第４部：『魔方陣の最新研究』 /摂田 寛二
第４章： 解説『魔方陣研究の諸技法』II
第 12 節：『新オイラー法』の特質と応用（2003 年版清書稿）

０．この節の目的

今回は，今までとは少し変わったことをしてみたい。『新オイラー法』の特質を論じ，またその新たな応用可能性を調べたい。

今までに，様々な次数・タイプの魔方陣を『新オイラー法』で作って見せた。この方法の適用できる範囲を調べるためだった。

低次の魔方陣では，重要な魔方陣のほとんど全てを作れることがわかった。七，八および九次でも，珍しいタイプの珠玉の魔方陣をいろいろと作れることがわかった。しかも，どの場合でも，素晴らしい実行スピードを示して，我々の期待に応えてくれた。

新オイラー法は，惚れぼれするくらいにパワフルな構成法である。

いったい何故なのか？　この方法には，何か問題点というものは無いのか？

１．『新オイラー法』の特質

（１）**新オイラー法**は，Ｎ進法分解図を使う。

三，五および七次は，そのまま三，五，七進法を用いるが，四次および八次は，二進法を用いる。九次は，三進法を使う。六次だけは例外で，「新オイラー法」といえども歯が立たない。何をしても汎六方陣や対称六方陣を作ることはできない。

なぜＮ進法が有効なのか？

その理由の１つは，正方配列そのものの構造にあると，思われる。

五次と七次の場合で説明しよう。

正方配列と言うのは，５×５，または７×７のマス目のことだが，五次の場合，各々の場所を二次元座標で表わすと，下図のようになる。この配列に０～24 の整数を順に入れて行き，それを五進法で表わすと，右図のようになる。内容が全く同じなことに，お気付きだろうか？

位置情報と数値情報を同じ形式で扱えると言うのは，何かと便利なものだ。実際に，それらを相互に入れ替える変換法も存在する。

また，汎魔方陣の汎対角線に見られる特有の円環構造も，Ｎ進法の数値（特に下位数）が常にｎの累乗で割った剰余であるために，うまく表現してくれてそれに対応できる。

＊　五方陣図　＊

［正方配列二次元座標］

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

［五進法数値］

00	01	02	03	04
10	11	12	13	14
20	21	22	23	24
30	31	32	33	34
40	41	42	43	44

［十進法数値］

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

［五進法表記図］ /N5i

00	01	02	03	04
10	11	12	13	14
20	21	22	23	24
30	31	32	33	34
40	41	42	43	44

［五進法分解図］ /D5i　H/　L/

0	0	0	0	0	0	1	2	3	4
1	1	1	1	1	0	1	2	3	4
2	2	2	2	2	0	1	2	3	4
3	3	3	3	3	0	1	2	3	4
4	4	4	4	4	0	1	2	3	4

＊　七方陣図　＊

［七進法表記図］

```
/N7i
00 01 02 03 04 05 06
10 11 12 13 14 15 16
20 21 22 23 24 25 26
30 31 32 33 34 35 36
40 41 42 43 44 45 46
50 51 52 53 54 55 56
60 61 62 63 64 65 66
```

［七進法分解図］

```
/D7i    High/          Low/
0 0 0 0 0 0 0   0 1 2 3 4 5 6
1 1 1 1 1 1 1   0 1 2 3 4 5 6
2 2 2 2 2 2 2   0 1 2 3 4 5 6
3 3 3 3 3 3 3   0 1 2 3 4 5 6
4 4 4 4 4 4 4   0 1 2 3 4 5 6
5 5 5 5 5 5 5   0 1 2 3 4 5 6
6 6 6 6 6 6 6   0 1 2 3 4 5 6
```

Ｎ進法分解図は，上位数と下位数を別の面に分けて書いたものだが，対応する場所の数値を次の式で結び付けると，元の十進法表記に戻る。

Ｖｎ ＝Ｈｎ＊５＋Ｌｎ　（n=1，2，3，...，24，25）... 五方陣

Ｖｎ ＝Ｈｎ＊７＋Ｌｎ　（n=1，2，3，...，47，49）... 七方陣

上位／下位数を別の面に分けたことで，上位／下位だけの数の並び方が見やすくなる。『ラテン方陣』特有の数字の美しい並び方が直に見て取れる。

さて，上の分解図を良く見てほしい。五次も七次も，位置情報をそのまま数値情報に読み替えて，それをＮ進法で表わしただけのものだが，上位面も下位面も，汎対角線の全てが同じ内容｛0，1，2，3，4｝の『ラテン型』に既になっているではないか。また上位面は縦の列が全て**ラテン型**，そして下位面は横の行が全て**ラテン型**に既になっているではないか。

これならば，規則的な数字の並び替えをするだけで，すぐに上位面／下位面ともに**ラテン型**の『オイラー方陣』を作れる可能性があることがわかる。

（２）「新オイラー法」は，先ずＮ進法分解図を構成し，それを用いて目的とする方陣を計算・合成する。

もちろん，その分解図は，上位面／下位面ともに**ラテン型**でなければならないが，その他にも目的方陣特有の条件を付加して設計・構築する。

この時に暗黙のうちに前提にしている重要な条件がある。

分解図の各面が，目的方陣と同じ性質・条件を分有・共有しているということ。

例えば，行列や汎対角線の定和はもとより，対称補数条件や正方連結条件の成立等々。

オイラー方陣を構成する時には，分解図の各面でも，それら全ての条件が成り立つと仮定する。その上で，各行列の内容にも注文を付けて，数字の構成が｛0，1，2，3，4｝，或いは｛0，1，2，3，4，5，6｝となるようにする。同じ数字を何度も使えないのだ。

『完全オイラー方陣』を構成する場合には，汎対角線の全てについてもこれを要求する。

これは，考えてみれば実に厳しい要求で，だから，**完全オイラー方陣**は，そう幾つも作れない。作れば，どれもが稀少価値の高い珠玉の魔方陣となる。とりわけ分解図の場合，構成できる「単位面」が，ほんの数個から数十個に限られる。

オイラー方陣にこだわりさえしなければ，分解図の各面でそんな厳しい条件は必ずしも成り立たなくても良い。全体として，各部の定和が成り立っていれば良い。

実際七次以上になると，「ノン - オイラー型」の魔方陣の方が「オイラー型」より断然多くなる。

ところが，四次と五次では，全ての汎魔方陣が**完全オイラー方陣**なのである。どうやっても，「ノン - オイラー型」の汎魔方陣は作れない。これも不思議の１つだ。

だから，四次と五次だけ見ていれば良かった時代には，**完全オイラー方陣**が全ての汎魔方陣の「本質」つまり「運命」であると，実感することができた。

だが，七次以上になると，事情が違って来る。全てが複雑に進化し，系・類・種が独立したり分化したりする。**完全オイラー方陣**は，稀少価値の高い「珍品」でのみ実現することになる。

（３）**新オイラー法**は，自分で作った「単位面」を組み合わせて**Ｎ進法分解図**に見立てる。そして，掛け算・足し算を用いて計算・合成して目的方陣を作る。

この時にまた，暗黙のうちに前提にしている事柄がある。

単位面は，どれも同格で，分解図のどの位取りにも来ることができる。だから，任意の組み合せが可能である。そして，その組み合せの多様さが，目的方陣の多様さ（つまり多数の解）の根拠であるいうことを暗に仮定している。実に大胆なことを主張しているのだ。

これは，結果から見て，事実その通りなのだが，実際には幾つか制限条件が存在する。

どれをどう自由に組み合わせても良いわけではない。結果が全て正解になるわけではないからだ。

どのケースでも，同一の**単位面**を繰り返し使ったり，お互いに**補数面**となるような２面を同時に組み合わせて使用してはいけない。さもないと誤答が生じてしまう。

そこで，例の『参照表』を用意する。相似度と補数度を計測した数値表だ。

任意の２面を組み合わせる前に，この表を見に行き，不適格な組み合せを避け，最適の数値の組み合せを選ぶようにする。

＊＊　両立型の対称汎五方陣を「新オイラー法」で作る　＊＊

［構成単位面］

```
        1/          2/          3/          4/
0 2 4 1 3   0 2 4 3 1   0 4 1 2 3   0 4 3 2 1
4 1 3 0 2   4 3 1 0 2   2 3 0 4 1   2 1 0 4 3
3 0 2 4 1   1 0 2 4 3   4 1 2 3 0   4 3 2 1 0
2 4 1 3 0   2 4 3 1 0   3 0 4 1 2   1 0 4 3 2
1 3 0 2 4   3 1 0 2 4   1 2 3 0 4   3 2 1 0 4
        5/          6/          7/          8/
1 2 3 0 4   1 2 3 4 0   1 3 0 2 4   1 3 4 2 0
3 0 4 1 2   3 4 0 1 2   2 4 1 3 0   2 0 1 3 4
4 1 2 3 0   0 1 2 3 4   3 0 2 4 1   3 4 2 0 1
2 3 0 4 1   2 3 4 0 1   4 1 3 0 2   0 1 3 4 2
0 4 1 2 3   4 0 1 2 3   0 2 4 1 3   4 2 0 1 3
        9/          10/         11/         12/
3 1 0 2 4   3 1 4 2 0   3 2 1 0 4   3 2 1 4 0
2 4 3 1 0   2 0 3 1 4   1 0 4 3 2   1 4 0 3 2
1 0 2 4 3   1 4 2 0 3   4 3 2 1 0   0 3 2 1 4
4 3 1 0 2   0 3 1 4 2   2 1 0 4 3   2 1 4 0 3
0 2 4 3 1   4 2 0 3 1   0 4 3 2 1   4 0 3 2 1
       13/          14/         15/         16/
4 0 1 2 3   4 0 3 2 1   4 2 0 1 3   4 2 0 3 1
2 3 4 0 1   2 1 4 0 3   0 1 3 4 2   0 3 1 4 2
0 1 2 3 4   0 3 2 1 4   3 4 2 0 1   1 4 2 0 3
3 4 0 1 2   1 4 0 3 2   2 0 1 3 4   2 0 3 1 4
1 2 3 4 0   3 2 1 4 0   1 3 4 2 0   3 1 4 2 0
```

[Count of Layer Units = 16]

［参照表：　任意の２面の相関度を示す数値表（２値合数表）］

```
SC|  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
--+-------------------------------------------------
 1| 25 15  5  5  5  5  5  5  5  5  5  5  5  5 15 25
 2| 15 25  5  5  5  5  5  5  5  5  5  5  5  5 25 15
 3|  5  5 25 15  5  5  5  5  5  5  5  5  5 15 25  5  5
 4|  5  5 15 25  5  5  5  5  5  5  5  5  5 25 15  5  5
 5|  5  5  5  5 25 15  5  5  5  5 15 25  5  5  5  5
 6|  5  5  5  5 15 25  5  5  5  5 25 15  5  5  5  5
 7|  5  5  5  5  5  5 25 15 15 25  5  5  5  5  5  5
```

```
 8|  5  5  5  5  5  5 15 25 25 15  5  5  5  5  5  5
 9|  5  5  5  5  5  5 15 25 25 15  5  5  5  5  5  5
10|  5  5  5  5  5  5 25 15 15 25  5  5  5  5  5  5
11|  5  5  5  5 15 25  5  5  5  5 25 15  5  5  5  5
12|  5  5  5  5 25 15  5  5  5  5 15 25  5  5  5  5
13|  5  5 15 25  5  5  5  5  5  5  5  5 25 15  5  5
14|  5  5 25 15  5  5  5  5  5  5  5  5 15 25  5  5
15| 15 25  5  5  5  5  5  5  5  5  5  5  5  5 25 15
16| 25 15  5  5  5  5  5  5  5  5  5  5  5  5 15 25
```

[構成解の一覧表：[リスト番号]　使用単位面の番号//]

```
[ 1]    3/  1/   [ 2]    3/  2/   [ 3]    3/  5/   [ 4]    3/  6/
 1 23 10 12 19     1 23 10 14 17     2 23  9 11 20     2 23  9 15 16
15 17  4 21  8    15 19  2 21  8    14 16  5 22  8    14 20  1 22  8
24  6 13 20  2    22  6 13 20  4    25  7 13 19  1    21  7 13 19  5
18  5 22  9 11    18  5 24  7 11    18  4 21 10 12    18  4 25  6 12
 7 14 16  3 25     9 12 16  3 25     6 15 17  3 24    10 11 17  3 24
[ 5]    3/ 11/   [ 6]    3/ 12/   [ 7]    3/ 15/   [ 8]    3/ 16/
 4 23  7 11 20     4 23  7 15 16     5 23  6 12 19     5 23  6 14 17
12 16  5 24  8    12 20  1 24  8    11 17  4 25  8    11 19  2 25  8
25  9 13 17  1    21  9 13 17  5    24 10 13 16  2    22 10 13 16  4
18  2 21 10 14    18  2 25  6 14    18  1 22  9 15    18  1 24  7 15
 6 15 19  3 22    10 11 19  3 22     7 14 20  3 21     9 12 20  3 21
[ 9]    4/  1/   [10]    4/  2/   [11]    4/  5/   [12]    4/  6/
 1 23 20 12  9     1 23 20 14  7     2 23 19 11 10     2 23 19 15  6
15  7  4 21 18    15  9  2 21 18    14  6  5 22 18    14 10  1 22 18
24 16 13 10  2    22 16 13 10  4    25 17 13  9  1    21 17 13  9  5
 8  5 22 19 11     8  5 24 17 11     8  4 21 20 12     8  4 25 16 12
17 14  6  3 25    19 12  6  3 25    16 15  7  3 24    20 11  7  3 24
[13]    4/ 11/   [14]    4/ 12/   [15]    4/ 15/   [16]    4/ 16/
 4 23 17 11 10     4 23 17 15  6     5 23 16 12  9     5 23 16 14  7
12  6  5 24 18    12 10  1 24 18    11  7  4 25 18    11  9  2 25 18
25 19 13  7  1    21 19 13  7  5    24 20 13  6  2    22 20 13  6  4
 8  2 21 20 14     8  2 25 16 14     8  1 22 19 15     8  1 24 17 15
16 15  9  3 22    20 11  9  3 22    17 14 10  3 21    19 12 10  3 21
[Count = 16]
```

この時最適の数値が，目的方陣の次数（とＮ進法）によって，いつも１つの値に決まる。五方陣では，５だ。七方陣では，７。

これが不思議だ。他の数値ではどうしても駄目で，これは小生にも理由がまだわからない。いつもいろいろ試験した結果を見て，最適値を決めている。

しかし，経験を何度か積むにつれて，これはどうも２面の「相関度」から見て一番ランダムな数値を必要としているらしい，と直観した。

数値が高過ぎても低過ぎても，正負の相関度が高まる。相関度が高過ぎると，結果的にオイラー方陣の定義条件（３）に反してしまう。それをどうも嫌うようなのだ。

（４）最終的にオイラー方陣の定義条件（３）のチェック - テストを行う。

これで，「原始解」が全部出そろう。リスト整形用の不等式のフィルターを通すと，総数が８分の１に絞られて「標準解」のリストが出来る。

考えてみれば，これが一番不思議だ。そのタイプの魔方陣の解の全数を過不足なく作れたのだから。

少数の単位面を適宜組み合わせただけで，どうして考え得る解の全数を作れるのか？

結果から見て，今までのテストで期待を裏切られたことは一度も無い。出なかった解も無いし，余計な解が出て来たことも無い。要するに，**新オイラー法**は，構成法として完璧だ。

考えるに，この少数の単位面は目的方陣の「基本要素」で，たとえて言うなら，元素と化合物のような関係なのではあるまいか。元素表と化学式があれば，その化合物を想像できるように，単位面のリストとその使用番号がわかれば，計算で解を合成できるからである。

