材料の取りあわせ問題
プログラムと解説

SHIMURA Masato
http://japla.sakura.ne.jp

2014 年 12 月 4 日

目次

1 材料取りの問題

JAPLA の研究会で山本洋一氏から次のような材料取りの問題が出された。

> 工作材料寸法が決まっている場合に、材料を何本入れ、どう切断すれば適切な切断ができるか

山本氏は Excel でチャレンジしておられるが、本数が増えると Excel を風船のように膨らまさなければならず、Excel の難問といえよう。

丁度魔方陣で組み合わせの問題で PC 苛めをしていたので安請け合いをしてしまったが、順次組み合わせを作成し、迷路のようなループと配列の拡張を行わなければならない。

*1

2　組み合わせ

最初に組み合わせの数学と J のプリミティブの概略を整理しておこう

1. 階乗　　$!4 = 4 \times 3 \times 2 \times 1$

```
   ! 4
24
```

2. 辞書式順序 A.
   - 単項の A. (Anagram Index) 組み合わせの位置（順番）を示す

```
      A.  0 1 3 2 ,: 2 3 0 1
1 16                    NB.  （０オリジンで）１番目と１６番目
```

```
tap=: i.@! A. i.    NB. Table of Permutations
```

3. A. を用いたすべての組み合わせ

```
tap=: i.@! A. i.    NB. Table of Permutations
```

```
  tap 3
0 1 2
0 2 1
1 0 2
1 2 0
2 0 1
2 1 0
```

4. 順列 Sequence without repetition

$$ {}_nP_m = n \cdot (n-1) \cdot (n-2) \cdot (n-m+1) = \frac{n!}{(n-k)!} $$

5. 順列の表

   順列の表はは順序を保持したままで辞書から取り出し同じものを掃き出す。

   ${}_4P_2$ ならば *tap 4* から最初の *2* 列を取り出し同じものを落す。

*1 急激に拡張するスクリプトの迷路上を走り回るゴキブリを追い掛け回す破目になった。組み合わせに取り組むには覚悟が必要。

```
  ~. 2{."1 tap 4
```

```
0 1
0 2
0 3
1 0
1 2
1 3
2 0
2 1
2 3
3 0
3 1
3 2
```

右は *cut* で *2* 行を区分した。重複は $4 \times 3 \times 2$ の *2* であり、${}_nP_m = n \times (n - m + 1)$ になる

```
   (’’;1 0 1 0)<;.1 tap 4
+---+---+
|0 1|2 3|
|0 1|3 2|
|0 2|1 3|
|0 2|3 1|
|0 3|1 2|
|0 3|2 1|
|1 0|2 3|
|1 0|3 2|
|1 2|0 3|
|1 2|3 0|
|1 3|0 2|
|1 3|2 0|
|2 0|1 3|
|2 0|3 1|
|2 1|0 3|
|2 1|3 0|
|2 3|0 1|
|2 3|1 0|
|3 0|1 2|
|3 0|2 1|
|3 1|0 2|
|3 1|2 0|
|3 2|0 1|
|3 2|1 0|
+---+---+
```

6. 組み合わせ *Combination*

$$ {}_nC_m = \frac{n \cdot (n-1) \cdot (n-2) \cdots (n-m+1)}{m \cdot (m-1) \cdots 1} $$

$$ \binom{n}{m} = \frac{{}_nP_m}{m!} $$

```
2!4
6
```

$$ {}_4C_2 = \frac{4 \times 3}{2 \times 1} $$

7. 組み合わせの表

   組み合わせは順序を問題にしないので辞書から取り出したものを行方向にソートしたうえで、同じものを落とす

```
   ~. /:~ ("1) 2{."1 tap 4
0 1
0 2
0 3
1 2
1 3
2 3
```

8. 組み合わせの表を求めるスクリプト

```
nCm=: 3 : 0
NB. calc table of  n C m
NB. nCm 5 3  <---> j form is 3 ! 5
'n0 m0'=. y
  ~. /:~"(1)  m0{."1 tap n0
)
```

```
   nCm 4 2
0 1
0 2
0 3
1 2
1 3
2 3
```

3　分割数

4 個のみかんを 4 人で分ける方法。一個づつ平等から独り占めまで 5 通りの方法がある。以前作った *partition* 関数を活用する。

http://japla.sakura.ne.jp の *Essay* の分割数を参照

```
partition 4
```

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

4　材料取り問題のスクリプトと解説

- 迷路のようなロジックを切り分けやすいようにスクリプトはいくつかに分ける
- 最適化は最後に検討できるようにし、次の部分ごとにスクリプトを作成する
  - 組み合わせ
  - 組み合わせに材料取りを落とし込み、採寸可能なものを取り出す
  - 最適化の資料を作成する

