JAPLA シンポジウム資料 2011-12-10

Ｊ言語からの群論の理解－その３
－直接置換、巡回置換、互換、隣接互換－

西川　利男

０．はじめに

先月の前報では、微分・積分など計算数学(Calculus)に対して、群論はかなり異なり、その理解にはＪの動詞、名詞の考え方が有効ある。また群論の操作の実験ツールとしてもＪが極めて強力であることを示した。

［1］西川利男「Ｊ言語からの群論の理解－その１－置換群」2011-11-26

１．直接置換と巡回置換

直接置換(direct permutation)とは別に巡回置換(cycle permutation)という操作があり、直接置換は巡回置換の積に変換できる。

以下、次の数学書［2]の説明にしたがって、Ｊで実験してみよう。

［2］難波誠「群と幾何学」p.54-57　現代数学社(1997).

例として、次の置換（直接置換）でやってみる。

$$\begin{pmatrix} 1\,2\,3\,4\,5 \\ 2\,3\,1\,5\,4 \end{pmatrix}$$

これは、

１番目の場所に、２番目の値を
２番目の場所に、３番目の値を
３番目の場所に、１番目の値を
４番目の場所に、５番目の値を
５番目の場所に、４番目の値を

同時に置き換えることを示す。

これは次のように巡回置換の積でも表せる。

(1 2 3) (4 5)

ここで、

(1 2 3)

は１番目の場所には２番目の値を、２番目の場所には３番目の値を、３番目の場所には１番目の値を、サイクリックに入れ換える操作を示し、これが巡回置換である。これから分かるように巡回置換(1 2 3)は巡回置換(2 3 1)とも(3 1 2)とも同じである。

つづいて

(4 5)

は４番目の場所には５番目の値を、５番目の場所には４番目の値を、サイクリックに入れ換える。

ここで、２つだけの巡回置換は特に互換(interchange, transposition, swap)という。

また、共通の文字がない巡回置換の積は交換可能である。直接置換は互いに共通の文字がないときにはただ１通りの巡回置換の積に変換される。

この相互の変換はＪのプリミティブC.により次のように行われる。Ｊではインデックスは０－オリジンであり、数学書の表記から１減じていることに注意。
直接置換はＪのプリミティブC.の単項を使用して巡回置換に変換される。

```
    C. 1 2 0 4 3
+-----+---+
|2 0 1|4 3|
+-----+---+
```

数学書に合わせたければ

```
 >: L:0 C. <: 2 3 1 5 4
+-----+---+
|3 1 2|5 4|
+-----+---+
```

となる。

実際に文字の集合の値に、直接置換と巡回置換の操作を行って比較してみる。
まず、直接置換を行う。これには、Ｊのプリミティブ C.を２項動詞として使って行われる。インデックスは０－オリジンである。

```
   1 2 0 4 3 C. 'ABCDE'
BCAED
```

これは下のようにしても同じである。

```
   1 2 0 4 3 { 'ABCDE'
BCAED
```

これを、先の巡回置換の結果に従い、それらの積つまり連続操作で行ってみる。同じＪのプリミティブC.を、左引数をボックスで与えると巡回置換が行われる。

```
   (<4 3) C. 'ABCDE'
ABCED
   (<2 0 1) C. (<4 3) C. 'ABCDE'
BCAED
```

積の順序を変えてみる。

```
   (<2 0 1) C. 'ABCDE'
BCADE
   (<4 3) C. (<2 0 1) C. 'ABCDE'
BCAED
```

これは、次のように一度に行える。

```
   (2 0 1;4 3) C. 'ABCDE'
BCAED
```

２．互換と隣接互換

巡回置換はつぎのようにして、互換の積へ変換できる。
(1 2 3) ＝(1 2) (1 3)
(1 2 3 4) ＝(1 2) (1 3) (1 4)
しかし、互換の積への分解は１通りには限らないことに注意。
さらに、互換の中で
(1 2), (2 3), (3,4),..
のようなものを隣接互換という。隣接互換へは次のようにして変換できる。

(i j) ＝ (i, i+1)(i+1, i+2) ‥ ‥ (j-1, j)(j-1, j-2)‥ ‥ (i+1, i)

Ｊによるこれらの変換の動詞を作った。定義は最後にあげてある。

巡回置換から互換への変換……cyc2swap

巡回置換から隣接互換への変換……cyc2adjswap

```
   cyc2swap (1 2 3 4)
+---+---+---+
|1 2|1 3|1 4|
+---+---+---+
   cyc2swap (1 2 3 4 5)
+---+---+---+---+
|1 2|1 3|1 4|1 5|
+---+---+---+---+

   cyc2adjswap (1 2 3 4)
+---+---+---+---+---+---+---+---+---+
|1 2|1 2|2 3|2 1|1 2|2 3|3 4|3 2|2 1|
+---+---+---+---+---+---+---+---+---+
   cyc2adjswap (1 2 3 4 5)
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|1 2|1 2|2 3|2 1|1 2|2 3|3 4|3 2|2 1|1 2|2 3|3 4|4 5|4 3|3 2|2 1|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

先の直接置換の例をやってみる。

```
   C. 2 3 1 5 4
+-+-----+---+
|0|3 1 2|5 4|
+-+-----+---+
```

なお、１－originの数学表記をＪの直接置換のC.で行うと、0を付加して、数学表記の巡回置換が得られる。

