บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้ จะให้คำจำกัดความและทฤษฎีของจำนวนที่เกี่ยวข้อง ซึ่งประกอบด้วย ระบบจำนวนเชิงซ้อนของเพนนีย์ ระบบจำนวนซ้ำซ้อน ค่าน้ำหนักเลขคณิต การคำนวนแบบหน้าต่างเลื่อนไหล และการคำนวณแบบเชื่อมตรง

2.1 ระบบจำนวนเชิงซ้อนของเพนนีย์

ในปี 1964 วอทเทอร์ เพนนีย์ [1] ได้นำเสนอระบบจำนวนเชิงซ้อนของเพนนีย์ขึ้นมา โดยใช้ฐาน $\beta = -1 + j$ และชุดตัวเลข (digit set) D = {0,1} เพราะฐานของระบบจำนวนเป็นจำนวนเชิงซ้อน ดังนั้นส่วนจริงและส่วนจินตภาพจึงสามารถเขียนให้อยู่ในรูปแบบแทนจำนวนชุดเดียวกันได้ และจำนวนเชิงซ้อน x ใดๆ สามารถเขียนอยู่ในรูปแบบแทนจำนวนของเพนนีย์ได้ดังนี้

$$x = (d_n d_{n-1} d_{n-2} \ldots d_0 . d_{-1} \ldots)_{1+j}$$

และค่าเชิงตัวเลขของ x คือ $\| x \| = \sum_{i=n}^{\infty} d_i$ โดยที่ $d_i \in \{0,1\}$

ตัวอย่างที่ 2.1 *จำนวนเชิงซ้อน* $4 + j$ *สามารถเขียนให้อยู่ในรูปแบบของจำนวนเชิงซ้อนของเพนนีย์ได้ทั้งนี้เพราะ*

$$\begin{aligned} -4 + j &= 1\times(-4) + 1\times(-1+j) + 1 \\ &= 1\times(-1+j)^4 + 1\times(-1+j) + 1\times(-1+j)^0 \end{aligned}$$

รูปแบบแทนจำนวนของ $4 + j$ คือ $(10011)_{1+j}$ □

2.2 ระบบจำนวนซ้ำซ้อน

ระบบจำนวนซ้ำซ้อน (redundant number system) ถูกพัฒนาขึ้นเพื่อเพิ่มประสิทธิภาพในด้านความเร็วในการคำนวณ โดยระบบจำนวนซ้ำซ้อนที่เป็นที่นิยมกันมาก คือ ระบบแทนจำนวนซ้ำซ้อนแบบมีเครื่องหมาย (redundant signed-digit number representation system) ซึ่งได้นำเสนอครั้งแรกโดย อวีเซียนีส (Avizienis) ในปี ค.ศ. 1961 [2]

โดยระบบจำนวนซ้ำซ้อนหมายถึง ระบบจำนวนที่มีอย่างน้อยหนึ่งจำนวน ที่สามารถเขียนด้วยรูปแบบแทนจำนวนแบบจำกัดได้มากกว่าหนึ่งรูปแบบ

ระบบจำนวนเชิงซ้อนของเพนนีย์ สามารถพัฒนามาเป็นระบบจำนวนซ้ำซ้อนได้โดยใช้ชุดตัวเลขโดยใช้ชุดตัวเลข $\{a, a+1, \ldots, b\}$ ภายใต้ข้อจำกัดที่ a เป็นจำนวนเต็มลบ และ b เป็นจำนวนเต็ม และ

$$| b - a + 1 | > 2$$

ตัวอย่างเช่น ในงานวิจัยของเซย์นี่และเดรชมุช (Zaini and Deshmukh) [3] ได้นำเสนอให้ใช้ชุดตัวเลขสมมาตร (symmetric digit set) นั่นคือชุดตัวเลข $\{-a, -a+1, \ldots, a\}$ โดย a เป็นจำนวนเต็ม และระบบนี้จะไม่เกิดกระจายของตัวทดในการทำการบวก

