最古の 2 進コンピューター
ネイピアの 2 進計算法

SHIMURA Masato
jcd02773@nifty.ne.jp

2012 年 1 月 21 日

目次

概要

ネイピアは数学分野では対数の発見やネイピアの骨棒で知られるが、最古のコンピューターとも言える 2 進計算法を残していた

最古の 2 進法

アメリカンサイエンスに永く連載された「マーチン · ガードナーの数学ゲーム」のうち初期に連載された赤攝也 · 冬子訳の 22 編が「II(新装版)] として 2011 年 12 月に日経サイエンス社から出版された。いずれも水準の高い力作であるので、幾つかを配列演算言語 J で料理してみよう。

ガードナーは「最古の 2 進コンピューター」(チェス盤を計算機に変える方法と”負の 2 進記数法”で計算す

る方法)」としてスコットランドの数学者ネイピアの今日では忘れ去られた興味深い方法を紹介している。

1　J と 2 進法

1.1　基底

2 進法と 10 進法への戻しは単項の `base(#:)` と `unti-base(#.)` を用いて簡潔に表すことができる

	base	untibase

単項
2 進
	`   #: 19`
	`1 0 0 1 1`
	`   #: 13`
	`1 1 0 1`
		`  #. 1 0 0 1 1`
		`19`
		
		`  #. 1 1 0 1`
		`13`

両項
n 進
	` 4 4 4  #: i.36`
	NB. 4 **進**
	
	`  24 60 60  #: 8230`
	`2 17 10`
	
	NB. **秒**->**日、時、秒**
		`4 4 4 #. 4 4 4 #: i.36`
		
		` 24 60 60 #. 2 17 10`
		`8230`

1.2　ビットワイズ演算

J はビットワイズ演算 (n b.) をサポートしている。ボキャブラリの Boolean に用法が載っている。整数や 2 進数の論理演算が中心である。(詳細は b. Boolean 参照)

n	*b.*の用例	意味	*primitive*	*primitive* の用例
1	`0 1 (1 b.)/ 0 1`	*and*	*.	`0 1 *./ 0 1`
14	`0 1 (14 b.)/ 0 1`	*not – and*	*:	`0 1 *:/ 0 1`
7	`0 1 (7 b.)/ 0 1`	*or*	+.	`0 1 +./ 0 1`
8	`0 1 (8 b.)/ 0 1`	*not – or*	+:	`0 1 +:/ 0 1`
10	`10  b. 0 1`	*not*	−.	`(i.9)  -. 2 3 5 7`
6	`1 0 (6  b.) 0 1`	*not – equal*	.	`1 0 ~: 0 1`

[*1]

[*1] n はここに掲げたものの他に各 2 個の数値が割り振られている。

1.3 J の 2 進での四則計算

ビットワイズは論理演算中心で四則計算は入っていない。コンピューターは 2 進法で計算しているが表には出さないで各種言語に委ねている。強引に計算したらどうなるか。

足し算 ∑1 2 3 4 5 6 7 8 9 10

```
(>:i.10),. #:>: i.10
 1 0 0 0 1
 2 0 0 1 0
 3 0 0 1 1
 4 0 1 0 0
 5 0 1 0 1
 6 0 1 1 0
 7 0 1 1 1
 8 1 0 0 0
 9 1 0 0 1
10 1 0 1 0
```

*2

```
 +/  #: >: i.10
3 4 5 5

 #. +/  #: >:i.10
55
```

引き算 89 − 26

```
  #:  89 26
1 0 1 1 0 0 1
0 0 1 1 0 1 0

  -/ #:  89 26
1 0 0 0 0 _1 1

 #. -/ #:  89 26
63
```

掛け算 89 × 26

```
 #. 89* #:26
2314
```

(×89) の掛け算は反則 (?)。2 進数のみで行うには 2 進数の 26 を 89 回加算する

```
#. +/ ;("1)  ,. 89#<   #: 26
2314
```

