JAPLA 研究会資料 2020/12/15
柏新田原セミナー 2020/12/13

Ｊでコラッツの問題を計算する

西川　利男

ふとした機会に、コラッツの問題というのにでっくわした。これは、1937 年に
ローター・コラッツが発表し、日本の数学者、角谷も関与しているそうだ。

１．コラッツの数列とは

計算のルールは、次のように簡単である。

任意の整数をとって

偶数ならば、２で割る

奇数ならば、３を掛けて１を加える

この計算で得られれた値に対して、同じ計算をさらに行う。これを次々とくりかえす。

例えば、3 のときは

3 10 5 16 8 4 2 1 4 2 1 …

とつづく。4 になるとその後は4 2 1 4 2 1　と続く。

この計算をＪでプログラミングしてみた。これを用いて、コラッツの数列は
次のようになる。

```
   collatz 1
1
   collatz 2
2 1
   collatz 3
3 10 5 16 8 4 2 1
   collatz 4
4 2 1
   collatz 5
5 16 8 4 2 1
   collatz 6
6 3 10 5 16 8 4 2 1
```

7のときは、急に長くなる。

```
   collatz 7
7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
   collatz 8
8 4 2 1
   collatz 9
9 28 14 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
   collatz 10
10 5 16 8 4 2 1

   collatz 20
20 10 5 16 8 4 2 1
   collatz 21
21 64 32 16 8 4 2 1
```

さらに、27のときには、とんでもないことが起こる。おどろくなかれ、コラッツの数列は、112項にもなるのである。

```
   collatz 27
27 82 41 124 62 31 94 47 142 71 214 107 322 161 484 242 121 364 182 91 274
137 412 206 103 310 155 466 233 700 350 175 526 263 790 395 1186 593 1780 890
445 1336 668 334 167 502 251 754 377 1132 566 283 850 425 1276 638 319 958
479 1438 719 2158 1079 3238 1619 4858 2429 7288 3644 1822 911 2734 1367 4102
2051 6154 3077 9232 4616 2308 1154 577 1732 866 433 1300 650 325 976 488 244
122 61 184 92 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
   # collatz 27
112
```

次にcollatz 82 は上の実験から分かるように初めの項を除いたもので111項になる。

```
   # collatz 82
111
   # collatz 41
110
   # collatz 124
109
…
```

のようになる。

まさに、ふしぎなコラッツ数である。

２．Ｊによるプログラミング

以上の計算を行うコラッツ数のＪプログラミングを述べていこう。

まず、基本の計算をつぎのようにＪの関数 col として定義する。

```
col =: 3 : 0          関数定義の宣言
if. 0 = 2|y.          関数の引数 y.が偶数かどうか (2 で割った余りが 0 かどうか)
 do.  y. % 2          偶数(真) ならば、2 で割る
 else. 1 + 3 * y.     奇数(偽)  ならば、3 を掛けて、1 を加える
end.
)
```

この関数を引数５ として、実行してみよう。

```
   col 5 => 16
```

が返される。

次に、今得られれた 16 を引数として、やってみる。

```
   col 16 => 8
```

これを、つぎつぎと繰り返しやってみる。

```
   col 8 => 4
   col 4 => 2
   col 2 => 1
```

これから、先は

```
   col 1 => 4
   col 4 => 2
   col 2 => 1
```

と繰り返す。

Ｊでは、繰り返し計算を簡便に行うことができる。col を 2 回おこなうと
つぎのようになるが、

```
   col (col 5) => 8
```

これにに対して、つぎの記号(^:)を用いると

```
   col (^: 2) => 8
```

になる。同様にして

```
   col (^: 3) => 4
   col (^: 4) => 2
   col (^: 5) => 1
```

となる。

さらに、これらの計算をまとめて行うことができる。Ｊの記号 i. を用いると

```
   i. 6 => 0 1 2 3 4 5
```

なる数値列を生成できるので、これを利用すると

```
   col (^: (i.6)) 5 => 5 16 8 4 2 1
```
ちなみに、つぎのようになる。
```
   col (^: (i.10)) 5 => 5 16 8 4 2 1 4 2 1 4
```

ところで、上の数列を1になったらところまでの数列としてを得たいものである。これには次のような工夫が必要である。

```
   C =: col(^: (i.10)) 5   とりあえず、10項をとりCとする。
   C
5 16 8 4 2 1 4 2 1 4
   1 = C                   数列Cの要素のうち、値が1かどうかを調べる。
0 0 0 0 0 1 0 0 1 0        1は値1である、0は値1でない。
   # C                     数列Cの項数
10
   C i. 1                  数列Cから最初に1になる場所が何番目かを探す
5                          0オリジンなので5が返されるが、6番目である。
   5 {. C
5 16 8 4 2
   6 {. C                  最初から6項までを取ればよい。
5 16 8 4 2 1
```

以上の実験をもとに、次のJプログラムを、collatzとして作成した。

関数collatzは、とりあえず120項を取り右引数に対するコレッツ数列を計算する。それでも足りないときは、collatzの左引数として指定して、計算するようにした。

```
collatz =: 3 : 0
120 collatz y.
:
COL =. col (^:(i. x.)) y.
n =. COL  i. 1
(n+1) {. COL
)
```

実行のようすは、前に示したとおりである。プログラムのコーディングは以下に示した。

```
NB. Collatz.ijs    on J4
NB.   in case, on J6, J9,  replace y. to y
NB.   programmed by T. Nishikawa
NB.   2020/12/9


col =: 3 : 0
if. 0 = 2|y.
  do.   y. % 2
  else. 1 + 3 * y.
end.
)

collatz =: 3 : 0
120 collatz y.
:
COL =. col (^:(i. x.)) y.
n =. COL  i. 1
(n+1) {. COL
)
```