もちろん，次数・n進数・位数などの情報も一緒に与えられるものとしての話だが。

そう，わが**新オイラー法**は，構成要素と構成式があれば，いつでも目的方陣の全体を再現できる「構成法」なのだ。

逆もできる。目的方陣の解の全数を元の「構成要素」と「構成式」にいつでも戻すことができる。ただN進法分解をするだけで。

つまり，**新オイラー法**は，目的方陣の解集合と単位面および使用番号のリストとを，N進法分解／合成によって，一対一に対応付ける変換法である。そう主張して良いと思う。

もしかしたら，この特質が一番重要なことなのかも知れない。

２．新オイラー法の応用

「新オイラー法」のこの特質を活かして，何か違った新しいことができないか。

次は，応用技術を考えてみたい。小生としては，珍しいチャレンジである。

（１）単位面のリストと使用番号の数列のリストがあれば，いつでも全体を復元できるのであるから，解のリストを保存する時に，もう全体を記録媒体にセーヴする必要が無くなる。単位面のリストと使用番号の数列のリストだけを記録に残せば良い。これは，メモリーと時間の節約になりそうだ。

解を復元する時は，単位面と使用番号の数列のリストを記録媒体から読み出して，計算で合成すれば良い。計算は，簡単なもので，手早く済む。解を初めて探索した時の，あの膨大な時間と手間とストレスは，もう要らない。そうした面倒は，最初の一度だけで済む。

四次の場合で，簡単な実験をして見よう。

次のリストは，新オイラー法で汎四方陣を構成した時のものだが，解 48 個を全部保存したい時に，例えば，一番最後に示すように，単位面のリストと使用番号のリストを DATA 文でセーヴする。

DATA 文は，キャラクターでデータを書く。コンマ‘,’で区切りながら十進数で書く。

こういう表現を受け入れてくれるソフトは，案外多い。C や PASCAL のコンパイラーなどでは，すぐに配列に取り込んでくれて，別のプログラムで活用することができる。

利用範囲が広い。データの電送にも適している。

```
＊＊　新オイラー法で汎四方陣を構成する　＊＊

[単位面のリスト]
     1/        2/        3/        4/
0 1 1 0   0 1 0 1   0 1 0 1   0 0 1 1
1 0 0 1   1 0 1 0   0 1 0 1   1 1 0 0
0 1 1 0   1 0 1 0   1 0 1 0   0 0 1 1
1 0 0 1   0 1 0 1   1 0 1 0   1 1 0 0
     5/        6/        7/        8/
1 1 0 0   1 0 1 0   1 0 1 0   1 0 0 1
0 0 1 1   1 0 1 0   0 1 0 1   0 1 1 0
1 1 0 0   0 1 0 1   0 1 0 1   1 0 0 1
0 0 1 1   0 1 0 1   1 0 1 0   0 1 1 0
[Count of Latin Units = 8]

[参照表：最適の組み合せをさがす]
```

```
 *|  1  2  3  4  5  6  7  8
--+------------------------
 1| 16  8  8  8  8  8  8  0
 2|  8 16  8  8  8  8  0  8
 3|  8  8 16  8  8  0  8  8
 4|  8  8  8 16  0  8  8  8
 5|  8  8  8  0 16  8  8  8
 6|  8  8  0  8  8 16  8  8
 7|  8  0  8  8  8  8 16  8
 8|  0  8  8  8  8  8  8 16
```

［構成解のリスト（n1＝１）：　単位面の使用番号////　リスト番号#］

```
  1/   2/   3/   4/           1#     1/   2/   4/   3/           2#
0110 0101 0101 0011   1 15 10  8   0110 0101 0011 0101   1 14 11  8
1001 1010 0101 1100  14  4  5 11   1001 1010 1100 0101  15  4  5 10
0110 1010 1010 0011   7  9 16  2   0110 1010 0011 1010   6  9 16  3
1001 0101 1010 1100  12  6  3 13   1001 0101 1100 1010  12  7  2 13
  1/   3/   2/   4/           3#     1/   3/   4/   2/           4#
0110 0101 0101 0011   1 15 10  8   0110 0101 0011 0101   1 14 11  8
1001 0101 1010 1100  12  6  3 13   1001 0101 1100 1010  12  7  2 13
0110 1010 1010 0011   7  9 16  2   0110 1010 0011 1010   6  9 16  3
1001 1010 0101 1100  14  4  5 11   1001 1010 1100 0101  15  4  5 10

  5# 1/4/2/3/    6# 1/4/3/2/    7# 3/1/2/4/    8# 3/1/4/2/
   1 12 13  8     1 12 13  8     1 15  6 12     1 14  7 12
  15  6  3 10    14  7  2 11     8 10  3 13     8 11  2 13
   4  9 16  5     4  9 16  5    11  5 16  2    10  5 16  3
  14  7  2 11    15  6  3 10    14  4  9  7    15  4  9  6

  9# 3/4/1/2/   10# 4/1/2/3/   11# 4/1/3/2/   12# 4/3/1/2/
   1 12  7 14     1  8 13 12     1  8 13 12     1  8 11 14
   8 13  2 11    15 10  3  6    14 11  2  7    12 13  2  7
  10  3 16  5     4  5 16  9     4  5 16  9     6  3 16  9
  15  6  9  4    14 11  2  7    15 10  3  6    15 10  5  4

   . . . . .
   [Count of Compositions = 48]
```

［データ記録の１つの方法：　DATA 文］

```
//
short tlu[9][16]={
  {0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1},
  {0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1},
  {0,1,0,1,0,1,0,1,1,0,1,0,1,0,1,0},
  {0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0},
  {1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1},
  {1,0,1,0,1,0,1,0,0,1,0,1,0,1,0,1},
  {1,0,1,0,0,1,0,1,0,1,0,1,1,0,1,0},
  {1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0},
  {-1}};
//
short tans[49][4]={
  {1,2,3,4}, {1,2,3,5}, {1,2,4,3}, {1,2,4,6},
  {1,2,5,3}, {1,2,5,6}, {1,2,6,4}, {1,2,6,5},
  {1,3,2,4}, {1,3,2,5}, {1,3,4,2}, {1,3,4,7},
  {1,3,5,2}, {1,3,5,7}, {1,3,7,4}, {1,3,7,5},
```

```
   {1,4,2,3}, {1,4,2,6}, {1,4,3,2}, {1,4,3,7},
   {1,4,6,2}, {1,4,6,7}, {1,4,7,3}, {1,4,7,6},
   {3,1,2,4}, {3,1,2,5}, {3,1,4,2}, {3,1,4,7},
   {3,1,5,2}, {3,1,5,7}, {3,1,7,4}, {3,1,7,5},
   {3,4,1,2}, {3,4,1,7}, {3,5,1,2}, {3,5,1,7},
   {4,1,2,3}, {4,1,2,6}, {4,1,3,2}, {4,1,3,7},
   {4,1,6,2}, {4,1,6,7}, {4,1,7,3}, {4,1,7,6},
   {4,3,1,2}, {4,3,1,7}, {4,6,1,2}, {4,6,1,7},
   {-1}};
//
```

単位面のリストにデータの圧縮効果は無いが，解の各リストの方は，感激ものだ。たった４個の使用番号だけで表わせる。何ともスマートだ。メモリーと記録用紙の節約になる。

この DATA 文を読み込んで，解のリストを全数再現するには，例えば，こうする。

プログラムのリストを示そう。

```
//** Read the DATA Strings, Decode them and Re-compose all  **
//** the Solutions of Pan-diagonal Magic Squares of Order 4 **
//** only by New Euler's Method with Binary System **
//** "CES4RdPr.c": Dictated by Kanji Setsuda  **
//** on Nov. 3, 2003; Revised on Mar. 9, 2013 **
//** with MacOSX 10.8.2 and Xcode 4.6 **
//
#include <stdio.h>
//
short int cnt, cnt2, cnt3;
short cntlu, cntans;
short uum[9];
short anm[4][22];
//
short tlu[9][16]={
   {0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1},
   {0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1},
   {0,1,0,1,0,1,0,1,1,0,1,0,1,0,1,0},
   {0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0},
   {1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1},
   {1,0,1,0,1,0,1,0,0,1,0,1,0,1,0,1},
   {1,0,1,0,0,1,0,1,0,1,0,1,1,0,1,0},
   {1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0},
   {-1}};
//
short tans[49][4]={
   {1,2,3,4}, {1,2,3,5}, {1,2,4,3}, {1,2,4,6},
   {1,2,5,3}, {1,2,5,6}, {1,2,6,4}, {1,2,6,5},
   {1,3,2,4}, {1,3,2,5}, {1,3,4,2}, {1,3,4,7},
   {1,3,5,2}, {1,3,5,7}, {1,3,7,4}, {1,3,7,5},
   {1,4,2,3}, {1,4,2,6}, {1,4,3,2}, {1,4,3,7},
   {1,4,6,2}, {1,4,6,7}, {1,4,7,3}, {1,4,7,6},
   {3,1,2,4}, {3,1,2,5}, {3,1,4,2}, {3,1,4,7},
   {3,1,5,2}, {3,1,5,7}, {3,1,7,4}, {3,1,7,5},
   {3,4,1,2}, {3,4,1,7}, {3,5,1,2}, {3,5,1,7},
   {4,1,2,3}, {4,1,2,6}, {4,1,3,2}, {4,1,3,7},
   {4,1,6,2}, {4,1,6,7}, {4,1,7,3}, {4,1,7,6},
   {4,3,1,2}, {4,3,1,7}, {4,6,1,2}, {4,6,1,7},
   {-1}};
//
void readdata(void);
void prlunit(void), prans(void);
```

```
void pr4ans(void), pr3ans(void), pr2ans(void), pr1ans(void);
void printdata(void);
//
//* Main Program: Read, Re-compose and Print *
int main(){
   printf("\n");
   printf(" ** Read the DATA Strings, Decode them and Re-compose all  **\n");
   printf(" ** the Solutions of Pan-diagonal Magic Squares of Order 4 **\n");
   printf(" **  only by the New Euler's Method with Binary System  **\n");
   readdata();
   printf("\n");
   printf(" [Latin Units]\n");
   prlunit();
   printf(" [Count of Latin Units = %d]\n",cntlu);
   printf("\n");
   cnt=0; cnt2=0; cnt3=0;
   printf(" [Re-compositions(n1==1): Used Units//// List Number#]\n");
   prans();
   if(cnt3==2){pr2ans();}
   if(cnt3==1){pr1ans();}
   printf(" [Count of Compositions = %2d]\n",cntans);
   printf("\n");
   printf(" [Data Strings]\n");
   printdata();
   printf(" OK!\n");
   printf("** Calculated and Listed by Kanji Setsuda\n");
   printf("   on Mar. 9, 2013 with MacOSX 10.8.2 and Xcode 4.6 **\n");
 return 0;
}
//
//* Read the DAIA Strings *
void readdata(){
 short t,n;
   t=0; while(tlu[t][0]>=0){t++;}
   cntlu=t;
 //
   t=0; while(tans[t][0]>=0){t++;}
   cntans=t;
   for(t=0;t<cntans;t++){
     for(n=0;n<4;n++){tans[t][n]--;}
   }
}
//
//* Print the Latin Units *
void prlunit(){
 short t,l,l4,m,n;
   for(t=0;t<cntlu;t=t+4){
     printf("%9d/%9d/%9d/%9d/\n",t+1,t+2,t+3,t+4);
     for(l=0;l<4;l++){l4=l*4;
       for(m=t;m<(t+4);m++){
         printf("  ");
         for(n=0;n<4;n++){printf("%2d",tlu[m][l4+n]);}
       }
       printf("\n");
     }
   }
}
/**/
/* Print The Answer */
void prans(){
```

```
 short t,n;
 short tu[4], nm[22];
   for(n=0;n<cntlu;n++){uum[n]=0;}
   for(t=0;t<cntans;t++){cnt++;
     for(n=0;n<4;n++){tu[n]=tans[t][n];}
     for(n=0;n<16;n++){
       nm[n+1]=tlu[tu[0]][n]*8+tlu[tu[1]][n]*4+tlu[tu[2]][n]*2+tlu[tu[3]][n]+1;
     }
     if(nm[1]==1){
       cnt2++;
       anm[cnt3][0]=cnt;
       for(n=1;n<17;n++){anm[cnt3][n]=nm[n];}
       for(n=0;n<4;n++){
         anm[cnt3][17+n]=tu[n]+1;
         uum[tu[n]]++;}
       anm[cnt3][21]=cnt2;
       cnt3++;
       if((cnt2<=4)&&(cnt3==2)){pr2ans(); cnt3=0;}
       if((cnt2>4)&&(cnt3==4)){pr4ans(); cnt3=0;}
     }
   }
}
//
//* Print 4 Answers *
void pr4ans(){
 short s,l,l4,n;
   for(s=0;s<4;s++){
     printf("%5d#%2d/%d/%d/%d/",
            anm[s][21],anm[s][17],anm[s][18],anm[s][19],anm[s][20]);}
   printf("\n");
   for(l=0;l<4;l++){l4=l*4;
     for(s=0;s<4;s++){
       printf("    ");
       for(n=1;n<5;n++){printf("%3d",anm[s][l4+n]);}
     }
     printf("\n");
   }
}
//
//* Print 2 Answers *
void pr2ans(){
 short s,l,l4,m,n;
 short tu[4];
   for(s=0;s<2;s++){
     printf("%5d/%4d/%4d/%4d/%12d# ",
            anm[s][17],anm[s][18],anm[s][19],anm[s][20],anm[s][21]);}
   printf("\n");
   for(l=0;l<4;l++){l4=l*4;
     for(s=0;s<2;s++){
       for(n=0;n<4;n++){tu[n]=anm[s][n+17]-1;}
       printf(" ");
       for(m=0;m<4;m++){
         printf(" ");
         for(n=0;n<4;n++){printf("%d",tlu[tu[m]][l4+n]);}
       }
       printf(" ");
       for(n=1;n<5;n++){printf("%3d",anm[s][l4+n]);}
       printf(" ");
     }
     printf("\n");
```

```
   }
}
//
//* Print The Answer *
void pr1ans(){
 short l,l4,m,n;
 short tu[4];
   printf("%9d/%8d/%8d/%8d/%12d#\n",
          anm[0][17],anm[0][18],anm[0][19],anm[0][20],anm[0][21]);
   for(l=0;l<4;l++){l4=l*4;
     for(n=0;n<4;n++){tu[n]=anm[0][n+17]-1;}
     printf(" ");
     for(m=0;m<4;m++){
       printf(" ");
       for(n=0;n<4;n++){printf("%2d",tlu[tu[m]][l4+n]);}
     }
     printf(" ");
     for(n=1;n<5;n++){printf("%3d",anm[0][l4+n]);}
     printf("\n");
   }
}
//
//* Print Data Strings *
void printdata(){
 short t,n,v;
   printf("short tlu[%d][16]={\n",cntlu+1);
   for(t=0;t<cntlu;t++){
     printf(" {");
     for(n=0;n<16;n++){
       printf("%d",tlu[t][n]);
       if(n<15){printf(",");}
     }
     printf("},\n");
   }
   printf(" {-1}};\n");
   printf("\n");
 //
   printf("short tans[%d][4]={\n",cntans+1);
   for(t=0;t<cntans;t++){
     printf(" {");
     for(n=0;n<4;n++){
       v=tans[t][n]+1;
       printf("%d",v);
       if(n<3){printf(",");}
     }
     printf("},");
     if((t+1)%4==0){printf("\n");}
   }
   printf(" {-1}};\n");
}
//
```

最後に DATA 文の出力プログラムも添えた。ただのプリント - プログラムなので，特に難しいところは無い。ただ，内部処理では「０系列」の整数列を用いているので，外部表現の「１系列」の自然数列との対応を取るために，少しだけ煩わしいことをしている。