割り算 メソポタミアでは割り算はまだなく逆数を掛けていた。2 進の逆数には補数を用いる

36 ÷ 12

```
  #. -.  _4{. #: #. 36* -. #: 12
```

*2 桁数が異なるとエラーになるので、桁数が異なる場合は左に 0 を足して右揃えする

3

この方法はいつも上手く行くとは限らない。分数、少数などの 2 進割り算を組み込んだコンピューター黎明期のエンジニアの努力が偲ばれるが、深入りするとアッセンブラを作ってしまう。

ネイピアと小数点　ネイピアは現代風の小数の用法も確立したようだ。

2　ライプニッツと 2 進法

1696　ライプニッツは 1696 年に著された「全ての数を 1 と 0 とによって表す驚くべき表記法。これは事物が神と無から由来すること、すなわち創造の神秘、を表現している。」で 0 と 1 という 2 つの数記号しか必要としない単純な表記法を提示している。

1703　ライプニッツは 1703 年にブーヴエ (Joachim Bouvet) の北京からの易の配列図を書いた便りを受け取り、2 進法により解読することを試みた。ライプニッツはこの前にウイーンでイエスズ会の神父とも会っている。
易経に著された陰陽は 2 進法と馴染む。

坤	艮	(炊)	(撰)	震	離	兌	乾
000	100	010	110	001	101	011	111
↖	↗	↖	↗	↖	↗	↖	↗
老陰		少陽		少陰		老陽	
00		10		01		11	
↖		↗		↖		↗	
陰				陽			
0				1			
↖				↗			
大極							

3　ネイピアと 2 進法

イギリスの最初の大数学者はスコットランド人のネイピア男爵 (1550-1617) である。ネイピア男爵は城主であって、黙示録の解説と失敗した終末の予言や対数の発見でよく知られているが、著書 [Rabdologia] の中でネイピアの骨棒や、チェス盤を用いた計算法を考案している。

チェス盤を用いた変換は特に 2 進法であると断ってはいないが、ガードナーはこれをライプニッツより凡そ 100 年早い、世界最初の 2 進コンピューターであると紹介している。

3.1　ネイピア盤

チェス盤 (8 × 8) はチェックがあって桁を数えるには便利であるが、碁盤、将棋盤、オセロ盤などでもできる。私は、囲碁の 9 路盤が有ったので、これを用いた。

- 盤は右下を原点とする。駒はチェスのポン、囲碁の黒石、将棋の歩など

- 足し算、引き算の 2 進桁は行のみ。掛け算、割り算、平方は行と列の 2 進桁を用いる

								$2^6 = 64$
								$2^5 = 32$
								$2^4 = 16$
								$2^3 = 8$
								$2^2 = 4$
								$2^1 = 2$
								$2^0 = 1$
2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1	

- 桁上げ *halving up*
  2 個を取り去って左隣の枡に *1* 個加える

	●●	

↙

●		

- 桁下げ *doubling down*
  1 個を取り去って右隣の枡に *2* 個加える

	●	

↘

		●●

3.2　2 進の足し算と引き算

足し算　82 + 19 + 26

- 盤に置く。足し算、引き算は右辺の 2^n は用いず、単に 3 段とする

	●		●			●		82
			●			●	●	19
			●	●		●		26
128	64	32	16	8	4	2	1	

- 最下段へ集める

	●		●●●	●		●●●	●
128	64	32	16	8	4	2	1

- 桁上げ。2 進では 1 枡に入るのは ● 一個のみ。2 個以上有る場合は 2 個を一組として繰り上げる。

	●	●	●	●	●	●	●	127
128	64	32	16	8	4	2	1	

$$64 + 32 + 16 + 8 + 4 + 2 + 1 = 127$$

この桁上げは *halving up* と呼ばれ、枡から ●*2* 個を一組として取り去り、左隣りの枡に ● 一個を加える。*n* の駒 *2* 個が *2n* の駒 *1* 個と置き換わる

