JAPLA 研究会資料 2011/6/25

正規表現により、パスカル三角形をピラミッド型で表示する

西川　利男

はじまり

最近、アイバーソンの経歴などを調べるため、志村正人氏よりご提供された VECTOR の Remembering Iverson Memorial 特集誌を見ていた。

そのとき、偶然、Eugene McDonnell による ″At Play with J: Token Counting″ というレポートの中に Pyramigram(ピラミッド表示)なる記載があるのを見つけた。

これは、Linda Alvord 女史が 1980 年の APL Quote Quad に出した問題に出した APL プログラムが発端だそうだ。ちなみに、彼女は米国、ニュージャージーの高校の先生で、シドニーの APL88 国際会議のとき、私と一緒に写真におさまったことがある。

以下のようなピラミッド型の文字パターンをなるべく少ないコーディングでやるにはどうするか、という問題である。

元々は α－ω 形式の APL のコードがあったようだが、これに対応した Roger Hui によるトリッキーな J のコードが載っている。

```
PG =: ([:i.[:-])(|.″0 1)1j1″_ #″0 1(([:/:(([:-/\]-~[:|.[:i.[:+:])#([:i.[:
+:])*[:*:])+[:?~[:*:]){[:,(],])
(('ABCDEFGHIJKLMNOPQRSTUVWXYZ'″_{~]?26″_){.~[:-.[:+:])$~],[:+:])$~],]
```

```
   PG 5
    S
   Q S
  L S Q
 S J Q L
J L S D Q
```

その後、Hui によりもっと短い J のコードが出された。

```
h =: /: # ? #
pyr =: i. &. - @ # |. ″ _1 [: 1j1 & # @ h ¥ h
```

```
   pyr 'qwert'
    t
   t w
  w t q
 q w r t
r q e t w
```

McDonnell の VECTOR のレポートはこれについてコメントしたものである。

私の今回の発表は、これを参考にパスカル三角形に適用してみた。ところがパスカル三角形で、１桁の数のときはよいが、２桁以上になると、ピラミッド表示はうまく

いかない。いろいろやってみたが、Jの正規表現プログラミングで行うことで、きれいにピラミッド表示が出来た。

１．パスカル三角形と素朴なピラミッド表示のＪコード

パスカル三角形の計算は、Jではたった一行で行える。

```
   (|:@(!/~)@i.) 5
1 0 0 0 0
1 1 0 0 0
1 2 1 0 0
1 3 3 1 0
1 4 6 4 1
   (|:@(!/~)@i.) 6
1 0  0  0 0 0
1 1  0  0 0 0
1 2  1  0 0 0
1 3  3  1 0 0
1 4  6  4 1 0
1 5 10 10 5 1
```

したがって、これを文字列化して、Hui の関数 pyr を使えば、簡単に出来るはずである。次のように関数として定義した。

```
NB. Pascal Triangle displayed in Pyramid =============
NB.   by Toshio Nishikawa, 2011/5/22
pas =: 3 : 0
p =. i. y.
|: p !/p
)
pas1 =: 3 : 0
″: pas y.
)
NB. simple version OK less than 5
pas2 =: 3 : 0
(i. &. - y.) |. ″ _1 pas1 y.
)
NB.   '*' subs 'b' 'abcd'
NB. a*cd
subs=: [. & ((((e.&) ((# i.@#)@)) (@])) })
pas3 =: ((' ' subs '0')″1) @ pas2

   pas3 5
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

実行すると、ここまでは良い。しかし、6 以上となると、数値の 10 がスペースになってしまったり、先の文字のピラミッド表示のようにはいかない。

これを解決するには、どうしても正規表現プログラミングに頼らざるをえない。

２．正規表現プログラミングによるピラミッド表示のＪコード

```
NB. revised for 6 more ====================
require 'regex'

NB. zero to space =============================
zerotosp =: 3 : 0
('[[:space:]]0[[:space:]]*';' ') rxrplc y.
)
pas10 =: (zerotosp"(1))^:(3) @ pas1
```

正規表現は、次のように利用される。パターン'[[:space:]]0[[:space:]]*'、つまり計算で得たパスカル三角形の数値を文字化したデータに対して、空白部分の数値0だけを正規表現関数rxrplcにより空白に置き換える。こうすれば、数値10はそのまま保たれる。

```
pas11 =: 3 : 0
(i. &. - y.) |. " _1 pas10 y.
)
```

ここで、Huiからの次のコードがピラミッド表示のポイントになる。ここで得られた値にそって、各行を右シフトすることによってピラミッド表示がなされる。

```
   i. _6
5 4 3 2 1 0
   (i. &. -) 6
_5 _4 _3 _2 _1 0

    pas11 6
      1
     1 1
    1 2  1
   1 3  3  1
  1 4  6  4 1
 1 5 10 10 5 1
```

さらに、大きい例でやってみる。

```
   pas11 10
         1 0
        1 1  0
       1 2  1
      1 3  3  1
     1 4  6  4   1
    1 5 10 10   5   1
   1 6 15 20  15   6  1
  1 7 21 35  35  21  7  1
 1 8 28 56  70  56 28  8 1
```