（２）賢い人なら，すぐに単位面のリストの方でも，データ圧縮の方法を思い付くだろう。例えば，こんな方法はどうであろうか？

単位面のリストには‘０’と‘１’の数字しか出て来ない。二進数だから当然なのだが，それなら横１行の４数をまとめて二進法表記と見て，十進化してはどうか。
例えば，{0，1，1，0} を ((０＊２＋１)＊２＋１)＊２＋０＝６(十) にしてしまう。
そうすると，１つの単位面を４個の数字で表せる：{6，9，6，9}
あるいは，いっそのこと 16 個のデータをそっくりまとめて，十進化してはどうか。
例えば {0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1} を
((((((((((((((0*2+1)*2+1)*2+0)*2+1)*2+0)*2+0)*2+1)*2+0)*2+1)*2+1)*2+0)*2+1)*2+0)*2+0)*2+1
＝ 26985(十)
デカい数字にはなるが，単位面が１個の数字で表わされ，全部で８個で済む。
圧縮と伸長をどうプログラミングするかは，あえて説明しない。読者自身が考えてみてほしい。

四次では，さほど有り難みを感じなくても，これが八次だと，データの圧縮効果を堪能できる。そして膨大なデータを，他のプログラムで簡単に利用できるようになる。メモリーに気兼ねが要らないのは，ストレス軽減に役立つ。
次のリストは，三連立型の正方連結対称汎八方陣の解を求めた時のものだ。
いま n１＝１の標準解 360 個をセーヴして，他のプログラムで読み出し，分類・解析したいとする。どうすればよいか？
解のリストを下例のように DATA 文で出力すれば良い。単位面の方は二進法で，使用番号の方は六進法でデータを圧縮してある。単位面の個数が少ない時は，こんな工夫ができる。

```
＊＊　新オイラー法で三連立型正方連結対称汎八方陣を作る　＊＊
［単位面］
     1/       2/       3/       4/       5/       6/
01010101 01010101 01011010 01100110 01010101 00111100
10101010 10101010 10100101 10011001 01010101 11000011
01010101 10101010 01011010 01100110 10101010 00111100
10101010 01010101 10100101 10011001 10101010 11000011
10101010 01010101 01011010 01100110 10101010 00111100
01010101 10101010 10100101 10011001 10101010 11000011
10101010 10101010 01011010 01100110 01010101 00111100
01010101 01010101 10100101 10011001 01010101 11000011
[Count of Latin Units = 6]

［構成解のリスト：　使用番号//////　リスト番号#］
     1/       2/       3/       4/       5/       6/                    1#
01010101 01010101 01011010 01100110 01010101 00111100   1 63  6 60 10 56 13 51
10101010 10101010 10100101 10011001 01010101 11000011  62  4 57  7 53 11 50 16
01010101 10101010 01011010 01100110 10101010 00111100  19 45 24 42 28 38 31 33
10101010 01010101 10100101 10011001 10101010 11000011  48 18 43 21 39 25 36 30
10101010 01010101 01011010 01100110 10101010 00111100  35 29 40 26 44 22 47 17
01010101 10101010 10100101 10011001 10101010 11000011  32 34 27 37 23 41 20 46
10101010 10101010 01011010 01100110 01010101 00111100  49 15 54 12 58  8 61  3
01010101 01010101 10100101 10011001 01010101 11000011  14 52  9 55  5 59  2 64
     1/       2/       3/       5/       4/       6/                    2#
01010101 01010101 01011010 01010101 01100110 00111100   1 63  4 62 10 56 11 53
10101010 10101010 10100101 01010101 10011001 11000011  60  6 57  7 51 13 50 16
01010101 10101010 01011010 10101010 01100110 00111100  21 43 24 42 30 36 31 33
10101010 01010101 10100101 10101010 10011001 11000011  48 18 45 19 39 25 38 28
10101010 01010101 01011010 10101010 01100110 00111100  37 27 40 26 46 20 47 17
01010101 10101010 10100101 10101010 10011001 11000011  32 34 29 35 23 41 22 44
10101010 10101010 01011010 01010101 01100110 00111100  49 15 52 14 58  8 59  5
01010101 01010101 10100101 01010101 10011001 11000011  12 54  9 55  3 61  2 64
```

```
     1/       2/       3/       5/       6/       4/                    3#
01010101 01010101 01011010 01010101 00111100 01100110   1 62  4 63 11 56 10 53
10101010 10101010 10100101 01010101 11000011 10011001  60  7 57  6 50 13 51 16
01010101 10101010 01011010 10101010 00111100 01100110  21 42 24 43 31 36 30 33
10101010 01010101 10100101 10101010 11000011 10011001  48 19 45 18 38 25 39 28
10101010 01010101 01011010 10101010 00111100 01100110  37 26 40 27 47 20 46 17
01010101 10101010 10100101 10101010 11000011 10011001  32 35 29 34 22 41 23 44
10101010 10101010 01011010 01010101 00111100 01100110  49 14 52 15 59  8 58  5
01010101 01010101 10100101 01010101 11000011 10011001  12 55  9 54  2 61  3 64

 No.4#  1/ 2/ 4/ 3/ 5/ 6/  No.5#  1/ 2/ 4/ 5/ 3/ 6/  No.6#  1/ 2/ 4/ 5/ 6/ 3/
   1 63 10 56  6 60 13 51    1 63 10 56  4 62 11 53    1 62 11 56  4 63 10 53
  62  4 53 11 57  7 50 16   60  6 51 13 57  7 50 16   60  7 50 13 57  6 51 16
  19 45 28 38 24 42 31 33   21 43 30 36 24 42 31 33   21 42 31 36 24 43 30 33
  48 18 39 25 43 21 36 30   48 18 39 25 45 19 38 28   48 19 38 25 45 18 39 28
  35 29 44 22 40 26 47 17   37 27 46 20 40 26 47 17   37 26 47 20 40 27 46 17
  32 34 23 41 27 37 20 46   32 34 23 41 29 35 22 44   32 35 22 41 29 34 23 44
  49 15 58  8 54 12 61  3   49 15 58  8 52 14 59  5   49 14 59  8 52 15 58  5
  14 52  5 59  9 55  2 64   12 54  3 61  9 55  2 64   12 55  2 61  9 54  3 64

 No.7#  1/ 2/ 5/ 3/ 4/ 6/  No.8#  1/ 2/ 5/ 3/ 6/ 4/  No.9#  1/ 2/ 5/ 4/ 3/ 6/
   1 63  4 62  6 60  7 57    1 62  4 63  7 60  6 57    1 63  6 60  4 62  7 57
  56 10 53 11 51 13 50 16   56 11 53 10 50 13 51 16   56 10 51 13 53 11 50 16
  25 39 28 38 30 36 31 33   25 38 28 39 31 36 30 33   25 39 30 36 28 38 31 33
  48 18 45 19 43 21 42 24   48 19 45 18 42 21 43 24   48 18 43 21 45 19 42 24
  41 23 44 22 46 20 47 17   41 22 44 23 47 20 46 17   41 23 46 20 44 22 47 17
  32 34 29 35 27 37 26 40   32 35 29 34 26 37 27 40   32 34 27 37 29 35 26 40
  49 15 52 14 54 12 55  9   49 14 52 15 55 12 54  9   49 15 54 12 52 14 55  9
   8 58  5 59  3 61  2 64    8 59  5 58  2 61  3 64    8 58  3 61  5 59  2 64

  10#  1/ 2/ 5/ 4/ 6/ 3/   11#  1/ 2/ 5/ 6/ 3/ 4/   12#  1/ 2/ 5/ 6/ 4/ 3/
   1 62  7 60  4 63  6 57    1 60  6 63  7 62  4 57    1 60  7 62  6 63  4 57
  56 11 50 13 53 10 51 16   56 13 51 10 50 11 53 16   56 13 50 11 51 10 53 16
  25 38 31 36 28 39 30 33   25 36 30 39 31 38 28 33   25 36 31 38 30 39 28 33
  48 19 42 21 45 18 43 24   48 21 43 18 42 19 45 24   48 21 42 19 43 18 45 24
  41 22 47 20 44 23 46 17   41 20 46 23 47 22 44 17   41 20 47 22 46 23 44 17
  32 35 26 37 29 34 27 40   32 37 27 34 26 35 29 40   32 37 26 35 27 34 29 40
  49 14 55 12 52 15 54  9   49 12 54 15 55 14 52  9   49 12 55 14 54 15 52  9
   8 59  2 61  5 58  3 64    8 61  3 58  2 59  5 64    8 61  2 59  3 58  5 64
  . . . . . . .

  355#  5/ 6/ 4/ 1/ 2/ 3/   356#  5/ 6/ 4/ 1/ 3/ 2/   357#  5/ 6/ 4/ 2/ 1/ 3/
   1 48 25 56 18 63 10 39    1 48 25 56 19 62 11 38    1 48 25 56 18 63 10 39
  32 49  8 41 15 34 23 58   32 49  8 41 14 35 22 59   32 49  8 41 15 34 23 58
  35 14 59 22 52 29 44  5   34 15 58 23 52 29 44  5   37 12 61 20 54 27 46  3
  62 19 38 11 45  4 53 28   63 18 39 10 45  4 53 28   60 21 36 13 43  6 51 30
  37 12 61 20 54 27 46  3   37 12 61 20 55 26 47  2   35 14 59 22 52 29 44  5
  60 21 36 13 43  6 51 30   60 21 36 13 42  7 50 31   62 19 38 11 45  4 53 28
   7 42 31 50 24 57 16 33    6 43 30 51 24 57 16 33    7 42 31 50 24 57 16 33
  26 55  2 47  9 40 17 64   27 54  3 46  9 40 17 64   26 55  2 47  9 40 17 64

  358#  5/ 6/ 4/ 2/ 3/ 1/   359#  5/ 6/ 4/ 3/ 1/ 2/   360#  5/ 6/ 4/ 3/ 2/ 1/
   1 48 25 56 19 62 11 38    1 48 25 56 21 60 13 36    1 48 25 56 21 60 13 36
  32 49  8 41 14 35 22 59   32 49  8 41 12 37 20 61   32 49  8 41 12 37 20 61
  37 12 61 20 55 26 47  2   34 15 58 23 54 27 46  3   35 14 59 22 55 26 47  2
  60 21 36 13 42  7 50 31   63 18 39 10 43  6 51 30   62 19 38 11 42  7 50 31
  34 15 58 23 52 29 44  5   35 14 59 22 55 26 47  2   34 15 58 23 54 27 46  3
  63 18 39 10 45  4 53 28   62 19 38 11 42  7 50 31   63 18 39 10 43  6 51 30
   6 43 30 51 24 57 16 33    4 45 28 53 24 57 16 33    4 45 28 53 24 57 16 33
  27 54  3 46  9 40 17 64   29 52  5 44  9 40 17 64   29 52  5 44  9 40 17 64
[Count = 360/360]
```

［DATA 文の出力］

```
short lutbl[7][8]={
 {  85, 170,  85, 170, 170,  85, 170,  85,},
 {  85, 170, 170,  85,  85, 170, 170,  85,},
 {  90, 165,  90, 165,  90, 165,  90, 165,},
 { 102, 153, 102, 153, 102, 153, 102, 153,},
 {  85,  85, 170, 170, 170, 170,  85,  85,},
 {  60, 195,  60, 195,  60, 195,  60, 195,},
 {-208}};

long anstbl[361]={
  1865,  1895,  1905,  2045,  2105,  2120,  2255,  2265,  2285,  2300,
  2355,  2360,  2945,  2975,  2985,  3305,  3395,  3415,  3515,  3525,
  3575,  3595,  3645,  3655,  4205,  4265,  4280,  4385,  4475,  4495,
  4805,  4820,  4835,  4855,  4940,  4945,  5495,  5505,  5525,  5540,
  5595,  5600,  5675,  5685,  5735,  5755,  5805,  5815,  5885,  5900,
  5915,  5935,  6020,  6025,  6315,  6320,  6345,  6355,  6380,  6385,
  8345,  8375,  8385,  8525,  8585,  8600,  8735,  8745,  8765,  8780,
  8835,  8840, 10505, 10535, 10545, 11045, 11165, 11190, 11255, 11265,
 11345, 11370, 11415, 11430, 11765, 11825, 11840, 12125, 12245, 12270,
 12545, 12560, 12605, 12630, 12710, 12720, 13055, 13065, 13085, 13100,
 13155, 13160, 13415, 13425, 13505, 13530, 13575, 13590, 13625, 13640,
 13685, 13710, 13790, 13800, 14055, 14060, 14115, 14130, 14150, 14160,
 15905, 15935, 15945, 16265, 16355, 16375, 16475, 16485, 16535, 16555,
 16605, 16615, 16985, 17015, 17025, 17525, 17645, 17670, 17735, 17745,
 17825, 17850, 17895, 17910, 19505, 19595, 19615, 19685, 19805, 19830,
 20315, 20335, 20345, 20370, 20485, 20490, 20795, 20805, 20855, 20875,
 20925, 20935, 20975, 20985, 21065, 21090, 21135, 21150, 21395, 21415,
 21425, 21450, 21565, 21570, 21825, 21835, 21855, 21870, 21925, 21930,
 23645, 23705, 23720, 23825, 23915, 23935, 24245, 24260, 24275, 24295,
 24380, 24385, 24725, 24785, 24800, 25085, 25205, 25230, 25505, 25520,
 25565, 25590, 25670, 25680, 25985, 26075, 26095, 26165, 26285, 26310,
 26795, 26815, 26825, 26850, 26965, 26970, 28565, 28580, 28595, 28615,
 28700, 28705, 28745, 28760, 28805, 28830, 28910, 28920, 28955, 28975,
 28985, 29010, 29125, 29130, 29600, 29605, 29630, 29640, 29665, 29670,
 31415, 31425, 31445, 31460, 31515, 31520, 31595, 31605, 31655, 31675,
 31725, 31735, 31805, 31820, 31835, 31855, 31940, 31945, 32235, 32240,
 32265, 32275, 32300, 32305, 32495, 32505, 32525, 32540, 32595, 32600,
 32855, 32865, 32945, 32970, 33015, 33030, 33065, 33080, 33125, 33150,
 33230, 33240, 33495, 33500, 33555, 33570, 33590, 33600, 33755, 33765,
 33815, 33835, 33885, 33895, 33935, 33945, 34025, 34050, 34095, 34110,
 34355, 34375, 34385, 34410, 34525, 34530, 34785, 34795, 34815, 34830,
 34885, 34890, 35045, 35060, 35075, 35095, 35180, 35185, 35225, 35240,
 35285, 35310, 35390, 35400, 35435, 35455, 35465, 35490, 35605, 35610,
 36080, 36085, 36110, 36120, 36145, 36150, 37635, 37640, 37665, 37675,
 37700, 37705, 37815, 37820, 37875, 37890, 37910, 37920, 38025, 38035,
 38055, 38070, 38125, 38130, 38240, 38245, 38270, 38280, 38305, 38310,
 -606};
 OK!
```