- *anti_base*

```
 #. 1 1 1 1 1 1 1
127
```

引き算 (1)-**オードソックスな方法**　引き算は 2 列を用いる。

- 120 − 89

```
#: 120 89
1 1 1 1 0 0 0
1 0 1 1 0 0 1
```

	●	●	●	●				120
	●		●	●			●	89
128	64	32	16	8	4	2	1	

- 右端からはじめる。
- 繰り下げ。引けないときは左の枡から桁下げを行う。左の枡の ●1 個 (=2n) を取り去り、右の枡に ●2 個 ($n \times 2$) を加える。このケースでは 3 枡隣から繰り下げを始める

	●	●	●		●	●	●●	120
	●		●	●			●	89
128	64	32	16	8	4	2	1	

- 次に上段が 0 で下段が 1 の場合は逐次繰り下げる。一度に行えば次になる

	●		●●	●●	●	●	●●	120
	●		●	●			●	89
128	64	32	16	8	4	2	1	

- 上段、下段各 1 個ずつの ● のペアを取り去ると解が上段に残る

			●	●	●	●	●	31
128	64	32	16	8	4	2	1	

- anti_base

```
#. 1 1 1 1 1
31
```

引き算 (2)-補数を用いる方法　下段を補数に置き換えて加算

- 120 − 89

	●	●	●	●				120
	●		●	●			●	89
128	64	32	16	8	4	2	1	

- 下段を補数に置き換える。空欄に駒 (石) を置き、最初の駒 (石) を取り去る

	●	●	●	●				120
		○			○	○		
128	64	32	16	8	4	2	1	

- 加算する。桁上げも行う

●			●	●	●	●		
128	64	32	16	8	4	2	1	

- もう 1 ステップ必要。一番左の駒を持ち上げ一番右に移す

			●	●	●	●	●	31
128	64	32	16	8	4	2	1	

3.3　掛け算と割り算

掛け算と割り算では縦も 2 進桁を用いる。活躍するのはチェスのビショップ、将棋の角だが、次の計算でも 11 行必要なので碁盤が向いている

掛け算　掛け算は以外にシンプルである

- 39 × 27

```
 #:  39 27
1 0 0 1 1 1
0 1 1 0 1 1
```

次のように石を埋める

			●	●		●	●	27	1
			●	●		●	●	27	1
			●	●		●	●	27	1
			●	●		●	●	27	1
128	64	32	16 1	8 1	4	2 1	1 1	27/	/39

- 最右列以外の石を全て斜め上へ動かし、一番右の列に集約する

				↗	↗	↗			
			●	●		●	●	27	1
				↗	↗	↗			
				↗	↗	↗			
			●	●	↗	●	●	27	1
			●	●	↗	●	●	27	1
			●	●	↗	●	●	27	1
128	64	32	16 1	8 1	4	2 1	1 1	27/	/39

								1024
							●	512
							●	256
								128
							●●	64
							●●●	32
							●●	16
							●●	8
							●●	4
							●●	2
							●	1
128	64	32	16	8	4	2	1	

- 最右行で加算のように 2 進桁上げを行う (延々と)

							●	1024
								512
								256
								128
								64
								32
							●	16
							●	8
							●	4
								2
							●	1
128	64	32	16	8	4	2	1	(1053)

- anti_base

```
#. 1 0 0 0 0 0 1 1 1 0 1
1053
```

ネイピアの骨棒　ネイピアに敬意を表してネイピアの骨棒で確認してみよう

		2		7
3	0	↗	2	↗
	↗	6	↗	1
9	1	↗	6	↗
	↗	8	↗	3

	2	1	
	6	6	
0	1	8	3
1	0	5	3

割り算　割り算は多少複雑でトライアル アンド エラーを繰り返す必要があるが、コツを掴んで馴れてしまおう。引き算より頻繁に桁下げを行う必要がある。

- 169 ÷ 26

```
 #:  169 26
1 0 1 0 1 0 0 1
0 0 0 1 1 0 1 0
```

169 は最右列に配置する
26 は行の枠外に目印を付けておく

								256
							●	128
								64
							●	32
								16
							●	8
								4
								2
							●	1
128	64	32	16	8	4	2	1	
			○	○		○		(26)/169