ตัวอย่างที่ 2.2 *จำนวนเชิงซ้อน $4+j$ สามารถเขียนให้อยู่ในรูปของจำนวนเชิงซ้อนของเพนนีย์แบบซ้ำซ้อนได้มากกว่าหนึ่งรูปแบบสำหรับชุดตัวเลข $\{-1, 0, 1\}$ ทั้งนี้เพราะ*

$$\begin{aligned} -4+i &= 1\times(-4) + (-1)\times(-2j) + (-1)\times(-1+j) + (-1)\times 1 \\ &= 1\times(-1+j)^4 + (-1)\times(-1+j)^2 + (-1)\times(-1+j) + (-1)\times(-1+j)^0 \end{aligned}$$

รูปแบบแทนจำนวนของ $4+j$ คือ $(1\bar{1}0\bar{1}\bar{1})_{1+j}$ และจากตัวอย่างที่ 1 แสดงให้เห็นว่า $-4+j = (10011)_{1+j}$ หมายความว่าสามารถเขียนได้อย่างน้อยสองรูปแบบที่ต่างกัน □

หมายเหตุ ตัวเลข $-a$ นิยมเขียนแทนด้วยสัญลักษณ์ $\bar{a}$

2.3 ค่าน้ำหนักเลขคณิต

ในระบบจำนวนซ้ำซ้อนจำนวนหนึ่งสามารถมีรูปแบบแทนจำนวนได้มากกว่าหนึ่งรูปแบบ โดยในแต่ละรูปแบบก็อาจมีค่าน้ำหนักที่แตกต่างกันไปได้ ค่าน้ำหนักเลขคณิต (Arithmetic weight) w(n) หมายถึง จำนวนของตัวเลขที่ไม่เป็นศูนย์ ในการคำนวณบางประเภท เช่น การคูณ ค่าน้ำหนักเลขคณิตจะเข้ามามีบทบาทสำคัญมากกับความเร็วที่ใช้ในการคำนวณ เพราะถ้าจำนวนนั้น มีค่าน้ำหนักน้อยย่อมมีขั้นตอนกระบวนการทำงานน้อยกว่าจำนวนที่มีค่าน้ำหนักมาก ดังนั้น ปัญหาการหารูปแบบแทนจำนวนที่มีค่าน้ำหนักที่น้อยลงจึงเป็นปัญหาที่น่าสนใจปัญหาหนึ่ง

ตัวอย่างที่ 2.3 *ในระบบเลขฐานของเพนนีย์ $(1+j)$ และชุดตัวเลขแบบมีเครื่องหมาย $\{\bar{1},0,1\}$ รูปแบบแทนจำนวนของจำนวน $(-31+2j)_{(-1+j)} = 01111101000101_{(-1+j)}$ สามารถแสดงได้ดังตารางที่ 1*

ตารางที่ 2.1 ค่าน้ำหนักของ -31+22j ในระบบฐานของเพนนีย์แบบซ้ำซ้อน

รูปแบบ	รูปแบบแทนจำนวน	ค่าน้ำหนัก
1	001111101000101	8
2	000001$\bar{1}$01110$\bar{1}$01	7
3	000001$\bar{1}$00$\bar{1}\bar{1}$0$\bar{1}$01	6
4	000001$\bar{1}$01000101	5

จะเห็นได้ว่าตัวเลขทั้งสี่แบบมีค่าเท่ากัน แต่ละแบบมีรูปแบบที่แตกต่างกันและชุดตัวเลขแบบที่สอง นั้นมีค่าน้ำหนักน้อยที่สุด คือ 5 เมื่อเปรียบเทียบกับรูปแบบอื่น □

จากตัวอย่างที่ 2.3 จะเห็นได้ว่ารูปแบบแทนจำนวนทั้งสี่แบบในแต่ละตัวอย่างนั้นแสดงจำนวนที่มีค่าเท่ากัน แต่แตกต่างกันที่รูปแบบและค่าน้ำหนัก