読み出す時は，次のようなプログラムを用意する。

```
//** Read DATA Strings, Decode them and Recompose all the Solutions **
//** of 3-T Simultaneous Magic Squares 8x8: Composite, **
//**  Self-Complementary and Pan-Diagonal;  **
//** 'CES8Sml3D.c' built by Kanji Setsuda first on Oct.30, 2003; **
//** Revised on Mar. 8, 2013 with MacOSX 10.8.2 and Tcode 4.6 **
//
```

```
#include <stdio.h>
//
short int cnt, cnt2, cnt3;
short cntlu, cntans;
short anm[4][72];
short uum[16];
short tlu[7][64], ntlu[7][64];
short tans[361][6], ntans[361][6];
//
short lutbl[7][8]={
   { 85,170, 85,170,170, 85,170, 85},
   { 85,170,170, 85, 85,170,170, 85},
   { 90,165, 90,165, 90,165, 90,165},
   {102,153,102,153,102,153,102,153},
   { 85, 85,170,170,170,170, 85, 85},
   { 60,195, 60,195, 60,195, 60,195},
   {-208}};
//
long anstbl[361]={
    1865,  1895,  1905,  2045,  2105,  2120,  2255,  2265,  2285,  2300,
    2355,  2360,  2945,  2975,  2985,  3305,  3395,  3415,  3515,  3525,
    3575,  3595,  3645,  3655,  4205,  4265,  4280,  4385,  4475,  4495,
    4805,  4820,  4835,  4855,  4940,  4945,  5495,  5505,  5525,  5540,
    5595,  5600,  5675,  5685,  5735,  5755,  5805,  5815,  5885,  5900,
    5915,  5935,  6020,  6025,  6315,  6320,  6345,  6355,  6380,  6385,
    8345,  8375,  8385,  8525,  8585,  8600,  8735,  8745,  8765,  8780,
    8835,  8840, 10505, 10535, 10545, 11045, 11165, 11190, 11255, 11265,
   11345, 11370, 11415, 11430, 11765, 11825, 11840, 12125, 12245, 12270,
   12545, 12560, 12605, 12630, 12710, 12720, 13055, 13065, 13085, 13100,
   13155, 13160, 13415, 13425, 13505, 13530, 13575, 13590, 13625, 13640,
   13685, 13710, 13790, 13800, 14055, 14060, 14115, 14130, 14150, 14160,
   15905, 15935, 15945, 16265, 16355, 16375, 16475, 16485, 16535, 16555,
   16605, 16615, 16985, 17015, 17025, 17525, 17645, 17670, 17735, 17745,
   17825, 17850, 17895, 17910, 19505, 19595, 19615, 19685, 19805, 19830,
   20315, 20335, 20345, 20370, 20485, 20490, 20795, 20805, 20855, 20875,
   20925, 20935, 20975, 20985, 21065, 21090, 21135, 21150, 21395, 21415,
   21425, 21450, 21565, 21570, 21825, 21835, 21855, 21870, 21925, 21930,
   23645, 23705, 23720, 23825, 23915, 23935, 24245, 24260, 24275, 24295,
   24380, 24385, 24725, 24785, 24800, 25085, 25205, 25230, 25505, 25520,
   25565, 25590, 25670, 25680, 25985, 26075, 26095, 26165, 26285, 26310,
   26795, 26815, 26825, 26850, 26965, 26970, 28565, 28580, 28595, 28615,
   28700, 28705, 28745, 28760, 28805, 28830, 28910, 28920, 28955, 28975,
   28985, 29010, 29125, 29130, 29600, 29605, 29630, 29640, 29665, 29670,
   31415, 31425, 31445, 31460, 31515, 31520, 31595, 31605, 31655, 31675,
   31725, 31735, 31805, 31820, 31835, 31855, 31940, 31945, 32235, 32240,
   32265, 32275, 32300, 32305, 32495, 32505, 32525, 32540, 32595, 32600,
   32855, 32865, 32945, 32970, 33015, 33030, 33065, 33080, 33125, 33150,
   33230, 33240, 33495, 33500, 33555, 33570, 33590, 33600, 33755, 33765,
   33815, 33835, 33885, 33895, 33935, 33945, 34025, 34050, 34095, 34110,
   34355, 34375, 34385, 34410, 34525, 34530, 34785, 34795, 34815, 34830,
   34885, 34890, 35045, 35060, 35075, 35095, 35180, 35185, 35225, 35240,
   35285, 35310, 35390, 35400, 35435, 35455, 35465, 35490, 35605, 35610,
   36080, 36085, 36110, 36120, 36145, 36150, 37635, 37640, 37665, 37675,
   37700, 37705, 37815, 37820, 37875, 37890, 37910, 37920, 38025, 38035,
   38055, 38070, 38125, 38130, 38240, 38245, 38270, 38280, 38305, 38310,
   -606};
//
//* Sub-Routines *
void rddcod(void), prlunit(void), prans(void);
void pr3ans(void);
```

```
void prdata(void), uumon(void);
//
//* Main Program: Reset All and Go! *
int main(){
   printf("\n");
   printf(" ** Read DATA Strings, Decode them and Recompose all the Solutions **\n");
   printf(" ** of Three-Type Simultaneous Magic Squares 8x8: Composite, **\n");
   printf(" **  Self-Complementary and Pan-Diagonal; by New Euler's Method **\n");
   cnt=0;
   rddcod();
   printf("\n");
   printf(" [Latin Units]\n");
   prlunit();
   cnt=0; cnt2=0; cnt3=0;
   printf("\n");
   printf(" [Re-compositions: List Number# used units//////]\n");
   prans();
   printf(" [Count = %d/%d#]\n",cnt2,cnt);
   cnt=0;
   printf("\n");
   printf(" [Used Unit Monitor]\n");
   uumon();
   printf("\n");
   printf(" [DATA Strings]\n");
   prdata();
   printf(" OK!\n");
   printf("** Calculated and Listed by Kanji Setsuda\n");
   printf("   on Mar. 8, 2013 with MacOSX 10.8.2 and Xcode4.6 **\n");
 return 0;
}
//
//* Read and Decode the Data Strings *
void rddcod(){
 long int v;
 short t,m,n,npos,bas,pos;
 short dnm[8];
   t=0; while(lutbl[t][0]>=0){t++;};
   cntlu=t; n=lutbl[cntlu][0]*(-1);
   bas=n/100; pos=n%100;
   for(t=0;t<cntlu;t++){
     for(n=0;n<pos;n++){npos=n*pos;
       v=lutbl[t][n]; m=pos-1;
       while(m>=0){
         dnm[m]=v%bas; v=v/bas;
         m--;}
       for(m=0;m<pos;m++){tlu[t][npos+m]=dnm[m];}
     }
   }
   tlu[cntlu][0]=-1;
 //
   t=0; while(anstbl[t]>=0){t++;};
   cntans=t; v=anstbl[cntans]*(-1);
   bas=v/100; pos=v%100;
   for(t=0;t<cntans;t++){
     v=anstbl[t]; m=pos-1;
     while(m>=0){
       dnm[m]=v%bas; v=v/bas;
       m--;}
     for(n=0;n<pos;n++){tans[t][n]=dnm[n];}
   }
```

```
   tans[cntans][0]=-1;
}
//
//* Print the Latin Units *
void prlunit(){
 short t,u,m,n,l,l8;
   if(cntlu>6){u=6;}else{u=cntlu;}
   for(t=0;t<cntlu;t=t+u){
     for(n=1;n<=u;n++){printf("%9d/",t+n);}
     printf("\n");
     for(l=0;l<8;l++){l8=l*8;
       for(m=t;m<(t+u);m++){
         printf("  ");
         for(n=0;n<8;n++){printf("%d",tlu[m][l8+n]);}
       }
       printf("\n");
     }
   }
   printf(" [Count of Latin Units = %d]\n",cntlu);
}
//
//* Recompose and Print the Answers *
void prans(){
 short t,m,n;
 short tu1,tu2,tu3,tu4,tu5,tu6;
   for(t=0;t<cntans;t++){cnt++;
     tu1=tans[t][0]; tu2=tans[t][1]; tu3=tans[t][2];
     tu4=tans[t][3]; tu5=tans[t][4]; tu6=tans[t][5];
     for(n=0;n<64;n++){
       m=tlu[tu1][n]*32+tlu[tu2][n]*16+tlu[tu3][n]*8+tlu[tu4][n]*4+tlu[tu5][n]*2+tlu[tu6][n]+1;
       anm[cnt3][n+1]=m;}
     anm[cnt3][65]=tu1+1; anm[cnt3][66]=tu2+1; anm[cnt3][67]=tu3+1;
     anm[cnt3][68]=tu4+1; anm[cnt3][69]=tu5+1; anm[cnt3][70]=tu6+1;
     if((anm[cnt3][1]==1)&&(anm[cnt3][64]==64)){
       uum[tu1]++; uum[tu2]++; uum[tu3]++; uum[tu4]++; uum[tu5]++; uum[tu6]++;
       for(n=0;n<6;n++){ntans[cnt2][n]=tans[t][n];}
       cnt2++; anm[cnt3][0]=cnt2;
       cnt3++;
       if(cnt3==3){pr3ans(); cnt3=0;}
     }
   }
   cntans=cnt2;
}
//
//* Print 3 Answers *
void pr3ans(){
 short l,l8,m,n;
   for(m=0;m<3;m++){
     printf("%4d#%4d/%2d/%2d/%2d/%2d/%2d/ ",
            anm[m][0],anm[m][65],anm[m][66],anm[m][67],anm[m][68],anm[m][69],anm[m][70]);}
   printf("\n");
   for(l=0;l<8;l++){l8=l*8;
     for(m=0;m<3;m++){
       printf("  ");
       for(n=1;n<9;n++){printf("%3d",anm[m][l8+n]);}}
     printf("\n");
   }
}
//
//* Used Unit Monitor *
```

```
void uumon(){
 short m,n,p;
   for(n=0;n<cntlu;n++){
     m=uum[n]; uum[n]=-1;
     if(m>0){
       printf("%2d/%d, ",n+1,m);
       for(p=0;p<64;p++){ntlu[cnt][p]=tlu[n][p];}
       uum[n]=cnt;
       cnt++;
     }
   }
   printf("\n");
   cntlu=cnt;
 //
   for(n=0;n<cntans;n++){
     for(m=0;m<6;m++){
       p=ntans[n][m];
       ntans[n][m]=uum[p];
     }}
}
//
//* Print the Data Strings *
void prdata(){
 long int v;
 short t,m,n,pos,bas;
 short tu1,tu2,tu3,tu4,tu5,tu6;
   pos=8; bas=2;
   printf("short lutbl[%d][8]={\n",cntlu+1);
   for(t=0;t<cntlu;t++){
     printf(" {");
     for(m=0;m<64;m=m+8){v=0;
       for(n=0;n<(pos-1);n++){v=(v+ntlu[t][m+n])*bas;}
       v=v+ntlu[t][m+pos-1];
       printf("%4ld,",v);
     }
     printf("},\n");
   }
   printf(" {%4d}};\n",(-1)*bas*100-pos);
 //
   pos=6; bas=cntlu;
   printf("\n");
   printf("long anstbl[%d]={\n",cntans+1);
   for(t=0;t<cntans;t++){
     tu1=ntans[t][0]; tu2=ntans[t][1]; tu3=ntans[t][2];
     tu4=ntans[t][3]; tu5=ntans[t][4]; tu6=ntans[t][5];
     v=((((tu1*bas+tu2)*bas+tu3)*bas+tu4)*bas+tu5)*bas+tu6;
     printf("%6ld,",v);
     if((t+1)%10==0){printf("\n");}
   }
   printf("%5d};\n",(-1)*bas*100-pos);
}
//
```

８×８の八方陣の解 360 個の全データが，１ページに収まってしまうのだから，きっと快感を覚える。

これでも，全体のリストを完璧に再現する。目的方陣の構造解析などに役立ててほしい。

最後に DATA 文の出力プログラムも添えておいた。

こうした技術は，昨今流行りの「暗号」の分野にも応用できそうだが，小生はその方の専

門家ではないので，ここでは触れない。
　それに魔方陣の知識をこれ以上秘儀化して怪しいものにしてしまうのは，小生の本意ではない。「付加価値を高める」てぇ功利的な発想も，小生の趣味ではない。

（3）「新オイラー法」のもう１つの応用可能性は，魔方陣の**構造解析**にある。
　新オイラー法によって，オイラー方陣の解が単位面とその使用番号に分解・還元されることがわかった。これからは，構想解析も，単位面そのものとその利用法について分析を行なう必要が生じる。
　例えば，汎四方陣では，次のような考察が可能である。次のリストをベースに論じよう。

＊＊　新オイラー法で汎四方陣の解 48 個を作る　＊＊

［単位面８個］

```
      1/          2/          3/          4/
0 0 1 1   0 1 0 1   0 1 0 1   0 1 1 0
1 1 0 0   0 1 0 1   1 0 1 0   1 0 0 1
0 0 1 1   1 0 1 0   1 0 1 0   0 1 1 0
1 1 0 0   1 0 1 0   0 1 0 1   1 0 0 1
      5/          6/          7/          8/
1 0 0 1   1 0 1 0   1 0 1 0   1 1 0 0
0 1 1 0   0 1 0 1   1 0 1 0   0 0 1 1
1 0 0 1   0 1 0 1   0 1 0 1   1 1 0 0
0 1 1 0   1 0 1 0   0 1 0 1   0 0 1 1
```

［構成解 48 個： リスト番号＃　使用単位面の番号////］

```
 1# 1/2/4/3/     2# 1/4/2/3/     3# 1/4/3/2/     4# 2/1/4/3/
  1  8 11 14       1  8 13 12       1  8 13 12      1 12  7 14
 12 13  2  7      14 11  2  7      15 10  3  6      8 13  2 11
  6  3 16  9       4  5 16  9       4  5 16  9     10  3 16  5
 15 10  5  4      15 10  3  6      14 11  2  7     15  6  9  4
 5# 2/4/1/3/     6# 2/4/3/1/     7# 4/1/2/3/     8# 4/1/3/2/
  1 14  7 12       1 15  6 12       1 12 13  8      1 12 13  8
  8 11  2 13       8 10  3 13      14  7  2 11     15  6  3 10
 10  5 16  3      11  5 16  2       4  9 16  5      4  9 16  5
 15  4  9  6      14  4  9  7      15  6  3 10     14  7  2 11
 9# 4/2/1/3/    10# 4/2/3/1/    11# 4/3/1/2/    12# 4/3/2/1/
  1 14 11  8       1 15 10  8       1 14 11  8      1 15 10  8
 12  7  2 13      12  6  3 13      15  4  5 10     14  4  5 11
  6  9 16  3       7  9 16  2       6  9 16  3      7  9 16  2
 15  4  5 10      14  4  5 11      12  7  2 13     12  6  3 13
13# 1/2/4/6/    14# 1/4/2/6/    15# 1/4/3/7/    16# 2/1/4/6/
  2  7 12 13       2  7 14 11       2  7 14 11      2 11  8 13
 11 14  1  8      13 12  1  8      16  9  4  5      7 14  1 12
  5  4 15 10       3  6 15 10       3  6 15 10      9  4 15  6
 16  9  6  3      16  9  4  5      13 12  1  8     16  5 10  3
17# 2/4/1/6/    18# 2/4/3/8/    19# 4/1/2/6/    20# 4/1/3/7/
  2 13  8 11       2 16  5 11       2 11 14  7      2 11 14  7
  7 12  1 14       7  9  4 14      13  8  1 12     16  5  4  9
  9  6 15  4      12  6 15  1       3 10 15  6      3 10 15  6
 16  3 10  5      13  3 10  8      16  5  4  9     13  8  1 12
```

```
 21# 4/2/1/6/    22# 4/2/3/8/    23# 4/3/1/7/    24# 4/3/2/8/
  2 13 12  7      2 16  9  7      2 13 12  7      2 16  9  7
 11  8  1 14     11  5  4 14     16  3  6  9     13  3  6 12
  5 10 15  4      8 10 15  1      5 10 15  4      8 10 15  1
 16  3  6  9     13  3  6 12     11  8  1 14     11  5  4 14
 25# 1/4/6/2/    26# 1/4/7/3/    27# 2/4/6/1/    28# 2/4/8/3/
  3  6 15 10      3  6 15 10      3 13  8 10      3 16  5 10
 13 12  1  8     16  9  4  5      6 12  1 15      6  9  4 15
  2  7 14 11      2  7 14 11      9  7 14  4     12  7 14  1
 16  9  4  5     13 12  1  8     16  2 11  5     13  2 11  8
 29# 4/1/6/2/    30# 4/1/7/3/    31# 4/2/6/1/    32# 4/2/8/3/
  3 10 15  6      3 10 15  6      3 13 12  6      3 16  9  6
 13  8  1 12     16  5  4  9     10  8  1 15     10  5  4 15
  2 11 14  7      2 11 14  7      5 11 14  4      8 11 14  1
 16  5  4  9     13  8  1 12     16  2  7  9     13  2  7 12
 33# 4/3/7/1/    34# 4/3/8/2/    35# 1/4/6/7/    36# 1/4/7/6/
  3 13 12  6      3 16  9  6      4  5 16  9      4  5 16  9
 16  2  7  9     13  2  7 12     14 11  2  7     15 10  3  6
  5 11 14  4      8 11 14  1      1  8 13 12      1  8 13 12
 10  8  1 15     10  5  4 15     15 10  3  6     14 11  2  7
 37# 2/4/6/8/    38# 2/4/8/6/    39# 4/1/6/7/    40# 4/1/7/6/
  4 14  7  9      4 15  6  9      4  9 16  5      4  9 16  5
  5 11  2 16      5 10  3 16     14  7  2 11     15  6  3 10
 10  8 13  3     11  8 13  2      1 12 13  8      1 12 13  8
 15  1 12  6     14  1 12  7     15  6  3 10     14  7  2 11
 41# 4/2/6/8/    42# 4/2/8/6/    43# 4/3/7/8/    44# 4/3/8/7/
  4 14 11  5      4 15 10  5      4 14 11  5      4 15 10  5
  9  7  2 16      9  6  3 16     15  1  8 10     14  1  8 11
  6 12 13  3      7 12 13  2      6 12 13  3      7 12 13  2
 15  1  8 10     14  1  8 11      9  7  2 16      9  6  3 16
 45# 1/7/4/3/    46# 2/8/4/3/    47# 1/7/4/6/    48# 2/8/4/6/
  5  4 15 10      5 16  3 10      6  3 16  9      6 15  4  9
 16  9  6  3      4  9  6 15     15 10  5  4      3 10  5 16
  2  7 12 13     14  7 12  1      1  8 11 14     13  8 11  2
 11 14  1  8     11  2 13  8     12 13  2  7     12  1 14  7
[Count of Compositions = 48/48]
```