- 最初のトライ。
  1. 最右列の 128 の駒を ○ の左端の行 · 列に動かす
  2. この行を目印の ○ に従って完成させる
  3. ●32 の駒は ○8 には入らない
  4. 元に戻す

								256
								128
						↙		64
					↙		●	32
				↙				16
			●				●	8
								4
								2
							●	1
128	64	32	16	8	4	2	1	
			○	○		○		

- 2 回目のトライ。
  1. 最右列の 128 の駒を桁下げする。(●128 の駒を外し ●64 に 2 個加える)
  2. 最右行の ●64 の内 1 個を ○16 の枠にマッチさせる。 右列の 4 の行の位置に入る
  3. 4 の行を完成させる。
     - ●64 はもう使えないので桁下げする
     - ●32 の一個を 4 の行の ○8 に移動する
     - ●8 を 4 の行の ○2 に移動する

								256
								128
							●	64
						↙	●	32
					↙			16
				↙			●	8
			●					4
								2
							●	1
128	64	32	16	8	4	2	1	
			○	○		○		

								256
								128
								64
							●●	32
						↙		16
					↙		↙	8
			●	●		●		4
								2
							●	1
128	64	32	16	8	4	2	1	
			○	○		○		

- トライを続ける。
  1. 最右列の ●32 の駒の内 1 個を ○16 にマッチさせる (4 の行は飛び越える)
  2. 2 の行に入る。
  3. 2 の行を完成させる。(完成できなければ 1 に戻る)
     - ●32 の駒は使わないので桁下げする
     - ●16 の 1 個を 2 の行の ○8 に移す
     - ●16 の駒は使わないので桁下げする
     - ●8 の 2 個の駒は使わないので桁下げする
     - ●8 の 1 個を 2 の行の ○2 に移動する
  4. ●4 の駒は ○16 には入れないので計算を終了する

								256
								128
								64
							●	32
						↙		16
					↙			8
			●	●		●		4
			●					2
							●	1
128	64	32	16	8	4	2	1	
			○	○		○		

								256	
								128	
								64	
								32	
								16	
								8	
			●	●		●	●●●	4	1
			●	●		●		2	1
							●	1	0
128	64	32	16	8	4	2	1		
			○	○		○			6

- 解は 2 進の 1 1 0 で 10 進では 6
- 余りは ●4 が 3 個なので桁下げし、 ●1 の 1 個を加えて 13。

3.4　2 乗 平方根

2 乗は対角線を挟んで対象に駒を並べる。

								256
↖								128
	↖							64
		↖						32
			↖					16
				↖				8
					↖			4
						↖		2
							↖	1
128	64	32	16	8	4	2	1	

4^2 **と** 16 .

								256
↖								128
	↖							64
		↖						32
			↖				●	16
				↖		↙ ↗		8
					●			4
						↖		2
							↖	1
128	64	32	16	8	4	2	1	

3^2 **と** 9 .