2.4 การแปลงชุดตัวเลขแบบหน้าต่างเลื่อนไหล

ในปี ค.ศ. 2004 ฟิลิปส์ (Philips) [4] ได้นำเสนอการแปลงชุดตัวเลขให้มีค่าน้ำหนักน้อยที่สุดโดยใช้หน้าต่างเลื่อนไหล (sliding window conversion) สำหรับระบบจำนวนฐานที่เป็นจำนวนเต็มบวกที่มากกว่าหรือเท่ากับสอง การแปลงแบบหน้าต่างเลื่อนไหลนี้สามารถกำหนดขนาดของหน้าต่างและชุดตัวเลขได้ กระบวนการนี้เป็นวิธีการแปลงที่นำตัวเลขที่ตำแหน่งติดกันมาจัดกลุ่มรวมกันเพื่อสร้างตัวเลขตัวใหม่โดยอาศัยตัวเลขในชุดตัวเลขซ้ำซ้อน และการแปลงนี้จะช่วยลดจำนวนของตัวเลขที่ไม่เป็นศูนย์โดยเฉลี่ย การแปลงชุดตัวเลขแบบหน้าต่างเลื่อนไหลนี้ทำจากขวาไปซ้าย คือตำแหน่งที่มีนัยสำคัญต่ำสุด (least significant digit first; LSDF)ไปตำแหน่งที่มีนัยสำคัญสูงสุด (most significant digit first; MSDF) อัลกอริทึมในการแปลงมีดังต่อไปนี้

Let $Y = \{y: l \le y \le u, y \neq 0 \bmod r\} \cup \{0\}$
$X = \{0, 1, 2, \ldots, r-1\}$
r = fixed radix which is an integer which is greater than or equal to 2
m = width of sliding window which is an integer and is greater than or equal to 1
l = lower bound on the digit set
u = upper bound on the digit set

/* Perform the conversion $Y = SW_{r,m,l,u}(X)$ */
$skip = 0$, $carry = 0$
$x_n = 0, x_{n+1} = 0, \ldots, x_{n+m} = 0$
for $i = 0$ to $n-1$
    if ($skip = 0$) then
        if ($x_i + carry \bmod r = 0$) then
            /* Skip over zero digits */
            $y_i = 0$
        else
            $x = \sum_{j=0}^{m-1} x_{i+j} \times r^j$
            if ($x + carry > u$) then
                /* Must choose a negative digit */
                $y_i = x + carry - r^m$
                $carry = 1$
            elseif (($x_{i+m} = r-1$) and ($x + carry \ge l + r^m$)) then
                /* Choosing a negative digit may reduce the weight */
                $y_i = x + \text{carry} - r^m$
                $carry = 1$
            else
                /* Choose a positive digit */
                $y_i = x + carry$
                $carry = 0$
            end if
            $skip = m - 1$
        end if
    else

$y_i = 0$
$skip = skip - 1$
end if
next i
$y_n = carry$

$A\ (r_A = 2)$		(1,	0,	1,	1,	1,	1,	1,	0,	1,	0,	0,	0,	1,	0,	1,	0)
$SW_{2,3,0,7}(A)$	(0,	0,	0,	5,	0,	0,	7,	0,	0,	5,	0,	0,	0,	0,	0,	5,	0)
$SW_{2,3,-7,7}(A)$	(0,	0,	3,	0,	0,	0,	0,	0,	0,	-3,	0,	0,	0,	0,	0,	5,	0)
$SW_{2,3,-1,5}(A)$	(0,	0,	3,	0,	0,	0,	-1,	0,	0,	5,	0,	0,	0,	0,	0,	5,	0)
$B\ (r_B = 4)$		(1,	0,	0,	1,	2,	2,	0,	3,	3,	1,	1,	0,	3,	1,	1,	0)
$SW_{4,2,-15,15}(B)$	(0,	1,	0,	0,	0,	6,	0,	9,	0,	0,	-3,	0,	5,	0,	0,	-11,	0)
$SW_{4,2,-7,7}(B)$	(0,	1,	0,	0,	0,	7,	0,	-7,	0,	-1,	0,	5,	0,	3,	0,	5,	0)
$C\ (r_C = 2)$		(1,	0,	0,	1,	0,	0,	1,	0,	0,	1,	0,	0,	1,	0,	0,	1)
$SW_{2,3,-5,5}(C)$	(0,	1,	0,	0,	1,	0,	0,	1,	0,	0,	1,	0,	0,	1,	0,	0,	1)
$D\ (r_D = 2)$		(1,	0,	0,	0,	1,	0,	0,	0,	1,	0,	0,	0,	1,	0,	0,	1)
$SW_{2,3,-7,7}(D)$	(1,	0,	0,	0,	-7,	0,	0,	0,	-7,	0,	0,	0,	-7,	0,	0,	0,	-7)