4.1　分割数を求める

5 の例。末尾の方からチェックする

```
   |.  partition 5
+-+---+---+-----+-----+-------+---------+
|5|4 1|3 2|3 1 1|2 2 1|2 1 1 1|1 1 1 1 1|
+-+---+---+-----+-----+-------+---------+
```

```
   5 part_combination 5
+---------+
|0 1 2 3 4|
+---------+
   4 1 part_combination 5
+-------+-+
|0 1 2 4|3|
+-------+-+
|0 1 3 4|2|
+-------+-+
|0 2 3 4|1|
+-------+-+
|1 2 3 4|0|
+-------+-+
```

```
   3 2 part_combination 5
+-----+---+
|0 1 2|3 4|
+-----+---+
|0 1 3|2 4|
+-----+---+
|0 1 4|2 3|
+-----+---+
|0 2 3|1 4|
+-----+---+
|0 2 4|1 3|
+-----+---+
|0 3 4|1 2|
+-----+---+
|1 2 3|0 4|
+-----+---+
|1 2 4|0 3|
+-----+---+
|1 3 4|0 2|
+-----+---+
|2 3 4|0 1|
+-----+---+
```

入力　\:˜で分割数の入力を降順に並べる

```
   part_combination=: 4 : 0
NB. x is  using partition i.g. 4 3 1
NB. y is number of material  ex. 8
NB. Usage: 3 1 1 part_combination 5
NB. - input x y-----------------------------------------
NB. i.g.  5 3 2 1 //is part of 10
partorder=: \:˜ x                    NB. order(part) of partition(downsort)
size =. y
```

前処理　分割 1 と *2* 以上 *(bigpart*）に区分。*bigpart* はループ回数

nlist　*nlist* は ${}_nC_m$ の *n* のリストを作成。*suffix*　\. を活用

```
   +/\. 3 2 2 1
8 5 3 1

                              NB. size is fix
NB. nnumber mnumber  is n and m  of nCm
NB. check input------------------
NB. procedure part of bignumber
bigpart=:  +/  2 <: partorder                NB. number of not  1
nlist=: +/\. partorder                       NB. siffix  5 3 1  -> tane n
```

前処理２　入力ミス, 全て *1* のケース

```
NB. ---------------------------------------------------------
if. -. size = +/ partorder do. 'number-over' goto_end. end. NB. not equal
if. 0= bigpart do. ans=. {@> i.y  goto_end. end.     NB. 1 1 1 1 1
NB. ----------------------------
```

${}_nC_m$　最初の n, m をセットし、${}_nC_m$ を計算する

${}_5C_3$,${}_3C_2$ と逐次 n, m をカウンターで管理する。

```
ans=. <''
nnumber=:  {. nlist                NB. take n
mnumber=:  {. partorder            NB. take m
stem=: { combi0=: nCm nnumber , mnumber
```

n, m が同数 ${}_3C_3$,${}_2C_2$ など自由度がなくなるときはループを外れ、後処理に回る

```
NB. ---procedure--big part----------------------------
for_ctr0.  i.  bigpart  do.                  NB. loop within bignumber
    ctr=: ctr0
    NB. ------skip loop case n=m-------------------------------------
    if. nnumber = mnumber  do. ans=.ans, stem ,.stem find_rest (L:0) size
        goto_skip. end.
```

ループの中心　.

- *combi0*　最初の ${}_nC_m$
- *combi1*　2 回目以降の ${}_nC_m$
- *stick combi0* を *combi1* の数だけ拡大するための整形
- *stem* ${}_nC_m$,$_n C_m$,$_n C_m$ を拡大しながら逐次（計算上シームレスに）取り込む（幹）

```
NB. ------------------------------------------------------------
    select.   0 < ctr0                          NB. 1st or 2nd
      case. 0 do. remain =: ({combi0) find_rest (L:0) size
      case. 1 do. combi1=: nCm nnumber,mnumber
                  stick=: >{ L:0 combi1 {L:0 remain
                  stem=: stem expand_box stick
                  remain =: stem find_rest (L:0) size
    end.
```

n, m の管理　n, m の数を次に進める。

```
    nnumber=: (>:ctr0) { nlist                  NB. take n for next
    mnumber=: (>:ctr0) { partorder              NB. take m for next
end.
```

解　.

```
ans=. ans,< stem,.stem find_rest (L:0) size
```

1 $\cdots$ 1 の部分　.

```
NB. ---------procedure part 1 1 1...-----------------
label_skip.
      if. 1 <: +/ partorder e. 1 do.            NB. managing 1..1
        tmp=. stem find_rest L:0 size
           if. 1< # > {.tmp do. tmp=.>{@> L:0 tmp end.  NB. 1|1|1
        ans=.stem ,. tmp
      end.
```