3^2 (行と列は必ず対称型となる)

								256
↖								128
	↖							64
		↖						32
			↖					16
				↖				8
					↖			4
						●	●	2
						●	●	1
128	64	32	16	8	4	2	1	

$3^2 \to 9$ は掛け算

3^2 を掛け算で最右列に集約して桁上げすれば 9 が現れる

$9 \to 3^2$

9

1. 最右列の ●8 は対角線に入らないので桁下げする
2. ●4 の内 1 個は対角線に入る。
3. 残りの 1 個を桁下げする
4. 2 の行を完成させ、1 の行を完成させると 3^2 になる。(余りはない)

								256	
↖								128	
	↖							64	
		↖						32	
			↖					16	
				↖			●	8	
					↖			4	
						↖		2	
							●	1	
128	64	32	16	8	4	2	1		

4　幾つかの基底とネイピア盤

両項で 2 進法　J の基底 (#:) を用いた 2 進法は両項でも用いることができる

```
(i.11),.2 2 2 2 2#:i.11
--------------
 0 |0 0 0 0 0
 1 |0 0 0 0 1
 2 |0 0 0 1 0
 3 |0 0 0 1 1
 4 |0 0 1 0 0
 5 |0 0 1 0 1
 6 |0 0 1 1 0
 7 |0 0 1 1 1
 8 |0 1 0 0 0
 9 |0 1 0 0 1
10 |0 1 0 1 0
```

4.1　負の 2 進法の基底

- 負の 2 進法の基底の研究は 1950 年頃から始まったようで、TEX で知られる D. クナースが高校生時代に既に参画していたようだ。
- −2 の累乗を基底とする
- 負の 2 進法では正負の記号を付けないで計算できる
- 負の 2 進法による整数の表記

```
(-i:24),. -  _2 _2 _2 _2 _2 _2 _2   #: i: 24
```

```
24 1 1 0 1 0 0 0
23 1 1 0 1 0 1 1
22 1 1 0 1 0 1 0
21 0 0 1 0 1 0 1
20 0 0 1 0 1 0 0
19 0 0 1 0 1 1 1
18 0 0 1 0 1 1 0
17 0 0 1 0 0 0 1
16 0 0 1 0 0 0 0
15 0 0 1 0 0 1 1
14 0 0 1 0 0 1 0
13 0 0 1 1 1 0 1
12 0 0 1 1 1 0 0
11 0 0 1 1 1 1 1
10 0 0 1 1 1 1 0
 9 0 0 1 1 0 0 1
```

```
 8 0 0 1 1 0 0 0
 7 0 0 1 1 0 1 1
 6 0 0 1 1 0 1 0
 5 0 0 0 0 1 0 1
 4 0 0 0 0 1 0 0
 3 0 0 0 0 1 1 1
 2 0 0 0 0 1 1 0
 1 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0
_1 0 0 0 0 0 1 1
_2 0 0 0 0 0 1 0
_3 0 0 0 1 1 0 1
_4 0 0 0 1 1 0 0
_5 0 0 0 1 1 1 1
_6 0 0 0 1 1 1 0
_7 0 0 0 1 0 0 1
_8 0 0 0 1 0 0 0
```

```
 _9 0 0 0 1 0 1 1
_10 0 0 0 1 0 1 0
_11 0 1 1 0 1 0 1
_12 0 1 1 0 1 0 0
_13 0 1 1 0 1 1 1
_14 0 1 1 0 1 1 0
_15 0 1 1 0 0 0 1
_16 0 1 1 0 0 0 0
_17 0 1 1 0 0 1 1
_18 0 1 1 0 0 1 0
_19 0 1 1 1 1 0 1
_20 0 1 1 1 1 0 0
_21 0 1 1 1 1 1 1
_22 0 1 1 1 1 1 0
_23 0 1 1 1 0 0 1
_24 0 1 1 1 0 0 0
```

- 負のネイピア盤

```
   _2^i.8
1 _2 4 _8 16 _32 64 _128
```

								$-2^6 = 64$
								$-2^5 = -32$
								$-2^4 = 16$
								$-2^3 = -8$
								$-2^2 = 4$
								$-2^1 = -2$
								$-2^0 = 1$
-2^7	-2^6	-2^5	-2^4	-2^3	-2^2	-2^1	-2^0	
-128	64	-32	16	-8	4	-2	1	