รูปภาพที่ 2.1 ตัวอย่างการแปลงชุดตัวเลขโดยใช้หน้าต่างแบบเลื่อนไหล

จากรูปภาพที่ 2.1 ได้แสดงให้เห็นถึงการผลลัพธ์ของการแปลงชุดตัวเลขให้มีน้ำหนักลดของแบบเลื่อนไหลของฟิลิปส์ โดยอ้างอิงจากอัลกอริทึมด้านบน โดยในรูปภาพจะแสดงการแปลงของจำนวนนำเข้าในระบบเลขฐาน 2 และฐาน 4 โดยใช้ชุดตัวเลขที่แตกต่างกันด้วยรูปแบบ $SW_{r,m,l,u}$ เมื่อ r เป็นระบบเลขฐาน m เป็น ความกว้างของหน้าต่าง l และ u เป็นค่าต่ำสุดและสูงสุดของชุดตัวเลขที่จะนำมาแปลงตามลำดับ และจะสังเกตได้ว่าอัลกอริทึมนี้สามารถหารูปแบบแทนจำนวนที่มีค่าน้ำหนักน้อยที่สุดได้โดยการกำหนดค่าพารามิเตอร์ในอัลกอริทึม คือ ชุดตัวเลข และขนาดของหน้าต่าง

โดยในระบบจำนวน (β, D) การหารูปแบบแทนจำนวนที่มีค่าน้ำหนักน้อยที่สุดโดยวิธีหน้าต่างการเลื่อนไหลขึ้นอยู่กับการกำหนดขนาดของหน้าต่าง แต่ในงานวิจัยนี้ไม่ได้กำหนดว่ารูปแบบที่มีค่าน้ำหนักน้อยที่สุดในระบบเกิดจากขนาดของหน้าต่างขนาดเท่าใด

2.5 การคำนวณแบบเชื่อมตรง

ในกระบวนการคำนวณพื้นฐานแบบลำดับ (serial computation) ส่วนใหญ่นั้นจะเริ่มต้นการทำงานจากตัวเลขในตำแหน่งที่มีนัยสำคัญน้อยที่สุด ซึ่งนั่นก็หมายถึงการทำงานจากตัวเลขตำแหน่งทางด้านขวาสุดไปยังตัวเลขตำแหน่งด้านซ้ายสุดนั่นเอง แต่ในการหารนั้นการทำงานกลับเริ่มจากตัวเลขตำแหน่งที่มีนัยสำคัญมากที่สุด ซึ่งจะแตกต่างกัน ดังนั้น ถ้าเราต้องการทำงานแบบท่อตรง (pipelining process) เราควรจะทำให้การทำงานนี้เป็นไปในทางเดียวกัน และนั่นก็คือจุดประสงค์ของการคำนวณแบบเชื่อมตรง