このリストは，不等式 n１<n16; n１<n４; n４<n13; によって整理してある。今までとちょっと違うが，この方が単位面相互の関連を調べるのに「見やすい」という特徴がある。

上の表を見て，使用番号の変化パターンをさがしてほしい。何かお気付きだろうか？

[1] n１＝１の標準解 12 個を見ると，使用番号が｛1/, 2/, 3/, 4/｝だけに収まっている。他の単位面は使われていない。これは印象的だが，考えてみれば，当然だ。

n１＝１は，古典的表記での話。内部表現の「０系列」で表わすと，n１＝０になる。分解図で考えると，n１＝A1×８＋B1×４＋C1×２＋D1×１＝０

（A1, B1, C1, D1＝｛0, 1｝）ならば，解は A1＝B1＝C1＝D1＝０ しかない。そういう単位面は，８面中｛1/, 2/, 3/, 4/｝だけだ。

[2] n１＝１の標準解 12 個と n１＝２の解 12 個とを順に比べて行くと，面白いことに気付く。使用単位面の最初の３面が同じで，最後の１面だけが変わっている。これは，規則的な対応と言って良いだろうか？ 3/は 6/に，2/は 7/に，そして 1/は 8/に変わっている。

これらのペアは，「補数面」の関係だ。対応するポジションの数値を足すと，全てが１になる。

An＋Bn＝１（n=1，2，3，4，... ，15，16）

{1/，8/}，{2/，7/}，{3/，6/}，{4/，5/} はどれも，同時には使えない「補数面」だ。その相手面に変わるというのは，面白い関係だ。規則的対応と言って良い。

・・・

いろいろ規則的な対応関係を発見すると，それを「見やすい形」に整理したくなる。入力 DATA 文の数字をあちこち入れ替えて，小生は最終的に次のようなリストを作った。

前の研究成果を活かして，先ずは 48 個の解を 16 個ずつ３つのグループに分けた。行列の内容が同じものを集めたのである。

そして，そのグループ内を整理して見やすい順序にした。

次にグループ相互の対応関係を同じパターンにそろえた。

「美しい対応関係」が，あちこちに見つかった。

[Group 1]

1# 1/2/4/3/	2# 2/4/3/1/	3# 4/2/1/3/	4# 4/3/1/2/
1 8 11 14	1 15 6 12	1 14 11 8	1 14 11 8
12 13 2 7	8 10 3 13	12 7 2 13	15 4 5 10
6 3 16 9	11 5 16 2	6 9 16 3	6 9 16 3
15 10 5 4	14 4 9 7	15 4 5 10	12 7 2 13
5# 1/2/4/6/	**6# 2/4/3/8/**	**7# 4/2/1/6/**	**8# 4/3/1/7/**
2 7 12 13	2 16 5 11	2 13 12 7	2 13 12 7
11 14 1 8	7 9 4 14	11 8 1 14	16 3 6 9
5 4 15 10	12 6 15 1	5 10 15 4	5 10 15 4
16 9 6 3	13 3 10 8	16 3 6 9	11 8 1 14
9# 1/7/4/3/	**10# 2/4/6/1/**	**11# 4/2/8/3/**	**12# 4/3/8/2/**
5 4 15 10	3 13 8 10	3 16 9 6	3 16 9 6
16 9 6 3	6 12 1 15	10 5 4 15	13 2 7 12
2 7 12 13	9 7 14 4	8 11 14 1	8 11 14 1
11 14 1 8	16 2 11 5	13 2 7 12	10 5 4 15
13# 1/7/4/6/	**14# 2/4/6/8/**	**15# 4/2/8/6/**	**16# 4/3/8/7/**
6 3 16 9	4 14 7 9	4 15 10 5	4 15 10 5
15 10 5 4	5 11 2 16	9 6 3 16	14 1 8 11
1 8 11 14	10 8 13 3	7 12 13 2	7 12 13 2
12 13 2 7	15 1 12 6	14 1 8 11	9 6 3 16

[Group 2]

1# 2/1/4/3/	2# 2/4/1/3/	3# 4/2/3/1/	4# 4/3/2/1/
1 12 7 14	1 14 7 12	1 15 10 8	1 15 10 8
8 13 2 11	8 11 2 13	12 6 3 13	14 4 5 11
10 3 16 5	10 5 16 3	7 9 16 2	7 9 16 2
15 6 9 4	15 4 9 6	14 4 5 11	12 6 3 13
5# 2/1/4/6/	**6# 2/4/1/6/**	**7# 4/2/3/8/**	**8# 4/3/2/8/**
2 11 8 13	2 13 8 11	2 16 9 7	2 16 9 7
7 14 1 12	7 12 1 14	11 5 4 14	13 3 6 12
9 4 15 6	9 6 15 4	8 10 15 1	8 10 15 1
16 5 10 3	16 3 10 5	13 3 6 12	11 5 4 14
9# 2/8/4/3/	**10# 2/4/8/3/**	**11# 4/2/6/1/**	**12# 4/3/7/1/**
5 16 3 10	3 16 5 10	3 13 12 6	3 13 12 6
4 9 6 15	6 9 4 15	10 8 1 15	16 2 7 9
14 7 12 1	12 7 14 1	5 11 14 4	5 11 14 4
11 2 13 8	13 2 11 8	16 2 7 9	10 8 1 15

```
 13# 2/8/4/6/    14# 2/4/8/6/    15# 4/2/6/8/    16# 4/3/7/8/
  6 15  4  9      4 15  6  9      4 14 11  5      4 14 11  5
  3 10  5 16      5 10  3 16      9  7  2 16     15  1  8 10
 13  8 11  2     11  8 13  2      6 12 13  3      6 12 13  3
 12  1 14  7     14  1 12  7     15  1  8 10      9  7  2 16
```

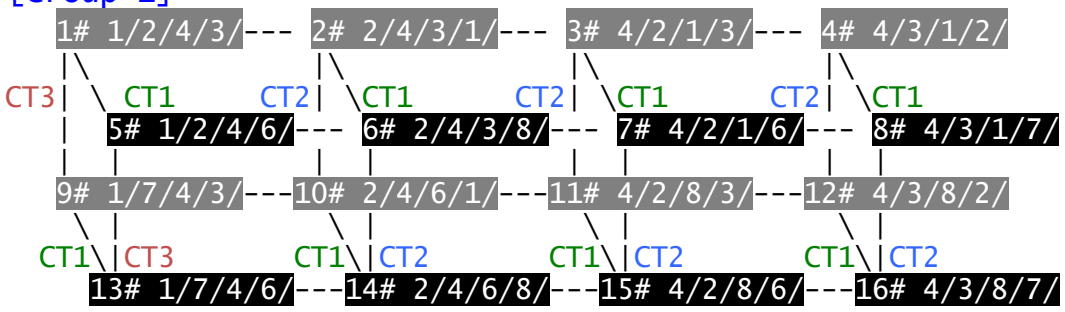
[Group 3]

```
  1# 1/4/2/3/     2# 1/4/3/2/     3# 4/1/2/3/     4# 4/1/3/2/
   1  8 13 12       1  8 13 12       1 12 13  8      1 12 13  8
  14 11  2  7      15 10  3  6      14  7  2 11     15  6  3 10
   4  5 16  9       4  5 16  9       4  9 16  5      4  9 16  5
  15 10  3  6      14 11  2  7      15  6  3 10     14  7  2 11
  5# 1/4/2/6/     6# 1/4/3/7/     7# 4/1/2/6/     8# 4/1/3/7/
   2  7 14 11       2  7 14 11       2 11 14  7      2 11 14  7
  13 12  1  8      16  9  4  5      13  8  1 12     16  5  4  9
   3  6 15 10       3  6 15 10       3 10 15  6      3 10 15  6
  16  9  4  5      13 12  1  8      16  5  4  9     13  8  1 12
  9# 1/4/7/3/     10# 1/4/6/2/     11# 4/1/7/3/    12# 4/1/6/2/
   3  6 15 10       3  6 15 10       3 10 15  6      3 10 15  6
  16  9  4  5      13 12  1  8      16  5  4  9     13  8  1 12
   2  7 14 11       2  7 14 11       2 11 14  7      2 11 14  7
  13 12  1  8      16  9  4  5      13  8  1 12     16  5  4  9
 13# 1/4/7/6/     14# 1/4/6/7/     15# 4/1/7/6/    16# 4/1/6/7/
   4  5 16  9       4  5 16  9       4  9 16  5      4  9 16  5
  15 10  3  6      14 11  2  7      15  6  3 10     14  7  2 11
   1  8 13 12       1  8 13 12       1 12 13  8      1 12 13  8
  14 11  2  7      15 10  3  6      14  7  2 11     15  6  3 10
```

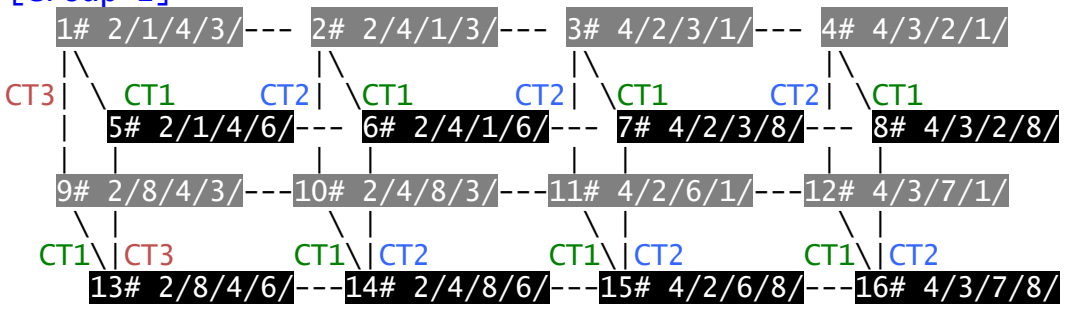
対応関係が，三次元に渡るので，下図を描いた。構造らしきものが見えて来た。

下図で‘CT1’とあるのは，最下位面どうしの「補数面変換」を表わす。‘CT2’とあるのは，下から２番目の面の「補数面変換」だ。‘CT3’は，下から３番目の「補数面変換」を表わす。

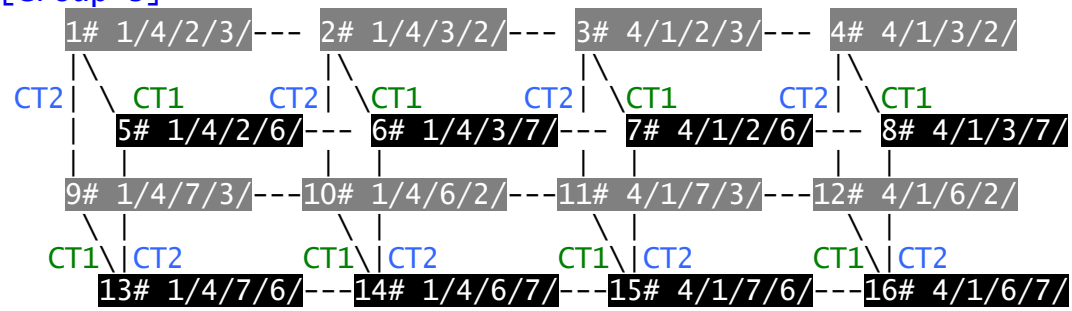
[Group 1]



[Group 2]



[Group 3]



だが，これで終わりではない。n１＝１の標準解 12 個内部の「規則的対応関係」がまだ見つかっていない。

これが，けっこう難問なのである。単純な２面交換で説明ができるペアもあるが，回転シフトで説明しなければならない別の１組もある。できるだけ少ない共通のパターンを見つけたいのだが，求める「美しい規則」がなかなかに見つからない。

