf("[26,24,22,18,16,12,10, 6]運算式的驗證碼為%d\n",c3);
for(q=0;q<8;q++)
{printf("%d×%d",a[q],p3[q]);
if (q<7)
printf("+");
else
printf("=%d\n",c3);}
printf("[27,25,23,19,17,13,11, 7]運算式的驗證碼為%d\n",c4);
for(q=0;q<8;q++)
{printf("%d×%d",a[q],p4[q]);
if (q<7)
printf("+");
else
printf("=%d\n",c4);
}
system("pause");}
```

附件五:

用 C++實際運算矩陣[3 5 7 11 13 17 19 23]
([a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8]可以設為 I 式、II 式、III 式)

```
#include<stdio.h>
main()
{
FILE *A;
A=fopen("1.txt","r+");
int a[8],c,p[8]={3,5,7,11,13,17,19,23};
for(a[0]=0;a[0]<10;a[0]++)
for(a[1]=0;a[1]<10;a[1]++)
for(a[2]=0;a[2]<10;a[2]++)
for(a[3]=0;a[3]<10;a[3]++)
for(a[4]=0;a[4]<10;a[4]++)
for(a[5]=0;a[5]<10;a[5]++)
for(a[6]=0;a[6]<10;a[6]++)
for(a[7]=0;a[7]<10;a[7]++)
{c=(a[0]*p[0]+a[1]*p[1]+p[2]*a[2]+p[3]*a[3]+p[4]*a[4]+p[5]*a[5]+p[6]*a[6]+p[7]*
a[7])%29;
fprintf(A,"%d %d %d %d %d %d %d %d
%d\n",a[0],a[1],a[2],a[3],a[4],a[5],a[6],a[7],c);}
fclose(A);
system("pause");}
```

附件六

```
#include<stdio.h>
main()
{
int a[10],c,d,b,i=0,num=0,num2=0;
printf("程式利用 26,24,22,18,16,12,10,6  &  +1 的式子運算\n");
printf("請輸入傳輸訊息,程式會顯示正確訊息\n");
printf("(1)~(8)為 code 碼\n");
printf("(9)和(10)為 26,24,22,18,16,12,10,6  &  +1 的驗證碼\n");
for(d=0;d<10;d++)
{num=num+1;
printf("(%d)=>",num);
scanf("%d",&a[d]);}
c=(a[0]*3+a[1]*5+a[2]*7+a[3]*11+a[4]*13+a[5]*17+a[6]*19+a[7]*23)%29;
if((c+a[8])%29==0&&(c+a[9])%29==(a[1]+a[2]+a[0]+a[3]+a[4]+a[5]+a[6]+a[7])%29)
printf("驗證輸入正確\n");
else if((c+a[8])%29==0||(c+a[9])%29==(a[1]+a[2]+a[0]+a[3]+a[4]+a[5]+a[6]+a[7])%29)
{
printf("後兩碼驗證碼其中一碼錯誤開始驗證\n");
if((c+a[8])%29!=0)
{a[8]=29-c;
printf("第一個驗證碼錯\n");}
if((c+a[9])%29!=(a[1]+a[2]+a[0]+a[3]+a[4]+a[5]+a[6]+a[7])%29)
{a[9]=(29-c+a[1]+a[2]+a[0]+a[3]+a[4]+a[5]+a[6]+a[7])%29;
printf("第二個驗證碼錯\n");}
}
else
{
printf("前面八碼中有錯開始驗證\n");
b=a[1]+a[2]+a[0]+a[3]+a[4]+a[5]+a[6]+a[7]-a[9]+a[8];
for(;b>9||b<-9;)
{
   if(b>9)
   b=b-29;
   if(b<-9)
   b=b+29;
}
```