ในปี ค.ศ. 1977 ทริเวดิและเอิซเซกโกเวก (Trivedi & Ercegovac) [5] ได้นำเสนอการคำนวณวิธีการทำงานแบบเชื่อมตรง (on-line computation) ซึ่งจะได้ผลลัพธ์ตำแหน่งต่อตำแหน่ง โดยการคำนวนแบบเชื่อมตรงนั้นจัดว่าเป็นอัลกอริทึมแบบเพิ่ม (incremental algorithm) คือเมื่อรับตัวเลขเข้าไปหนึ่งตำแหน่ง จะสามารถสร้างตัวเลขนำออกได้หนึ่งตำแหน่ง แต่การคำนวณแบบเชื่อมตรงยังมีคุณสมบัติที่สำคัญอีกประการหนึ่งคือค่าความหน่วงเชื่อมตรง (on-line delay) δ

ค่าความหน่วงแบบเชื่อมตรงนี้เป็นเลขจำนวนเต็มบวกขนาดเล็กใช้เพื่อระบุว่าผลลัพธ์ตำแหน่งแรกสามารถคำนวณได้จากจำนวน n ตำแหน่งของข้อมูลนำเข้ากับค่าความหน่วงเชื่อมตรง หรือ $n + \delta$ อย่างไรก็ตามการทำงานแบบเชื่อมตรงนั้นจะต้องใช้กับระบบจำนวนซ้ำซ้อน

2.5.1 การบวกแบบเชื่อมตรง

การบวกแบบเชื่อมตรงสามารถทำได้กับจำนวนซ้ำซ้อนโดยมีค่าความหน่วงเชื่อมตรง δ สามารถพิสูจน์ได้ดังทฤษฎีบทที่ 2.1 ด้านล่างนี้

ทฤษฎีบทที่ 2.1 *กำหนดให้ β เป็นจำนวนเต็ม โดย $\beta > 1$ และ D เป็นชุดของตัวเลขที่จำกัด โดย $D = \{-b, -b-1, \ldots, 0, 1, \ldots, b\}$ เมื่อ b เป็นจำนวนเต็มและ $\beta/2 \le b \le \beta-1$ และมีค่าความหน่วงเชื่อมตรง δ โดย*

$$\delta = \begin{cases} 2 : b = \beta/2 \\ 1 : otherwise \end{cases}$$

การบวกแบบเชื่อมตรงสามารถพิสูจน์ได้ตามอัลกอริทึมด้านล่างนี้ และได้แสดงวิธีการทำในตัวอย่างที่ 2.4

บทพิสูจน์ของอัลกอริทึมสามารถดูรายละเอียดเพิ่มเติมได้ที่ [3]

อัลกอริทึมการบวกแบบเซื่อมตรง

Input : $X := (x_m x_{m-1}...)_\beta$ and $Y := (y_m y_{m-1}...)_\beta$ where $x_i, y_i \in D$

Output : $Z := (z_m z_{m-1}...)_\beta$ and $z_i \in D$

begin

$r_{m+1} = 0$;

$j := m$;

while $j \le m$ do

$s_j := x_j + y_j$;

if $-2b \le s_j < -b+1$

then $c_{j+1} := -1$; $r_j := s_j + n$; endif;

if $-b+1 \le s_j \le b-1$

then $c_{j+1} := 0$; $r_j := s_j$; endif

if $b-1 < s_j \le 2b$

then $c_{j+1} := 1$; $r_j := s_j - n$; endif;

$z_{j+1} := c_{j+1} + r_{j+1}$;