```
1 9 36 84 126 126 84 36 9 1
```

一見、良いように見えるが、ピラミッドが左右対称でなくゆがんでいる。さらにパスカル三角形の値の区切りが空白１つであったり、２つであったり目障りである。

せっかくなので、もう一歩改善してみよう。

３．最終版のパスカル三角形のピラミッド表示のＪコード

ここでの改良点は区切りの２つ以上のスペースを１つにする。

```
NB. adjust over two spaces to one space ========
PAT =: ('[[:digit:]]+([[:space:]][[:space:]]+)[[:digit:]]+')
space_adj =: 3 : 0"(1)
pq0 =. PAT rxmatches y.
if. 0 = #pq0 do. y. return. end.
((PAT;,1);' ') rxrplc y.
)
```

各行をセンターリングする。

```
NB. centering ==================================
CENPAT =: '1([[:space:]][[:digit:]]+)*'
cent =: 3 : 0"(1)
LINE =. # y.
CP =. CENPAT rxmatches y.
LEN =. {:,{."2 CP
>. -: LEN - LINE
)

pascent =: cent |."_1 ]

pascal =: pascent@(space_adj^:(3) @(pb@pa))

   pascal 10
             1
            1 1
           1 2 1
          1 3 3 1
         1 4 6 4 1
       1 5 10 10 5 1
     1 6 15 20 15 6 1
    1 7 21 35 35 21 7 1
  1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
```

```
NB. Pascal Triangle in Pyramigram

NB. Pyramigram
NB.   by Eugene McDonnel, VECTOR Vol.22, No.3, p.49-54

h =: /: # ? #

pyr =: i. &. - @ # |. " _1 [: 1j1 & # @ h ¥ h

py0 =: i. &. - @ # |. " _1 [: 1j1 & # @ ] ¥ ]

NB. Pascal Triangle displayed in Pyramid =============
NB.   by Toshio Nishikawa, 2011/5/22
pas =: 3 : 0
p =. i. y.
|: p !/p
)

pas1 =: 3 : 0
": pas y.
)

NB. simple version OK less than 5
pas2 =: 3 : 0
(i. &. - y.) |. " _1 pas1 y.
)

NB.   '*' subs 'b' 'abcd'
NB. a*cd
subs=: [. & ((((e.&) ((# i.@#)@)) (@])) })
pas3 =: (' ' subs '0')"1) @ pas2

NB.   pas3 5
NB.     1
NB.    1 1
NB.   1 2 1
NB.  1 3 3 1
NB. 1 4 6 4 1

NB. revised for 6 more ====================
require 'regex'
pas10 =: (zerotosp"(1))^:(3) @ pas1

pas11 =: 3 : 0
(i. &. - y.) |. " _1 pas10 y.
)
```

```
NB.    pas11 6
NB.       1
NB.      1 1
NB.     1 2  1
NB.    1 3  3  1
NB.   1 4  6  4 1
NB. 1 5 10 10 5 1

NB. Pascal Triangle in Pyramid ================
NB.    by Toshio Nishikawa
NB.    the latest version 2011/5/23
NB.  using regular expression

pa =: |: @ (i. !/ i.)

pb0 =: (' ' subs '0')"(1) @(":)

require 'regex'
NB. zero to space ============================
zerotosp =: 3 : 0
('[[:space:]]0[[:space:]]*';' ') rxrplc y.
)

pb =: (zerotosp"(1))^:(4) @(":)

NB. display in pyramid
pyx =: i. &. -
pasc =: pyx |."_1 (pb@pa)

NB. adjust over two spaces to one space ========
space_adj =: 3 : 0"(1)
PAT =: ('[[:digit:]]+([[:space:]][[:space:]]+)[[:digit:]]+')
pq0 =. PAT rxmatches y.
if. 0 = #pq0 do. y. return. end.
((PAT;,1);' ') rxrplc y.
)

pascal0 =: space_adj^:(4) @ pasc

NB.    pascal 10
NB.          1
NB.         1 1
NB.        1 2 1
NB.       1 3 3 1
NB.      1 4 6 4 1
NB.     1 5 10 10 5 1
NB.    1 6 15 20 15 6 1
NB.   1 7 21 35 35 21 7 1
```

```
NB.  1 8 28 56 70 56 28 8 1
NB. 1 9 36 84 126 126 84 36 9 1

NB. final version with centering ================
NB.   2011/5/25
CENPAT =: '1([[:space:]][[:digit:]]+)*'

cent =: 3 : 0"(1)
LINE =. # y.
CP =. CENPAT rxmatches y.
LEN =. {:,{."2 CP
>. -: LEN - LINE
)

pascent =: cent |."_1 ]

pascal =: pascent@(space_adj^:(3) @(pb@pa))

wr =: 1!:2&2
pasd =: 3 : 0
PA =. y.
wr cent PA
(cent PA) |."_1 PA
)
```