- 精算規則 (桁上げ)
  1. 3 個の精算

     3 個を取り去る

•	••	

↓

  2. 2 個の精算

     2 個を取り去り、左の 2 桁に 1 個づつ入れる

		••

↓

●	●	

乗算　−4 × −6

- セット

								256	
								−128	
								64	
								−32	
								16	
				●	●			−8	1
				●	●			4	1
				●	●			−2	1
								1	0
−128	64	−32	16	−8	4	−2	1		
				1	1	0	0	−4	/ − 6

- 掛け算の右寄せ

								256	
								−128	
							●	64	
							●●	−32	
							●●	16	
							●	−8	
								4	
								−2	
								1	0
−128	64	−32	16	−8	4	−2	1		

- 精算 (上ならでも下からでも良いが、上から取ったほうが精算が早い)

								256	
								−128	
							●	64	
							●	−32	
								16	
							●	−8	
								4	
								−2	
								1	0
−128	64	−32	16	−8	4	−2	1		

1 1 0 1 0 0 0 = 24

加算　-6 + 19 + -17

- セット

				•	•	•			−6
			•		•	•	•		19
		•	•			•	•		−17
−128	64	−32	16	−8	4	−2	1		

- 最下段に集約

		•	••	•	••	•••	••		
−128	64	−32	16	−8	4	−2	1		

- 最下段に集約し (-32,16),(-8,4) を先に取り去る
- (−1, 1) の 3 個を取り去ると −2 が 2 個残る

				•	•			−4	
−128	64	−32	16	−8	4	−2	1		

```
1 1 0 0 = -4
```

4.2　フィナボッチ基底とフィボナッチ進法

イタリア · ピサ生まれのフィボナッチ (1170 頃-1250 頃) は父の貿易商としてムワヒッド朝 (今のアルジェリア) への居住に伴って科学の先進地アラブで学んだ

D. クナースはネイピア盤を用いてフィボナッチ基底でも計算している

フィボナッチ数　.

```
   {: (] ,[: +/ _2&{.)^:(i.20)  1 1
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946
```

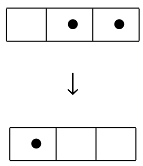
```
fib=: 3 : '{: (] ,[: +/ _2&{.)^:(i.y)  1 1'
```

フィボナッチ盤 .

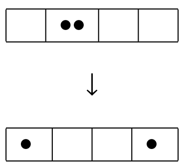
								34
								21
								13
								8
								5
								3
								2
								1
34	21	13	8	5	3	2	1	

桁上げ規則　フィボナッチ数のルールにより行う

1. 2 個の桁上げ (1)
   例えば 5 + 3 → 8
   駒 (石) の数を減らすためこちらを多用する。



2. 2 個の桁上げ (2)
   2 個を取り去り、左の枡と右の 2 個隣の枡に 1 個づつ入れる。駒の数が減らないので補助手段とする
   例えば 8 + 2 ← 5 × 2



4.2.1　**J のフィボナッチ基底**

J でフィボナッチ基底を試みる　J の基底の左引数にフィボナッチ数を与える。

```
}:"1  (fib),. 10}."1 (}.|.fib), }:"1}.(|.fib)   #: fib
```

```
  n 55 34 21 13  8 5 3 2 1
--------------------------
  1  0  0  0  0  0 0 0 0 1
  2  0  0  0  0  0 0 0 1 0
  3  0  0  0  0  0 0 0 1 1
  5  0  0  0  0  0 0 0 2 1
  8  0  0  0  0  0 0 1 1 0
 13  0  0  0  0  0 0 2 0 1
 21  0  0  0  0  0 0 3 1 1
 34  0  0  0  0  0 1 0 2 0
 55  0  0  0  0  0 1 4 0 1
 89  0  0  0  0  0 2 4 2 1
144  0  0  0  0  0 4 4 0 0
233  0  0  0  0  0 7 3 2 1
377  0  0  0  0  1 4 2 2 1
610  0  0  0  0  2 4 1 2 0
987  0  0  0  0  4 0 4 1 1
1597 0  0  0  0  6 5 1 0 1
2584 0  0  0  0 10 6 0 2 0
4181 0  0  0  1  4 3 1 2 1
6765 0  0  0  2  2 1 2 1 1
10946 0  0  0  3  6 4 4 1 0
```