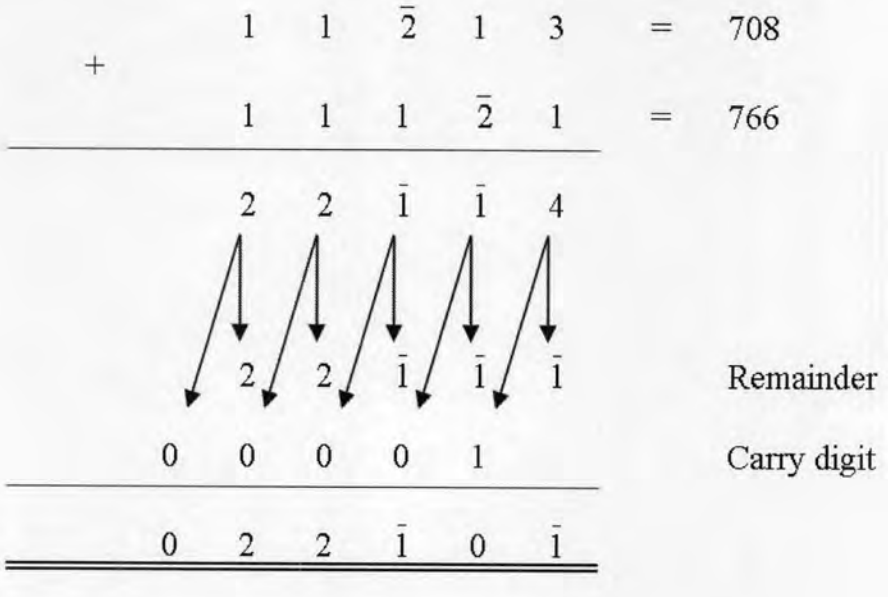
$j := j-1$;

enddo;

end;

ตัวอย่างที่ 2.4 *จงแสดงการบวกแบบเชื่อมตรงเมื่อ $\beta = 5$, $D = \{\bar{3}, \bar{2}, \bar{1}, 0, 1, 2, 3\}$ ระหว่าง $11\bar{2}13$ ซึ่งมีค่าเชิงตัวเลขคือ 708 และ $111\bar{2}1$ ที่มีค่าเชิงตัวเลขที่ 766 ด้วยค่าความหน่วงเชื่อมตรง $\delta = 1$*

รูปภาพที่ 2.2 การบวกแบบเชื่อมตรงของ 708 และ 766 ในระบบ (5, {$\bar{3}$, $\bar{2}$, $\bar{1}$, 0, 1, 2, 3})



คำตอบคือ $(022\bar{1}2\bar{1})_5 = (2 \times 5^4) + (2 \times 5^3) + (\bar{1} \times 5^2) + (\bar{1})$
$= 1250 + 250 - 25 - 1$
$= 1474$ □

จากตัวอย่างที่ 2.4 ด้านบนนั้น แสดงให้เห็นว่า ผลลัพธ์ของการบวกแบบเชื่อมตรงจะถูกผลิตออกมาที่ละดิจิตจากทางซ้ายด้วยค่าความหน่วงเชื่อมตรงเท่ากับ 1

ในปี ค.ศ. 2003 ฟรูนีย์และสุรฤกษ์ (Frougny & Surarerks) [7] ได้นำเสนอผลงานที่แสดงให้เห็นว่าการคำนวณแบบเชื่อมตรงนั้นสามารถทำได้ต่อเนื่องโดยจำเป็นต้องอาศัยระบบจำนวนซ้ำซ้อนแบบมีเครื่องหมาย เนื่องจากข้อดีของการทำงานแบบเชื่อมตรงนั้นคือสามารถทำการคำนวณได้โดยที่ไม่ต้องรอให้มีจำนวนนำเข้ามาทั้งหมดก็สามารถเริ่มทำการคำนวณหาผลลัพธ์ได้เลย ด้วยวิธีการนี้จะส่งผลให้สามารถทำงานได้เร็วขึ้น และผลลัพธ์ที่ได้สามารถนำไปคำนวนต่อได้เลย ทำให้การคำนวนอยู่ในรูปแบบทำงานแบบท่อตรง ข้อดีอีกประการหนึ่งคือ เมื่อนำวิธีการนี้มาทำให้เกิดผล (implement) ทรัพยากรที่ใช้ เช่น รีจิสเตอร์ (register) จะไม่มีความจำเป็นต้องใช้เป็นจำนวนมากเนื่องจากดิจิตที่ใช้ในการคำนวนมีจำนวนน้อย โดยจะขึ้นอยู่กับค่าความหน่วงเชื่อมตรงที่กำหนด ดังนั้นการคำนวนแบบเชื่อมตรงสามารถประหยัดทรัพยากรลงได้มากทีเดียว