```
   cyc2swap 3 1 2
+---+---+
|1 3|2 3|
+---+---+
   cyc2adjswap 3 1 2
1 2
2 3
2 1
2 3
3 4
+---+---+---+---+---+
|1 2|2 3|2 1|2 3|3 4|
+---+---+---+---+---+
```

いままでの結果からわかるように、任意の置換は隣接互換の積で表せる。

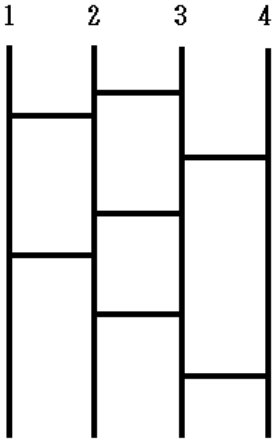
３．「あみだくじ」は隣接互換の積の図示である。[2]

「あみだくじ」は置換を隣接互換の積で表したものである。
たとえば巡回置換は

(1 2 3 4) ＝ (2 3)(1 2)(3 4)(2 3)(1 2)(2 3)(3 4)

となる。

これに対応する「あみだくじ」は図のようになる。



これをＪにより検証してみよう。まず、巡回置換として行う。

```
   (< <: 1 2 3 4) C. 1 2 3 4
2 3 4 1
```

隣接互換の積として右辺を計算してみると、以下のようになった。

```
   amida =: <: L:(0) 2 3;1 2;3 4;2 3;1 2;2 3;3 4
   amida
+---+---+---+---+---+---+---+
|1 2|0 1|2 3|1 2|0 1|1 2|2 3|
+---+---+---+---+---+---+---+
   amida C. 1 2 3 4
2 3 4 1
```

これを、直接置換で行ってみると次のようになる。

```
   C. (< <: 1 2 3 4)
1 2 3 0
   1 2 3 0 C. 1 2 3 4
2 3 4 1
```

４．置換操作に関しての数学表記とＪの実行

前報[1]にも述べたが、数学書の表記とＪの実行にはいくつかの異同がある。

・数学書：１オリジンとＪ：０オリジン

・Ｊの品詞からの発想　置換操作（置換群の元）：動詞
　　置換で変化する集合の値：名詞

これに加えて群の元の積（＝連続操作）の順序に関して微妙な違いが起きる。

先の「あみだくじ」の例で検討してみる。集合の値には、数字でなく「A B C D」の文字を用いた方がよい。

まず、数学書の表記に対して、手動でおこなう。

巡回置換　(1 2 3 4) とは、集合「A B C D」を次々と左に移動、回転させることに他ならないから、結果は「B C D A」となることは明らかである。

互換の連続操作として「あみだくじ」の図の通りに「左から右へ」と操作してみると、ABCD → ACBD → CABD → CADB → CDAB → DCAB → DACB → DABC となる。

しかし、これを「右から左へ」と操作すると、

ABCD → ABDC → ADBC → DABC → DBAC → DBCA → BDCA → BCDA となる。

さて、「あみだくじ」を実際にたどってみると「A B C D」は「B C D A」となる。すると「あみだくじ」とは何を図示しているのだろうか？

以上の違いを「左から右へ」の合成を「置換としての合成」
　　「右から左へ」の合成を「関数としての合成」
と区別している。実際にはよく注意しなければならない。

Ｊで実行したときは、つぎのようになる。

```
   amida
+---+---+---+---+---+---+---+
|1 2|0 1|2 3|1 2|0 1|1 2|2 3|
+---+---+---+---+---+---+---+
   amida C. 'ABCD'
BCDA
```

これは、「右から左へ」の連続実行に等しい。

```
   ](<2 3)C.'ABCD'
ABDC
   ](<1 2)C.(<2 3)C.'ABCD'
ADBC
   ](<0 1)C.(<1 2)C.(<2 3)C.'ABCD'
DABC
   ](<1 2)C.(<0 1)C.(<1 2)C.(<2 3)C.'ABCD'
DBAC
   ](<2 3)C.(<1 2)C.(<0 1)C.(<1 2)C.(<2 3)C.'ABCD'
DBCA
   ](<0 1)C.(<2 3)C.(<1 2)C.(<0 1)C.(<1 2)C.(<2 3)C.'ABCD'
BDCA
   ](<1 2)C.(<0 1)C.(<2 3)C.(<1 2)C.(<0 1)C.(<1 2)C.(<2 3)C.'ABCD'
BCDA
```

置換群とは「席替えをするのに、イスは動かさずに、その番号札をだけをやりとりして、最後に決まった札の番号で、イスに座る」やり方だ、と私は解している。

Jのプログラムリスト

```
NB. 直接置換表示の偶奇性
odd_even =: 3 : 0
if. (2 | # C. 1 + y.)
  do.   'odd'
  else. 'even'
end.
)

NB. 巡回置換表示から互換表示への変換
cyc2swap =: 3 : 0
CY =. /:~ L:0 <"(1) ({. y.) ,. (}. y.)
)

NB. 巡回置換表示から隣接互換表示への変換
cyc2adjswap =: 3 : 0
CY =. /:~ L:0 <"(1) ({. y.) ,. (}. y.)
; adjswap L:0 CY
)

NB. 隣接互換への変換
adjswap =: 3 : 0
'I J' =. y.
R =. ''
k =. 0
while. k < (<: J)
  do.
    wr (I+k),(I+1+k)
    R =. R, < (I+k),(I+1+k)
    k =. k + 1
end.
if. I = J -1 do. R return. end.
i =. J - 1
j =. J - 2
k =. 0
while. k < 5
   do.
    wr (i-k), (j-k)
    R =. R, < (i-k),(j-k)
    if. I = (j-k) do. break. end.
    k =. k + 1
end.
R
)
```