フィボナッチ基底は

- 1 の列は 2 進
- 2 の列は 0 1 2 の 3 進
- 3 の列は 0 1 2 3 の 4 進

とフィボナッチ数に従って進法が増える

J のフィボナッチ基底の構造　J のフィボナッチ基底の構造は次のようになっている。

```
   (0,b),. a,    (a=.34 21 13 8 5 3 2 1  )   #: b=.5 6 29 30 239 240
   34 21 13 8 5 3 2 1
  -------------------
  5  0  0  0 0 0 2 1
  6  0  0  0 0 1 0 0
 29  0  0  0 0 4 2 1
 30  0  0  0 1 0 0 0
239  0  0  0 7 4 2 1
240  0  0  1 0 0 0 0
```

- 1 の列は 0 1 の 2 進
- 2 の列は 0 1 2 の 3 進

0	1	2	3	4	5
$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$	$(2,0)$	$(2,1)$

  5 まで、6 で繰り上がり
- 3 の列は (1,0,0) は 6 となり、6 進。
  (4,2,1) は $(29 = 4 \times 6 + 2 \times 2 + 1)$ であり、30 で繰り上がり

- 5 の列は (1,0, 0,0) は 30。
  (7,4,2,1) は ($239 = 30 \times 7 + 4 \times 6 + 2 \times 2 + 1$) であり、240 で繰り上がり

この基底はネイピア盤とのなじみが良くない

4.2.2 クナースのフィボナッチ基底

クナースのネイピア盤を用いたフィボナッチ基底は最初各枡に入る駒は一個限定とする。

ガードナーは本文で割り算や掛け算はできないけれど全ての整数をなるべく少ない個数のフィボナッチ数の和として表せば、足し算や引き算はできるとしている。解答欄では、カリフォルニア州サンタバーバラの *John Haris* が考えた枠外にもう一桁付け足す掛け算のアルゴリズムを紹介している。

0/1 表示のフィボナッチ数 2 進で 1024 打ち出して、これを指標としてフィボナッチ数を取り出し合計を求めると 10 桁で 0-231 の連続した数が求まる。組み合わせに何通りかの重複がある

```
   $ ~.  /:~ +/"1  (#:i.1024) # }. fib 10
232
```

0/1 表示のフィボナッチ数を求める 19 を総当り法で求める。

```
   fib_all 19
89 55 34 21 13 8 5 3 2 1
 0  0  0  0  1 0 1 0 0 1
 0  0  0  0  1 0 0 1 1 1
 0  0  0  0  0 1 1 1 1 1
```

```
   fib_all 100
89 55 34 21 13 8 5 3 2 1
 1  0  0  0  0 1 0 1 0 0
 0  1  1  0  0 1 0 1 0 0
 0  1  0  1  1 1 0 1 0 0
 1  0  0  0  0 1 0 0 1 1
 0  1  1  0  0 1 0 0 1 1
 0  1  0  1  1 1 0 0 1 1
 1  0  0  0  0 0 1 1 1 1
 0  1  1  0  0 0 1 1 1 1
 0  1  0  1  1 0 1 1 1 1
```

総当り法の Script 盤のサイズでとどめてあるが (1024, 10) を変えれば広げられる

```
fib_all=: 3 : 0
NB. u 19
tmp1=:  (#:i.1024) # }. tmp0=: fib 10
ind0=: (+/"1 tmp1) e. y
ind1=: |."1 ind0 # tmp1
tmp2,(tmp2=.|.}. tmp0) e."1 ind1
)
```

足し算と引き算 .