しばらくの間必死に取り組んで，悩んだ末に，ここは発想を変えることにした。基本形から変換する方法をさがすのは止めよう。

12 解の成り立ちそのものをもう１度見直すことにしよう。

{1/，2/，3/，4/} の４面から４面を取る順列は，$_4P_4 = 4 \times 3 \times 2 \times 1 = 24$

24 個の原始解から 12 個の最終解を選んだ理由は何だったか？

［原始解 24 個： 使用単位面の番号//// リスト番号#］

```
  1/   2/   3/   4/            1#     1/   2/   4/   3/            2#
0011 0101 0101 0110   1  8 10 15   0011 0101 0110 0101   1  8 11 14
1100 0101 1010 1001  12 13  3  6   1100 0101 1001 1010  12 13  2  7
0011 1010 1010 0110   7  2 16  9   0011 1010 0110 1010   6  3 16  9
1100 1010 0101 1001  14 11  5  4   1100 1010 1001 0101  15 10  5  4
  1/   3/   2/   4/            3#     1/   3/   4/   2/            4#
0011 0101 0101 0110   1  8 10 15   0011 0101 0110 0101   1  8 11 14
1100 1010 0101 1001  14 11  5  4   1100 1010 1001 0101  15 10  5  4
0011 1010 1010 0110   7  2 16  9   0011 1010 0110 1010   6  3 16  9
1100 0101 1010 1001  12 13  3  6   1100 0101 1001 1010  12 13  2  7
  1/   4/   2/   3/            5#     1/   4/   3/   2/            6#
0011 0110 0101 0101   1  8 13 12   0011 0110 0101 0101   1  8 13 12
1100 1001 0101 1010  14 11  2  7   1100 1001 1010 0101  15 10  3  6
0011 0110 1010 1010   4  5 16  9   0011 0110 1010 1010   4  5 16  9
1100 1001 1010 0101  15 10  3  6   1100 1001 0101 1010  14 11  2  7
  2/   1/   3/   4/            7#     2/   1/   4/   3/            8#
0101 0011 0101 0110   1 12  6 15   0101 0011 0110 0101   1 12  7 14
0101 1100 1010 1001   8 13  3 10   0101 1100 1001 1010   8 13  2 11
1010 0011 1010 0110  11  2 16  5   1010 0011 0110 1010  10  3 16  5
1010 1100 0101 1001  14  7  9  4   1010 1100 1001 0101  15  6  9  4
  2/   3/   1/   4/            9#     2/   3/   4/   1/           10#
0101 0101 0011 0110   1 14  4 15   0101 0101 0110 0011   1 15  4 14
0101 1010 1100 1001   8 11  5 10   0101 1010 1001 1100   8 10  5 11
1010 1010 0011 0110  13  2 16  3   1010 1010 0110 0011  13  3 16  2
1010 0101 1100 1001  12  7  9  6   1010 0101 1001 1100  12  6  9  7
  2/   4/   1/   3/           11#     2/   4/   3/   1/           12#
0101 0110 0011 0101   1 14  7 12   0101 0110 0101 0011   1 15  6 12
0101 1001 1100 1010   8 11  2 13   0101 1001 1010 1100   8 10  3 13
1010 0110 0011 1010  10  5 16  3   1010 0110 1010 0011  11  5 16  2
1010 1001 1100 0101  15  4  9  6   1010 1001 0101 1100  14  4  9  7
```

```
  3/   1/   2/   4/          13#     3/   1/   4/   2/          14#
0101 0011 0101 0110   1 12  6 15   0101 0011 0110 0101   1 12  7 14
1010 1100 0101 1001  14  7  9  4   1010 1100 1001 0101  15  6  9  4
1010 0011 1010 0110  11  2 16  5   1010 0011 0110 1010  10  3 16  5
0101 1100 1010 1001   8 13  3 10   0101 1100 1001 1010   8 13  2 11
  3/   2/   1/   4/          15#     3/   2/   4/   1/          16#
0101 0101 0011 0110   1 14  4 15   0101 0101 0110 0011   1 15  4 14
1010 0101 1100 1001  12  7  9  6   1010 0101 1001 1100  12  6  9  7
1010 1010 0011 0110  13  2 16  3   1010 1010 0110 0011  13  3 16  2
0101 1010 1100 1001   8 11  5 10   0101 1010 1001 1100   8 10  5 11
  3/   4/   1/   2/          17#     3/   4/   2/   1/          18#
0101 0110 0011 0101   1 14  7 12   0101 0110 0101 0011   1 15  6 12
1010 1001 1100 0101  15  4  9  6   1010 1001 0101 1100  14  4  9  7
1010 0110 0011 1010  10  5 16  3   1010 0110 1010 0011  11  5 16  2
0101 1001 1100 1010   8 11  2 13   0101 1001 1010 1100   8 10  3 13
  4/   1/   2/   3/          19#     4/   1/   3/   2/          20#
0110 0011 0101 0101   1 12 13  8   0110 0011 0101 0101   1 12 13  8
1001 1100 0101 1010  14  7  2 11   1001 1100 1010 0101  15  6  3 10
0110 0011 1010 1010   4  9 16  5   0110 0011 1010 1010   4  9 16  5
1001 1100 1010 0101  15  6  3 10   1001 1100 0101 1010  14  7  2 11
  4/   2/   1/   3/          21#     4/   2/   3/   1/          22#
0110 0101 0011 0101   1 14 11  8   0110 0101 0101 0011   1 15 10  8
1001 0101 1100 1010  12  7  2 13   1001 0101 1010 1100  12  6  3 13
0110 1010 0011 1010   6  9 16  3   0110 1010 1010 0011   7  9 16  2
1001 1010 1100 0101  15  4  5 10   1001 1010 0101 1100  14  4  5 11
  4/   3/   1/   2/          23#     4/   3/   2/   1/          24#
0110 0101 0011 0101   1 14 11  8   0110 0101 0101 0011   1 15 10  8
1001 1010 1100 0101  15  4  5 10   1001 1010 0101 1100  14  4  5 11
0110 1010 0011 1010   6  9 16  3   0110 1010 1010 0011   7  9 16  2
1001 0101 1100 1010  12  7  2 13   1001 0101 1010 1100  12  6  3 13
```

選別基準は，リスト整形用の不等式： n４＜n13 であった。解の右上角の数と左下角の数を比較して n４＞n13 の解を外したのであった。単位面を見ると，n４＜n13 の大小判断に影響を与えるのは，｛3/，4/｝の２面のみ。使用番号リストで言うと，3/面が 4/面より上位にあると，必ず n４＞n13 になる。どこにあっても，4/面は 3/面より上位になければならない。

だから，3/面が最上位に来ることは，絶対にない。4/面が最下位に来ることもない。

ならば，「その基準で ₄P₄ の 24 解をふるいにかけ，12 個の解を残す。そして，行列の内容によって，例の３つのグループに４個ずつ仕分けする」。求める規則は，それで良いのではないか。最初の順列は，単に使用番号のリストを作るだけで良い。機械的にやって，

```
{1/,2/,3/,4/}, {1/,2/,4/,3/}, {1/,3/,2/,4/}, {1/,3/,4/,2/},
{1/,4/,2/,3/}, {1/,4/,3/,2/}, {2/,1/,3/,4/}, {2/,1/,4/,3/},
. . . . .
{4/,2/,1/,3/}, {4/,2/,3/,1/}, {4/,3/,1/,2/}, {4/,3/,2/,1/}
```

そして，4/面が 3/面より上位にあるものだけを残して，12 解を作るのである。

美しいとは必ずしも言えない説明ではあるが，成り立ちがそうなっているからには，ストレートにそれを認めるしかないのではあるまいか。

単位面内部にまで踏み入らなくても，番号だけでもけっこう面白い性質が見つかる。新しい「構造研究」の方法の可能性をアピールしたい，と思う。

やはり，オイラー方陣の二進法分解図てぇものは，「ただもの」ではなかった。

わが**新オイラー法**が，魔方陣研究の新たな地平を開くことを切に願ってやまない。

３．「ノン - オイラー方陣」について

オイラー方陣ばかりを多数見て来ると，魔方陣はこれだけのように思えてしまう。

が，七次以上の魔方陣では，オイラー方陣はあくまで少数派だ。「ノン - オイラー方陣」の方が圧倒的に多い。

下の例を見てほしい。完全八方陣（65 の補数ペアが全て汎対角線上にある汎八方陣）の解に二進法分解図を添えたものだが，分解図の各面を見てほしい。

どの行も列も，０と１が好き勝手に並んでいる。

比較的ましなのは汎対角線で，だいたいが｛0，0，0，0，1，1，1，1｝の顔ぶれになっている。

＊＊　完全八方陣の例と二進法分解図　＊＊

```
                        1#  /D2i 32*      16*       8*       4*       2*       1*
 1 62  5 59  2 61 12 58  01010101 01010101 01010111 01100100 00010010 01001011
57 14 50 48  9 18 45 19  10110010 10100101 11011010 01010010 00010001 01110100
10 27 34 25 54 33 36 41  00101111 01011000 11010001 00001000 01000010 10101010
49 30 26 23 52 21 22 37  10001001 11111110 01100000 01010111 00011000 01101010
63  4 53  7 64  3 60  6  10101010 10101010 10001010 10111001 11011110 01001011
56 47 20 46  8 51 15 17  11010100 10100101 01010010 11011010 11101110 10111000
11 32 29 24 55 38 31 40  00001101 01111010 11100010 01111111 11011011 01010101
13 44 43 28 16 35 39 42  01100111 00010000 11111001 10001010 01111110 01011001
                        9#  /D2i 32*      16*       8*       4*       2*       1*
 1 62  5 59  2 61 12 58  01010101 01010101 01010111 01100100 00010010 01001011
57 14 50 48  9 18 45 19  10110010 10100101 11011010 01010010 00010001 01110100
10 28 35 25 54 33 36 39  00101111 01011000 11010000 00001001 01100011 11001010
49 31 24 22 52 23 21 38  10001001 11111110 01000000 01110111 01101100 00111001
63  4 53  7 64  3 60  6  10101010 10101010 10001010 10111001 11011110 01001011
56 47 20 46  8 51 15 17  11010100 10100101 01010010 11011010 11101110 10111000
11 32 29 26 55 37 30 40  00001101 01111010 11110010 01101111 11001001 01010011
13 42 44 27 16 34 41 43  01100111 00010000 11111011 10001000 00111001 01101100
                       17#  /D2i 32*      16*       8*       4*       2*       1*
 1 62  5 59  2 61 12 58  01010101 01010101 01010111 01100100 00010010 01001011
57 14 50 48  9 18 45 19  10110010 10100101 11011010 01010010 00010001 01110100
10 29 32 25 54 28 38 44  00001011 01111100 11110101 01101010 00100101 10101111
49 26 30 23 52 24 22 34  10001001 11111110 01100000 00110110 00011100 01101111
63  4 53  7 64  3 60  6  10101010 10101010 10001010 10111001 11011110 01001011
56 47 20 46  8 51 15 17  11010100 10100101 01010010 11011010 11101110 10111000
11 37 27 21 55 36 33 40  01001111 00111000 10100000 01011001 10101101 00000101
13 41 43 31 16 39 35 42  01100111 00010000 11111001 10011100 00111110 00001001
                       33#  /D2i 32*      16*       8*       4*       2*       1*
 1 62  5 59  2 61 12 58  01010101 01010101 01010111 01100100 00010010 01001011
57 14 50 48  9 18 45 19  10110010 10100101 11011010 01010010 00010001 01110100
10 30 34 22 54 28 38 44  00101011 01011100 11000101 01011010 00000101 11111111
49 26 29 24 52 25 23 32  10001000 11111111 01100101 00110011 00011011 01011001
63  4 53  7 64  3 60  6  10101010 10101010 10001010 10111001 11011110 01001011
56 47 20 46  8 51 15 17  11010100 10100101 01010010 11011010 11101110 10111000
11 37 27 21 55 35 31 43  01001101 00111010 10100011 01011010 10101111 00000000
13 40 42 33 16 39 36 41  01110111 00000000 10101001 11001100 01001110 01101010
.....
```

それに比べると，下にあげた例は，まことに端正だ。
どの行列もどの汎対角線も，完璧にオイラー型だ。
こういう**完全オイラー方陣**は，何種類もあるが，どのタイプも数が少ない。

＊＊　完全オイラー八方陣の例　＊＊

```
                        1#  /D2i 32*      16*       8*       4*       2*       1*
 1 63  6 60 14 52  9 55  01010101 01010101 01011010 01101001 01010101 00111100
62  4 57  7 49 15 54 12  10101010 10101010 10100101 10010110 01010101 11000011
19 45 24 42 32 34 27 37  01010101 10101010 01011010 01101001 10101010 00111100
48 18 43 21 35 29 40 26  10101010 01010101 10100101 10010110 10101010 11000011
51 13 56 10 64  2 59  5  10101010 10101010 01011010 01101001 10101010 00111100
16 50 11 53  3 61  8 58  01010101 01010101 10100101 10010110 10101010 11000011
33 31 38 28 46 20 41 23  10101010 01010101 01011010 01101001 01010101 00111100
30 36 25 39 17 47 22 44  01010101 10101010 10100101 10010110 01010101 11000011
                        2#  /D2i 32*      16*       8*       4*       2*       1*
 1 63  6 60 22 44 17 47  01010101 01011010 01010101 01101001 01010101 00111100
62  4 57  7 41 23 46 20  10101010 10100101 10101010 10010110 01010101 11000011
11 53 16 50 32 34 27 37  01010101 01011010 10101010 01101001 10101010 00111100
56 10 51 13 35 29 40 26  10101010 10100101 01010101 10010110 10101010 11000011
43 21 48 18 64  2 59  5  10101010 01011010 10101010 01101001 10101010 00111100
24 42 19 45  3 61  8 58  01010101 10100101 01010101 10010110 10101010 11000011
33 31 38 28 54 12 49 15  10101010 01011010 01010101 01101001 01010101 00111100
30 36 25 39  9 55 14 52  01010101 10100101 10101010 10010110 01010101 11000011
                        3#  /D2i 32*      16*       8*       4*       2*       1*
 1 63  6 60 38 28 33 31  01011010 01010101 01010101 01101001 01010101 00111100
62  4 57  7 25 39 30 36  10100101 10101010 10101010 10010110 01010101 11000011
11 53 16 50 48 18 43 21  01011010 01010101 10101010 01101001 10101010 00111100
56 10 51 13 19 45 24 42  10100101 10101010 01010101 10010110 10101010 11000011
27 37 32 34 64  2 59  5  01011010 10101010 10101010 01101001 10101010 00111100
40 26 35 29  3 61  8 58  10100101 01010101 01010101 10010110 10101010 11000011
17 47 22 44 54 12 49 15  01011010 10101010 01010101 01101001 01010101 00111100
46 20 41 23  9 55 14 52  10100101 01010101 10101010 10010110 01010101 11000011
.....
```

前節の研究で，八方陣を二進法分解すると，各単位面の行列を定和にした場合，数字のパターンが｛0，0，0，0，1，1，1，1｝に限られるとした。そのために八次では，オイラー方陣が作りやすいと説明した。ところが，事実はそうなっていない。

別の説明が必要なことがわかる。

ここで，前節の答案を再録・補充する。三，四，八次については，順列の数も数えた。

＊＊　定和の時の数字パターン　＊＊

［三方陣：三進法］

```
If [n1+n2+n3==3] then [Comb{n1,n2,n3}={0,1,2},{1,1,1}]
If [n1+n2+n3==3] then [Perm{n1,n2,n3}=
{0,1,2},{0,2,1},{1,0,2},{1,1,1},{1,2,0},{2,0,1},{2,1,0}]
* Count = 7
```

［四方陣：二進法］

```
If [n1+n2+n3+n4==2] then [Comb{n1,n2,n3,n4}={0,0,1,1}]
If [n1+n2+n3+n4==2] then [Perm{n1,n2,n3,n4}=
{0,0,1,1},{0,1,0,1},{0,1,1,0},{1,0,0,1},{1,0,1,0},{1,1,0,0}]
* Count = 6
```

［五方陣：五進法］

```
If [n1+n2+n3+n4+n5==10] then [Comb{n1,n2,n3,n4,n5}=
 {0,0,2,4,4}, {0,0,3,3,4}, {0,1,1,4,4}, {0,1,2,3,4}, {0,1,3,3,3},
 {0,2,2,2,4}, {0,2,2,3,3}, {1,1,1,3,4}, {1,1,2,2,4}, {1,1,2,3,3},
 {1,2,2,2,3}, {2,2,2,2,2}]
* Count = 12
```