```
for(d=0;d<10;d++)
{
a[d]=a[d]-b;
if(a[d]>-1)
{
c=(a[0]*3+a[1]*5+a[2]*7+a[3]*11+a[4]*13+a[5]*17+a[6]*19+a[7]*23)%29;
printf("開始驗證第%d 碼\n",d+1);
if((c+a[8])%29==0&&(c+a[9])%29==(a[1]+a[2]+a[0]+a[3]+a[4]+a[5]+a[6]+a[7])%29)
{
   i=1;
   printf("找出第%d 碼錯誤\n",d+1);
   break;
}
}
if(i==1)
  break;
else
a[d]=a[d]+b;
printf("第%d 碼為正確碼，",d+1);
}
}
printf("正確碼為\n");
for(d=0;d<10;d++)
```

伍、研究結果：

在 code 碼或驗證碼都有可能錯誤的情況下
利用：

[3, 5, 7,11,13,17,19,23]
[26,24,22,18,16,12,10, 6]
[27,25,23,19,17,13,11, 7]

三列式子判斷錯誤的數字碼，[3, 5, 7,11,13,17,19,23]這列則用於檢驗
至於[26,24,22,18,16,12,10, 6]、[27,25,23,19,17,13,11, 7]列出驗證碼
做驗證碼的比對運算

此為傳輸信息
字面上顯示信息為：

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	——	B_1	C_1

A_1 可以由 $A_1+B_1=29$ 得知

$$\begin{pmatrix} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \\ 26 & 24 & 22 & 18 & 16 & 12 & 10 & 6 \\ 27 & 25 & 23 & 19 & 17 & 13 & 11 & 7 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \bmod 29 = \begin{pmatrix} A_1 \\ B_1 \\ C_1 \end{pmatrix}$$

此為 a_1~a_8 運算後得到的驗證信息

$$\begin{pmatrix} 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 \\ 26 & 24 & 22 & 18 & 16 & 12 & 10 & 6 \\ 27 & 25 & 23 & 19 & 17 & 13 & 11 & 7 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \bmod 29 = \begin{pmatrix} A_2 \\ B_2 \\ C_2 \end{pmatrix}$$

(1)前面一碼錯誤的情況下：

利用 C_2-B_2 的差得知修改過後的量值：

$E \in N$

由 $A_1+ E【(C_2-B_2)-(C_1-B_1)】\bmod 29=A_2$

$【(C_2-B_2)-(C_1-B_1)】$ 得知修改過後變動的量值，

E 必定為[3, 5, 7,11,13,17,19,23]的其中一個數字，並且 E 可以得知①~⑧對應的 a_1 ~a_8 位置，知道被改過的位置。

(2)後面一碼錯誤的情況下：

比對

$$\begin{pmatrix} \mathbf{B_1} \\ \mathbf{C_1} \end{pmatrix} 、 \begin{pmatrix} \mathbf{B_2} \\ \mathbf{C_2} \end{pmatrix}$$

若是 B_1 或 C_1 只有一碼不等於對應位置的 B_2、C_2
表示錯誤的碼在後面，若是錯在前面的 code 碼，在後面的 B_1、C_1 一定不會和對應的 B_2、C_2 相同，若是相同，則此一編碼就沒有錯誤。（因為改前面的碼，相對來說後面並不會被修改，改後面一碼就能拿兩碼比對。）若是不同，就能以 B_2、C_2 為基準，將更改的 B_1、C_1 修改回來。

整合後

將編碼公式化：

(1)前面一碼錯誤的情況下：

$$\begin{pmatrix} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & k_7 & k_8 \\ b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 \\ c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \text{ mod P} = \begin{pmatrix} A_1 \\ B_1 \\ C_1 \end{pmatrix}$$

$$\begin{pmatrix} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & k_7 & k_8 \\ b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 \\ c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \text{ mod P} = \begin{pmatrix} A_2 \\ B_2 \\ C_2 \end{pmatrix}$$

$k_1 \sim k_8 \in \mathbf{N}$、$b_1 \sim b_8 \in \mathbf{N}$、$c_1 \sim c_8 \in \mathbf{N}$　$E \in \mathbf{N}$

P 滿足$(P, k_1 \sim k_8)=1$ 、$(P, b_1 \sim b_8)=1$ 、$(P, c_1 \sim c_8)=1$

而且 $k_1+b_1=k_2+b_2=k_3+b_3=k_4+b_4=k_5+b_5=k_6+b_6=k_7+b_7=k_8+b_8=P$

又 $b_1+1=c_1$，$b_2+1=c_2$，$b_3+1=c_3$，$b_4+1=c_4$，$b_5+1=c_5$，$b_6+1=c_6$，$b_7+1=c_7$，$b_8+1=c_8$

由 $A_1+ E【(C_2-B_2)-(C_1-B_1)】\text{mod}29=A_2$

得知 E 為 $k_1 \sim k_8$ 的其中一個數字，並且 E 可以得知①~⑧對應的 $a_1 \sim a_8$ 位置，知道被改過的位置。

$【(C_2-B_2)-(C_1-B_1)】$得知修改數字碼的變動量值

(2)後面一碼錯誤的情況下：

若是 B_1 或 C_1 只有一碼不等於對應位置的 B_2、C_2， 表示錯誤的碼在後面，就能以 B_2、C_2 為基準，將更改的 B_1、C_1 修改回來。

我們能利用編碼信息，在被改一碼的情況下，準確的改回原本的數字碼，但是我們不能分辨他實際上錯的碼數量，也沒有從錯兩碼以上的碼，改回原本正確的碼的可能。

陸、結論:

我們可以藉由這組編碼方式，來補足之前書籍編碼所無法找出錯誤碼的情況。

科學應用與未來展望：

整合上述研究結果，此公式在日常生活中也能夠加以應用，對於條碼的編列也有很大的助益，例如：身分證編號、問卷調查表的編號歸類、圖書管理、物料編碼、產品編號。可以將我們做的結論以編碼的方式呈現，以產品編號舉例：

a_1	a_2	—	a_3	a_4	a_5	—	a_6	a_7	a_8	—	B_1	C_1

a_1~a_2、a_3~a_5、a_6~a_8、B_1、C_1 可以分別代表：出版國家或語言代碼、出版商代碼、書籍分配的號碼、檢查碼 B、檢查碼 C，由研究的結論，書商在檢查編列條碼時，不僅可以得知是否其中有錯誤的一碼，還能利用後兩驗證碼檢查錯誤的數字碼，可用於辨別資料的正確性，並且得知資料原數字碼。

柒、參考文獻：

網站

http://zh.wikipedia.org/wiki/%E7%9F%A9%E9%99%A3%E4%B9%98%E6%B3%95

http://zh.wikipedia.org/w/index.php?title=ISBN&variant=zh-tw

書籍

數學小魔女 p63~p117

近代密碼學與其應用 p2-21~p2-24、p3-80~p3-82、p5-2~p5-15