- $82 + 19 + 26$
- フィボナッチ数の (0, 1)

```
   fib_all  82
89 55 34 21 13 8 5 3 2 1
 0  1  0  1  0 0 1 0 0 1
 0  1  0  0  1 1 1 0 0 1
 0  0  1  1  1 1 1 0 0 1
 0  1  0  1  0 0 0 1 1 1
 0  1  0  0  1 1 0 1 1 1
 0  0  1  1  1 1 0 1 1 1
```

```
   fib_all  19
89 55 34 21 13 8 5 3 2 1
 0  0  0  0  1 0 1 0 0 1
 0  0  0  0  1 0 0 1 1 1
 0  0  0  0  0 1 1 1 1 1
   fib_all  26
89 55 34 21 13 8 5 3 2 1
 0  0  0  1  0 0 1 0 0 0
 0  0  0  0  1 1 1 0 0 0
 0  0  0  1  0 0 0 1 1 0
 0  0  0  0  1 1 0 1 1 0
```

- 駒のセット

										(8)	
										(5)	
		•	•	•	•		•	•	•	(3)	82
				•			•	•	•	(2)	19
				•	•		•	•		(1)	26
89	55	34	21	13	8	5	3	2	1		

- 下段に寄せる

										(3)
										(2)
		•	•	•••	••		•••	•••	••	(1)
89	55	34	21	13	8	5	3	2	1	

- 桁上げ。 駒を減らす繰上げを優先する

										(3)	
										(2)	
	•	•	•		•	•	•		•	(1)	127
89	55	34	21	13	8	5	3	2	1		

- J の anti-base は?

```
 34 21 13 8 5 3 2 1  #.  1 1 3 2 0 3 3 2
69431

    55 34 21 13 8 5 3 2  1  #.  1 1 1 0 1 1 1 0 1
2296359
  +/  55 34 21 13 8 5 3 2  1  #~  1 1 1 0 1 1 1 0 1
127
```

引き算

- 120 – 89

```
fib_all 120
89 55 34 21 13 8 5 3 2 1
 1  0  0  1  0 1 0 0 1 0
 0  1  1  1  0 1 0 0 1 0
 1  0  0  1  0 0 1 1 1 0
 0  1  1  1  0 0 1 1 1 0
 1  0  0  0  1 1 1 1 1 0
 0  1  1  0  1 1 1 1 1 0
```

```
fib_all 89
89 55 34 21 13 8 5 3 2 1
 1  0  0  0  0 0 0 0 0 0
 0  1  1  0  0 0 0 0 0 0
 0  1  0  1  1 0 0 0 0 0
 0  1  0  1  0 1 1 0 0 0
 0  1  0  1  0 1 0 1 1 0
```

- 駒のセット

											(8)	
											(5)	
											(3)	
		•	•		•	•	•	•	•		(2)	120
		•		•		•		•	•		(1)	89
144	89	55	34	21	13	8	5	3	2	1		

- 下の駒の上に必ず上の駒が来るように桁下げする。繰下げは桁上げの逆

											(8)	
											(5)	
											(3)	
		•		•	••	•	•	•	•		(2)	120
		•		•		•		•	•		(1)	89
144	89	55	34	21	13	8	5	3	2	1		

- 上下揃った駒を取り去る

											(8)
											(5)
											(3)
					••		•				(2)
											(1)
144	89	55	34	21	13	8	5	3	2	1	

- 桁上げを行い整理する

											(8)	
											(5)	
											(3)	
				•		•			•		(2)	31
											(1)	
144	89	55	34	21	13	8	5	3	2	1		

References

河田直樹「ライプニッツ 普遍数学への旅」 現代数学社 2010

マーチン · ガードナー/赤攝也 · 冬子訳「マーチン · ガードナーの数学ゲーム II(新装版)」日経サイエンス社 2011

J 言語の入手

`http://www.jsoftware.com` から DL する。