［七方陣：七進法］

```
If [n1+n2+n3+n4+n5+n6+n7==21] then [Comb{n1,n2,n3,n4,n5,n6,n7}=
 {0,0,0,3,6,6,6}, {0,0,0,4,5,6,6}, {0,0,0,5,5,5,6}, {0,0,1,2,6,6,6},
```

```
 {0,0,1,3,5,6,6}, {0,0,1,4,4,6,6}, {0,0,1,4,5,5,6}, {0,0,1,5,5,5,5},
 {0,0,2,2,5,6,6}, {0,0,2,3,4,6,6}, {0,0,2,3,5,5,6}, {0,0,2,4,4,5,6},
 {0,0,2,4,5,5,5}, {0,0,3,3,3,6,6}, {0,0,3,3,4,5,6}, {0,0,3,3,5,5,5},
 {0,0,3,4,4,4,6}, {0,0,3,4,4,5,5}, {0,0,4,4,4,4,5}, {0,1,1,1,6,6,6},
 {0,1,1,2,5,6,6}, {0,1,1,3,4,6,6}, {0,1,1,3,5,5,6}, {0,1,1,4,4,5,6},
 {0,1,1,4,5,5,5}, {0,1,2,2,4,6,6}, {0,1,2,2,5,5,6}, {0,1,2,3,3,6,6},
 {0,1,2,3,4,5,6}, {0,1,2,3,5,5,5}, {0,1,2,4,4,4,6}, {0,1,2,4,4,5,5},
 {0,1,3,3,3,5,6}, {0,1,3,3,4,4,6}, {0,1,3,3,4,5,5}, {0,1,3,4,4,4,5},
 {0,1,4,4,4,4,4}, {0,2,2,2,3,6,6}, {0,2,2,2,4,5,6}, {0,2,2,2,5,5,5},
 {0,2,2,3,3,5,6}, {0,2,2,3,4,4,6}, {0,2,2,3,4,5,5}, {0,2,2,4,4,4,5},
 {0,2,3,3,3,4,6}, {0,2,3,3,3,5,5}, {0,2,3,3,4,4,5}, {0,2,3,4,4,4,4},
 {0,3,3,3,3,3,6}, {0,3,3,3,3,4,5}, {0,3,3,3,4,4,4}, {1,1,1,1,5,6,6},
 {1,1,1,2,4,6,6}, {1,1,1,2,5,5,6}, {1,1,1,3,3,6,6}, {1,1,1,3,4,5,6},
 {1,1,1,3,5,5,5}, {1,1,1,4,4,4,6}, {1,1,1,4,4,5,5}, {1,1,2,2,3,6,6},
 {1,1,2,2,4,5,6}, {1,1,2,2,5,5,5}, {1,1,2,3,3,5,6}, {1,1,2,3,4,4,6},
 {1,1,2,3,4,5,5}, {1,1,2,4,4,4,5}, {1,1,3,3,3,4,6}, {1,1,3,3,3,5,5},
 {1,1,3,3,4,4,5}, {1,1,3,4,4,4,4}, {1,2,2,2,2,6,6}, {1,2,2,2,3,5,6},
 {1,2,2,2,4,4,6}, {1,2,2,2,4,5,5}, {1,2,2,3,3,4,6}, {1,2,2,3,3,5,5},
 {1,2,2,3,4,4,5}, {1,2,2,4,4,4,4}, {1,2,3,3,3,3,6}, {1,2,3,3,3,4,5},
 {1,2,3,3,4,4,4}, {1,3,3,3,3,3,5}, {1,3,3,3,3,4,4}, {2,2,2,2,2,5,6},
 {2,2,2,2,3,4,6}, {2,2,2,2,3,5,5}, {2,2,2,2,4,4,5}, {2,2,2,3,3,3,6},
 {2,2,2,3,3,4,5}, {2,2,2,3,4,4,4}, {2,2,3,3,3,3,5}, {2,2,3,3,3,4,4},
 {2,3,3,3,3,3,4}, {3,3,3,3,3,3,3}]
* Count = 94
```

［八方陣：二進法］

```
If [n1+n2+n3+n4+n5+n6+n7+n8==4] then [Comb{n1,n2,n3,n4,n5,n6,n7,n8}=
 {0,0,0,0,1,1,1,1}]
If [n1+n2+n3+n4+n5+n6+n7+n8==4] then [Perm{n1,n2,n3,n4,n5,n6,n7,n8}=
 {0,0,0,0,1,1,1,1}, {0,0,0,1,0,1,1,1}, {0,0,0,1,1,0,1,1}, {0,0,0,1,1,1,0,1},
 {0,0,0,1,1,1,1,0}, {0,0,1,0,0,1,1,1}, {0,0,1,0,1,0,1,1}, {0,0,1,0,1,1,0,1},
 {0,0,1,0,1,1,1,0}, {0,0,1,1,0,0,1,1}, {0,0,1,1,0,1,0,1}, {0,0,1,1,0,1,1,0},
 {0,0,1,1,1,0,0,1}, {0,0,1,1,1,0,1,0}, {0,0,1,1,1,1,0,0}, {0,1,0,0,0,1,1,1},
 {0,1,0,0,1,0,1,1}, {0,1,0,0,1,1,0,1}, {0,1,0,0,1,1,1,0}, {0,1,0,1,0,0,1,1},
 {0,1,0,1,0,1,0,1}, {0,1,0,1,0,1,1,0}, {0,1,0,1,1,0,0,1}, {0,1,0,1,1,0,1,0},
 {0,1,0,1,1,1,0,0}, {0,1,1,0,0,0,1,1}, {0,1,1,0,0,1,0,1}, {0,1,1,0,0,1,1,0},
 {0,1,1,0,1,0,0,1}, {0,1,1,0,1,0,1,0}, {0,1,1,0,1,1,0,0}, {0,1,1,1,0,0,0,1},
 {0,1,1,1,0,0,1,0}, {0,1,1,1,0,1,0,0}, {0,1,1,1,1,0,0,0}, {1,0,0,0,0,1,1,1},
 {1,0,0,0,1,0,1,1}, {1,0,0,0,1,1,0,1}, {1,0,0,0,1,1,1,0}, {1,0,0,1,0,0,1,1},
 {1,0,0,1,0,1,0,1}, {1,0,0,1,0,1,1,0}, {1,0,0,1,1,0,0,1}, {1,0,0,1,1,0,1,0},
 {1,0,0,1,1,1,0,0}, {1,0,1,0,0,0,1,1}, {1,0,1,0,0,1,0,1}, {1,0,1,0,0,1,1,0},
 {1,0,1,0,1,0,0,1}, {1,0,1,0,1,0,1,0}, {1,0,1,0,1,1,0,0}, {1,0,1,1,0,0,0,1},
 {1,0,1,1,0,0,1,0}, {1,0,1,1,0,1,0,0}, {1,0,1,1,1,0,0,0}, {1,1,0,0,0,0,1,1},
 {1,1,0,0,0,1,0,1}, {1,1,0,0,0,1,1,0}, {1,1,0,0,1,0,0,1}, {1,1,0,0,1,0,1,0},
 {1,1,0,0,1,1,0,0}, {1,1,0,1,0,0,0,1}, {1,1,0,1,0,0,1,0}, {1,1,0,1,0,1,0,0},
 {1,1,0,1,1,0,0,0}, {1,1,1,0,0,0,0,1}, {1,1,1,0,0,0,1,0}, {1,1,1,0,0,1,0,0},
 {1,1,1,0,1,0,0,0}, {1,1,1,1,0,0,0,0}]
* Count = 70
```

［九方陣：三進法］

```
If [n1+n2+n3+n4+n5+n6+n7+n8+n9==9] then [Comb{n1,n2,n3,n4,n5,n6,n7,n8,n9}=
 {0,0,0,0,1,2,2,2,2}, {0,0,0,1,1,1,2,2,2}, {0,0,1,1,1,1,1,2,2},
 {0,1,1,1,1,1,1,1,2}, {1,1,1,1,1,1,1,1,1}]
* Count = 5
```

八次の場合，組み合せでも順列でも，確かに全てがオイラー型になっている。これでは，オイラー型しか作れないではないかと思いがちだが，事実はそうなっていない。

「各面で定和が成り立つ」という仮定そのものが崩れているためだ。「ノン - オイラー方陣」では，各面で定和が成り立たなくても別にかまわないのである。全体で定和が成り立て

ば，それで十分である。
　そこで，今度は「全体で定和が成り立つ」具体的な条件を調べてみた。
　二進法分解図は，そのまま用いる。いま解の任意の行列の和が，いつもＳ(＝252）になるとする。
　そのとき分解図の各面で対応する行列の和の関係式を，次のように表わす。
　An＊32＋Bn＊16＋Cn＊８＋Dn＊４＋En＊２＋Fn＝Sn（n＝1，2，3，4，...，15，16）
　もちろん｛An，Bn，Cn，Dn，En，Fn｝の取りうる値は，｛0，1，2，3，4，5，6，7，8｝のいずれかで，同じ値を何回取ってもかまわないとする。
　何通りの解が可能か？

　下のリストは，各次数について調べた「模範答案」だ。

＊＊　全体で定和なら，各面の行列の和が取りうる値は？　＊＊

［三方陣：三進法］
```
If [3*An+Bn==12] then [{An,Bn}={2,6}, {3,3}, {4,0}]
* Count = 3
```

［四方陣：二進法］
```
If [8*An+4*Bn+2*Cn+Dn==30] then
[{An,Bn,Cn,Dn}={1,3,3,4},{1,3,4,2},{1,4,1,4},{1,4,2,2},{1,4,3,0},{2,1,3,4},
{2,1,4,2},{2,2,1,4},{2,2,2,2},{2,2,3,0},{2,3,0,2},{2,3,1,0},
{3,0,1,4},{3,0,2,2},{3,0,3,0},{3,1,0,2},{3,1,1,0}]
* Count = 17
```

［五方陣：五進法］
```
If [5*An+Bn==60] then [{An,Bn}=
 {8,20}, {9,15}, {10,10}, {11,5}, {12,0}]
* Count = 5
```

［七方陣：七進法］
```
If [7*An+Bn==168] then [{An,Bn}=
 {18,42}, {19,35}, {20,28}, {21,21}, {22,14}, {23,7}, {24,0}]
* Count = 7
```

［八方陣：二進法］
```
If [32*An+16*Bn+8*Cn+4*Dn+2*En+Fn==252] then [{An,Bn,Cn,Dn,En,Fn}=
 {1,7,7,7,8,8}, {1,7,7,8,6,8}, {1,7,7,8,7,6}, {1,7,7,8,8,4}, {1,7,8,5,8,8},
 {1,7,8,6,6,8}, {1,7,8,6,7,6}, {1,7,8,6,8,4}, {1,7,8,7,4,8}, {1,7,8,7,5,6},
 {1,7,8,7,6,4}, {1,7,8,7,7,2}, {1,7,8,7,8,0}, {1,7,8,8,2,8}, {1,7,8,8,3,6},
 {1,7,8,8,4,4}, {1,7,8,8,5,2}, {1,7,8,8,6,0}, {1,8,5,7,8,8}, {1,8,5,8,6,8},
 {1,8,5,8,7,6}, {1,8,5,8,8,4}, {1,8,6,5,8,8}, {1,8,6,6,6,8}, {1,8,6,6,7,6},
 {1,8,6,6,8,4}, {1,8,6,7,4,8}, {1,8,6,7,5,6}, {1,8,6,7,6,4}, {1,8,6,7,7,2},
 {1,8,6,7,8,0}, {1,8,6,8,2,8}, {1,8,6,8,3,6}, {1,8,6,8,4,4}, {1,8,6,8,5,2},
 {1,8,6,8,6,0}, {1,8,7,3,8,8}, {1,8,7,4,6,8}, {1,8,7,4,7,6}, {1,8,7,4,8,4},
 {1,8,7,5,4,8}, {1,8,7,5,5,6}, {1,8,7,5,6,4}, {1,8,7,5,7,2}, {1,8,7,5,8,0},
 {1,8,7,6,2,8}, {1,8,7,6,3,6}, {1,8,7,6,4,4}, {1,8,7,6,5,2}, {1,8,7,6,6,0},
 {1,8,7,7,0,8}, {1,8,7,7,1,6}, {1,8,7,7,2,4}, {1,8,7,7,3,2}, {1,8,7,7,4,0},
 {1,8,7,8,0,4}, {1,8,7,8,1,2}, {1,8,7,8,2,0}, {1,8,8,1,8,8}, {1,8,8,2,6,8},
 {1,8,8,2,7,6}, {1,8,8,2,8,4}, {1,8,8,3,4,8}, {1,8,8,3,5,6}, {1,8,8,3,6,4},
 {1,8,8,3,7,2}, {1,8,8,3,8,0}, {1,8,8,4,2,8}, {1,8,8,4,3,6}, {1,8,8,4,4,4},
 {1,8,8,4,5,2}, {1,8,8,4,6,0}, {1,8,8,5,0,8}, {1,8,8,5,1,6}, {1,8,8,5,2,4},
 {1,8,8,5,3,2}, {1,8,8,5,4,0}, {1,8,8,6,0,4}, {1,8,8,6,1,2}, {1,8,8,6,2,0},
 {1,8,8,7,0,0}, {2,5,7,7,8,8}, {2,5,7,8,6,8}, {2,5,7,8,7,6}, {2,5,7,8,8,4},
 {2,5,8,5,8,8}, {2,5,8,6,6,8}, {2,5,8,6,7,6}, {2,5,8,6,8,4}, {2,5,8,7,4,8},

 {3,8,3,0,0,4}, {3,8,3,0,1,2}, {3,8,3,0,2,0}, {3,8,3,1,0,0}, {4,1,7,7,8,8},
 {4,1,7,8,6,8}, {4,1,7,8,7,6}, {4,1,7,8,8,4}, {4,1,8,5,8,8}, {4,1,8,6,6,8},
```