リサイズ　　*stem* で組み合わせの区分線を計算を容易にするため除いていたのを入れなおす

- *cut* <;.1
- 列のカットの指標。行ならば前の” に入れる

```
          ('';1 0 0 0 1 0 1)<;.1 i.5 7
+-----------+-----+--+
| 0  1  2  3| 4  5| 6|
| 7  8  9 10|11 12|13|
|14 15 16 17|18 19|20|
|21 22 23 24|25 26|27|
|28 29 30 31|32 33|34|
+-----------+-----+--+
```

```
NB. --------------------------
partorder resize_stem_sub  }.ans
NB. }. ans
label_end.
)
```

4.2　採寸可能な組み合わせを求める

- 材と寸法を指定する
  goods=: 6500;4000 3000 2000 1000 500
- *5 3* の組み合わせ
- サイズをチェックして可能な組み合わせを抜き出す

```
   ] a=. 3 2 part_combination 5
+-----+---+
|0 1 2|3 4|
+-----+---+
|0 1 3|2 4|
+-----+---+
|0 1 4|2 3|
+-----+---+
|0 2 3|1 4|
+-----+---+
|0 2 4|1 3|
+-----+---+
|0 3 4|1 2|
+-----+---+
|1 2 3|0 4|
+-----+---+
|1 2 4|0 3|
+-----+---+
|1 3 4|0 2|
+-----+---+
|2 3 4|0 1|
+-----+---+
```

```
   goods select_inner a
+--------------+---------+
|4000 2000 500 |3000 1000|
+--------------+---------+
|4000 1000 500 |3000 2000|
+--------------+---------+
|3000 2000 1000|4000 500 |
+--------------+---------+
|3000 2000 500 |4000 1000|
+--------------+---------+
|3000 1000 500 |4000 2000|
+--------------+---------+
```

```
NB. ----------------------------------------
select_inner=: 4 : 0
NB. Usage:
NB.  goods=. 65000;4000 3000 2000 1000 500
NB.  goods select_inner 4 1 part_combination 5
'material piece'=: x
pindex=: y   NB.  m part_combination n
cand=: pindex { L:0 piece
tmp=: > material >: L:0 +/ L:0  cand
select. +/ */"1 tmp
  case. 0 do. 'nothing in this partition '
```

```
  fcase.  do. (*/"1 tmp) # cand
end.
)
```

4.3 最適化への解析

${}_5C_3$,2 C_2 の場合は *5* ケースとれる。どれが最適化選ぶ材料を提供している。この組み合わせでは余材の合計や率は全て同じとなる。

```
   goods find_fitting a
+--------------+---------+----+----+----+
|4000 2000 500 |3000 1000|0   |2500|2500|
+--------------+---------+----+----+----+
|4000 1000 500 |3000 2000|1000|1500|2500|
+--------------+---------+----+----+----+
|3000 2000 1000|4000 500 |500 |2000|2500|
+--------------+---------+----+----+----+
|3000 2000 500 |4000 1000|1000|1500|2500|
+--------------+---------+----+----+----+
|3000 1000 500 |4000 2000|2000|500 |2500|
+--------------+---------+----+----+----+
  5C3             2C2       余材  余材  残の計
```

${}_5C_4$ の場合は *1* ケース可能。最適化は目で追うこととする。

```
   goods find_fitting 4 1 part_combination 5
+------------------+----+-+----+----+
|3000 2000 1000 500|4000|0|2500|2500|
+------------------+----+-+----+----+
 5C4                 1C1          残の計
```

```
find_fitting=: 4 : 0
NB. Usage: (6500;4000 3000 2000 1000 500) find_fitting a
'material piece'=: x
```

```
NB. y is result of a=. m part_combination y
cand=. x select_inner  y
try. tmp1=: material - L:0  +/ L:0 cand
catch. 'nothing in this partition'
goto_end.
end.
cand,.tmp1,.{@>+/"1 > tmp1
label_end.
)
```

5　組み合わせの組み合わせ

${}_6C_2$ からコンビネーションの先頭の倍数で急激に増殖する

$$ {}_2C_2 = 2 $$

$$ {}_4C_2 \times_2 C_2 = 6 $$

$$ {}_6C_2 \times_4 C_2 \times_2 C_2 = 90 $$

$$ {}_8C_2 \times_6 C_2 \times_4 C_2 \times_2 C_2 = 2520 $$

$$ {}_{10}C_2 \times_8 C_2 \times_6 C_2 \times_4 C_2 \times_2 C_2 = 113400 $$

```
   2!2                    #  2  part_combination 2
2                         2
   2!4                    # 2 2  part_combination 4
6                         6
   2!6                    # 2 2 2 part_combination 6
15                        90
   2!8                    # 2 2 2 2 part_combination 8
28                        2520
   2!10                   # 2 2 2 2 2  part_combination 10
45                        113400
```

References

Script yamamoto_combination.ijs
スクリプトは次から *DL* できます。　*japla.sakura.ne.jp*
J 言語は次から入手できます。*www.jsoftware.com*