{4,1,8,6,7,6}, {4,1,8,6,8,4}, {4,1,8,7,4,8}, {4,1,8,7,5,6}, {4,1,8,7,6,4},
{4,1,8,7,7,2}, {4,1,8,7,8,0}, {4,1,8,8,2,8}, {4,1,8,8,3,6}, {4,1,8,8,4,4},
{4,1,8,8,5,2}, {4,1,8,8,6,0}, {4,2,5,7,8,8}, {4,2,5,8,6,8}, {4,2,5,8,7,6},
{4,2,5,8,8,4}, {4,2,6,5,8,8}, {4,2,6,6,6,8}, {4,2,6,6,7,6}, {4,2,6,6,8,4},
{4,2,6,7,4,8}, {4,2,6,7,5,6}, {4,2,6,7,6,4}, {4,2,6,7,7,2}, {4,2,6,7,8,0},
{4,2,6,8,2,8}, {4,2,6,8,3,6}, {4,2,6,8,4,4}, {4,2,6,8,5,2}, {4,2,6,8,6,0},
{4,2,7,3,8,8}, {4,2,7,4,6,8}, {4,2,7,4,7,6}, {4,2,7,4,8,4}, {4,2,7,5,4,8},
{4,2,7,5,5,6}, {4,2,7,5,6,4}, {4,2,7,5,7,2}, {4,2,7,5,8,0}, {4,2,7,6,2,8},
{4,2,7,6,3,6}, {4,2,7,6,4,4}, {4,2,7,6,5,2}, {4,2,7,6,6,0}, {4,2,7,7,0,8},
{4,2,7,7,1,6}, {4,2,7,7,2,4}, {4,2,7,7,3,2}, {4,2,7,7,4,0}, {4,2,7,8,0,4},
{4,2,7,8,1,2}, {4,2,7,8,2,0}, {4,2,8,1,8,8}, {4,2,8,2,6,8}, {4,2,8,2,7,6},
{4,2,8,2,8,4}, {4,2,8,3,4,8}, {4,2,8,3,5,6}, {4,2,8,3,6,4}, {4,2,8,3,7,2},
{4,2,8,3,8,0}, {4,2,8,4,2,8}, {4,2,8,4,3,6}, {4,2,8,4,4,4}, {4,2,8,4,5,2},
{4,2,8,4,6,0}, {4,2,8,5,0,8}, {4,2,8,5,1,6}, {4,2,8,5,2,4}, {4,2,8,5,3,2},
{4,2,8,5,4,0}, {4,2,8,6,0,4}, {4,2,8,6,1,2}, {4,2,8,6,2,0}, {4,2,8,7,0,0},
{4,3,3,7,8,8}, {4,3,3,8,6,8}, {4,3,3,8,7,6}, {4,3,3,8,8,4}, {4,3,4,5,8,8},
{4,3,4,6,6,8}, {4,3,4,6,7,6}, {4,3,4,6,8,4}, {4,3,4,7,4,8}, {4,3,4,7,5,6},
{4,3,4,7,6,4}, {4,3,4,7,7,2}, {4,3,4,7,8,0}, {4,3,4,8,2,8}, {4,3,4,8,3,6},
{4,3,4,8,4,4}, {4,3,4,8,5,2}, {4,3,4,8,6,0}, {4,3,5,3,8,8}, {4,3,5,4,6,8},
{4,3,5,4,7,6}, {4,3,5,4,8,4}, {4,3,5,5,4,8}, {4,3,5,5,5,6}, {4,3,5,5,6,4},
{4,3,5,5,7,2}, {4,3,5,5,8,0}, {4,3,5,6,2,8}, {4,3,5,6,3,6}, {4,3,5,6,4,4},
{4,3,5,6,5,2}, {4,3,5,6,6,0}, {4,3,5,7,0,8}, {4,3,5,7,1,6}, {4,3,5,7,2,4},
{4,3,5,7,3,2}, {4,3,5,7,4,0}, {4,3,5,8,0,4}, {4,3,5,8,1,2}, {4,3,5,8,2,0},
{4,3,6,1,8,8}, {4,3,6,2,6,8}, {4,3,6,2,7,6}, {4,3,6,2,8,4}, {4,3,6,3,4,8},
{4,3,6,3,5,6}, {4,3,6,3,6,4}, {4,3,6,3,7,2}, {4,3,6,3,8,0}, {4,3,6,4,2,8},
{4,3,6,4,3,6}, {4,3,6,4,4,4}, {4,3,6,4,5,2}, {4,3,6,4,6,0}, {4,3,6,5,0,8},
{4,3,6,5,1,6}, {4,3,6,5,2,4}, {4,3,6,5,3,2}, {4,3,6,5,4,0}, {4,3,6,6,0,4},
{4,3,6,6,1,2}, {4,3,6,6,2,0}, {4,3,6,7,0,0}, {4,3,7,0,6,8}, {4,3,7,0,7,6},
{4,3,7,0,8,4}, {4,3,7,1,4,8}, {4,3,7,1,5,6}, {4,3,7,1,6,4}, {4,3,7,1,7,2},
{4,3,7,1,8,0}, {4,3,7,2,2,8}, {4,3,7,2,3,6}, {4,3,7,2,4,4}, {4,3,7,2,5,2},
{4,3,7,2,6,0}, {4,3,7,3,0,8}, {4,3,7,3,1,6}, {4,3,7,3,2,4}, {4,3,7,3,3,2},
{4,3,7,3,4,0}, {4,3,7,4,0,4}, {4,3,7,4,1,2}, {4,3,7,4,2,0}, {4,3,7,5,0,0},
{4,3,8,0,2,8}, {4,3,8,0,3,6}, {4,3,8,0,4,4}, {4,3,8,0,5,2}, {4,3,8,0,6,0},
{4,3,8,1,0,8}, {4,3,8,1,1,6}, {4,3,8,1,2,4}, {4,3,8,1,3,2}, {4,3,8,1,4,0},
{4,3,8,2,0,4}, {4,3,8,2,1,2}, {4,3,8,2,2,0}, {4,3,8,3,0,0}, {4,4,1,7,8,8},
{4,4,1,8,6,8}, {4,4,1,8,7,6}, {4,4,1,8,8,4}, {4,4,2,5,8,8}, {4,4,2,6,6,8},
{4,4,2,6,7,6}, {4,4,2,6,8,4}, {4,4,2,7,4,8}, {4,4,2,7,5,6}, {4,4,2,7,6,4},
{4,4,2,7,7,2}, {4,4,2,7,8,0}, {4,4,2,8,2,8}, {4,4,2,8,3,6}, {4,4,2,8,4,4},
{4,4,2,8,5,2}, {4,4,2,8,6,0}, {4,4,3,3,8,8}, {4,4,3,4,6,8}, {4,4,3,4,7,6},
{4,4,3,4,8,4}, {4,4,3,5,4,8}, {4,4,3,5,5,6}, {4,4,3,5,6,4}, {4,4,3,5,7,2},
{4,4,3,5,8,0}, {4,4,3,6,2,8}, {4,4,3,6,3,6}, {4,4,3,6,4,4}, {4,4,3,6,5,2},
{4,4,3,6,6,0}, {4,4,3,7,0,8}, {4,4,3,7,1,6}, {4,4,3,7,2,4}, {4,4,3,7,3,2},
{4,4,3,7,4,0}, {4,4,3,8,0,4}, {4,4,3,8,1,2}, {4,4,3,8,2,0}, {4,4,4,1,8,8},
{4,4,4,2,6,8}, {4,4,4,2,7,6}, {4,4,4,2,8,4}, {4,4,4,3,4,8}, {4,4,4,3,5,6},
{4,4,4,3,6,4}, {4,4,4,3,7,2}, {4,4,4,3,8,0}, {4,4,4,4,2,8}, {4,4,4,4,3,6},
{4,4,4,4,4,4}, {4,4,4,4,5,2}, {4,4,4,4,6,0}, {4,4,4,5,0,8}, {4,4,4,5,1,6},
{4,4,4,5,2,4}, {4,4,4,5,3,2}, {4,4,4,5,4,0}, {4,4,4,6,0,4}, {4,4,4,6,1,2},
{4,4,4,6,2,0}, {4,4,4,7,0,0}, {4,4,5,0,6,8}, {4,4,5,0,7,6}, {4,4,5,0,8,4},
{4,4,5,1,4,8}, {4,4,5,1,5,6}, {4,4,5,1,6,4}, {4,4,5,1,7,2}, {4,4,5,1,8,0},
{4,4,5,2,2,8}, {4,4,5,2,3,6}, {4,4,5,2,4,4}, {4,4,5,2,5,2}, {4,4,5,2,6,0},
{4,4,5,3,0,8}, {4,4,5,3,1,6}, {4,4,5,3,2,4}, {4,4,5,3,3,2}, {4,4,5,3,4,0},
{4,4,5,4,0,4}, {4,4,5,4,1,2}, {4,4,5,4,2,0}, {4,4,5,5,0,0}, {4,4,6,0,2,8},
{4,4,6,0,3,6}, {4,4,6,0,4,4}, {4,4,6,0,5,2}, {4,4,6,0,6,0}, {4,4,6,1,0,8},
{4,4,6,1,1,6}, {4,4,6,1,2,4}, {4,4,6,1,3,2}, {4,4,6,1,4,0}, {4,4,6,2,0,4},
{4,4,6,2,1,2}, {4,4,6,2,2,0}, {4,4,6,3,0,0}, {4,4,7,0,0,4}, {4,4,7,0,1,2},
{4,4,7,0,2,0}, {4,4,7,1,0,0}, {4,5,0,5,8,8}, {4,5,0,6,6,8}, {4,5,0,6,7,6},
{4,5,0,6,8,4}, {4,5,0,7,4,8}, {4,5,0,7,5,6}, {4,5,0,7,6,4}, {4,5,0,7,7,2},
{4,5,0,7,8,0}, {4,5,0,8,2,8}, {4,5,0,8,3,6}, {4,5,0,8,4,4}, {4,5,0,8,5,2},
{4,5,0,8,6,0}, {4,5,1,3,8,8}, {4,5,1,4,6,8}, {4,5,1,4,7,6}, {4,5,1,4,8,4},
{4,5,1,5,4,8}, {4,5,1,5,5,6}, {4,5,1,5,6,4}, {4,5,1,5,7,2}, {4,5,1,5,8,0},

{4,5,1,6,2,8}, {4,5,1,6,3,6}, {4,5,1,6,4,4}, {4,5,1,6,5,2}, {4,5,1,6,6,0},
{4,5,1,7,0,8}, {4,5,1,7,1,6}, {4,5,1,7,2,4}, {4,5,1,7,3,2}, {4,5,1,7,4,0},
{4,5,1,8,0,4}, {4,5,1,8,1,2}, {4,5,1,8,2,0}, {4,5,2,1,8,8}, {4,5,2,2,6,8},
{4,5,2,2,7,6}, {4,5,2,2,8,4}, {4,5,2,3,4,8}, {4,5,2,3,5,6}, {4,5,2,3,6,4},
{4,5,2,3,7,2}, {4,5,2,3,8,0}, {4,5,2,4,2,8}, {4,5,2,4,3,6}, {4,5,2,4,4,4},
{4,5,2,4,5,2}, {4,5,2,4,6,0}, {4,5,2,5,0,8}, {4,5,2,5,1,6}, {4,5,2,5,2,4},
{4,5,2,5,3,2}, {4,5,2,5,4,0}, {4,5,2,6,0,4}, {4,5,2,6,1,2}, {4,5,2,6,2,0},
{4,5,2,7,0,0}, {4,5,3,0,6,8}, {4,5,3,0,7,6}, {4,5,3,0,8,4}, {4,5,3,1,4,8},
{4,5,3,1,5,6}, {4,5,3,1,6,4}, {4,5,3,1,7,2}, {4,5,3,1,8,0}, {4,5,3,2,2,8},
{4,5,3,2,3,6}, {4,5,3,2,4,4}, {4,5,3,2,5,2}, {4,5,3,2,6,0}, {4,5,3,3,0,8},
{4,5,3,3,1,6}, {4,5,3,3,2,4}, {4,5,3,3,3,2}, {4,5,3,3,4,0}, {4,5,3,4,0,4},
{4,5,3,4,1,2}, {4,5,3,4,2,0}, {4,5,3,5,0,0}, {4,5,4,0,2,8}, {4,5,4,0,3,6},
{4,5,4,0,4,4}, {4,5,4,0,5,2}, {4,5,4,0,6,0}, {4,5,4,1,0,8}, {4,5,4,1,1,6},
{4,5,4,1,2,4}, {4,5,4,1,3,2}, {4,5,4,1,4,0}, {4,5,4,2,0,4}, {4,5,4,2,1,2},
{4,5,4,2,2,0}, {4,5,4,3,0,0}, {4,5,5,0,0,4}, {4,5,5,0,1,2}, {4,5,5,0,2,0},
{4,5,5,1,0,0}, {4,6,0,1,8,8}, {4,6,0,2,6,8}, {4,6,0,2,7,6}, {4,6,0,2,8,4},
{4,6,0,3,4,8}, {4,6,0,3,5,6}, {4,6,0,3,6,4}, {4,6,0,3,7,2}, {4,6,0,3,8,0},
{4,6,0,4,2,8}, {4,6,0,4,3,6}, {4,6,0,4,4,4}, {4,6,0,4,5,2}, {4,6,0,4,6,0},
{4,6,0,5,0,8}, {4,6,0,5,1,6}, {4,6,0,5,2,4}, {4,6,0,5,3,2}, {4,6,0,5,4,0},
{4,6,0,6,0,4}, {4,6,0,6,1,2}, {4,6,0,6,2,0}, {4,6,0,7,0,0}, {4,6,1,0,6,8},
{4,6,1,0,7,6}, {4,6,1,0,8,4}, {4,6,1,1,4,8}, {4,6,1,1,5,6}, {4,6,1,1,6,4},
{4,6,1,1,7,2}, {4,6,1,1,8,0}, {4,6,1,2,2,8}, {4,6,1,2,3,6}, {4,6,1,2,4,4},
{4,6,1,2,5,2}, {4,6,1,2,6,0}, {4,6,1,3,0,8}, {4,6,1,3,1,6}, {4,6,1,3,2,4},
{4,6,1,3,3,2}, {4,6,1,3,4,0}, {4,6,1,4,0,4}, {4,6,1,4,1,2}, {4,6,1,4,2,0},
{4,6,1,5,0,0}, {4,6,2,0,2,8}, {4,6,2,0,3,6}, {4,6,2,0,4,4}, {4,6,2,0,5,2},
{4,6,2,0,6,0}, {4,6,2,1,0,8}, {4,6,2,1,1,6}, {4,6,2,1,2,4}, {4,6,2,1,3,2},
{4,6,2,1,4,0}, {4,6,2,2,0,4}, {4,6,2,2,1,2}, {4,6,2,2,2,0}, {4,6,2,3,0,0},
{4,6,3,0,0,4}, {4,6,3,0,1,2}, {4,6,3,0,2,0}, {4,6,3,1,0,0}, {4,7,0,0,2,8},
{4,7,0,0,3,6}, {4,7,0,0,4,4}, {4,7,0,0,5,2}, {4,7,0,0,6,0}, {4,7,0,1,0,8},
{4,7,0,1,1,6}, {4,7,0,1,2,4}, {4,7,0,1,3,2}, {4,7,0,1,4,0}, {4,7,0,2,0,4},
{4,7,0,2,1,2}, {4,7,0,2,2,0}, {4,7,0,3,0,0}, {4,7,1,0,0,4}, {4,7,1,0,1,2},
{4,7,1,0,2,0}, {4,7,1,1,0,0}, {5,0,5,7,8,8}, {5,0,5,8,6,8}, {5,0,5,8,7,6},

{7,0,0,1,8,8}, {7,0,0,2,6,8}, {7,0,0,2,7,6}, {7,0,0,2,8,4}, {7,0,0,3,4,8},
{7,0,0,3,5,6}, {7,0,0,3,6,4}, {7,0,0,3,7,2}, {7,0,0,3,8,0}, {7,0,0,4,2,8},
{7,0,0,4,3,6}, {7,0,0,4,4,4}, {7,0,0,4,5,2}, {7,0,0,4,6,0}, {7,0,0,5,0,8},
{7,0,0,5,1,6}, {7,0,0,5,2,4}, {7,0,0,5,3,2}, {7,0,0,5,4,0}, {7,0,0,6,0,4},
{7,0,0,6,1,2}, {7,0,0,6,2,0}, {7,0,0,7,0,0}, {7,0,1,0,6,8}, {7,0,1,0,7,6},
{7,0,1,0,8,4}, {7,0,1,1,4,8}, {7,0,1,1,5,6}, {7,0,1,1,6,4}, {7,0,1,1,7,2},
{7,0,1,1,8,0}, {7,0,1,2,2,8}, {7,0,1,2,3,6}, {7,0,1,2,4,4}, {7,0,1,2,5,2},
{7,0,1,2,6,0}, {7,0,1,3,0,8}, {7,0,1,3,1,6}, {7,0,1,3,2,4}, {7,0,1,3,3,2},
{7,0,1,3,4,0}, {7,0,1,4,0,4}, {7,0,1,4,1,2}, {7,0,1,4,2,0}, {7,0,1,5,0,0},
{7,0,2,0,2,8}, {7,0,2,0,3,6}, {7,0,2,0,4,4}, {7,0,2,0,5,2}, {7,0,2,0,6,0},
{7,0,2,1,0,8}, {7,0,2,1,1,6}, {7,0,2,1,2,4}, {7,0,2,1,3,2}, {7,0,2,1,4,0},
{7,0,2,2,0,4}, {7,0,2,2,1,2}, {7,0,2,2,2,0}, {7,0,2,3,0,0}, {7,0,3,0,0,4},
{7,0,3,0,1,2}, {7,0,3,0,2,0}, {7,0,3,1,0,0}, {7,1,0,0,2,8}, {7,1,0,0,3,6},
{7,1,0,0,4,4}, {7,1,0,0,5,2}, {7,1,0,0,6,0}, {7,1,0,1,0,8}, {7,1,0,1,1,6},
{7,1,0,1,2,4}, {7,1,0,1,3,2}, {7,1,0,1,4,0}, {7,1,0,2,0,4}, {7,1,0,2,1,2},
{7,1,0,2,2,0}, {7,1,0,3,0,0}, {7,1,1,0,0,4}, {7,1,1,0,1,2}, {7,1,1,0,2,0},
{7,1,1,1,0,0}]
* Count = 2081

[九方陣：三進法]
If [27*An+9*Bn+3*Cn+Dn==360] then [{An,Bn,Cn,Dn}=
{ 5,17,18,18}, { 5,18,15,18}, { 5,18,16,15}, { 5,18,17,12}, { 5,18,18, 9},
{ 6,14,18,18}, { 6,15,15,18}, { 6,15,16,15}, { 6,15,17,12}, { 6,15,18, 9},
{ 6,16,12,18}, { 6,16,13,15}, { 6,16,14,12}, { 6,16,15, 9}, { 6,16,16, 6},
{ 6,16,17, 3}, { 6,16,18, 0}, { 6,17, 9,18}, { 6,17,10,15}, { 6,17,11,12},
{ 6,17,12, 9}, { 6,17,13, 6}, { 6,17,14, 3}, { 6,17,15, 0}, { 6,18, 6,18},
{ 6,18, 7,15}, { 6,18, 8,12}, { 6,18, 9, 9}, { 6,18,10, 6}, { 6,18,11, 3},

```
 { 6,18,12, 0}, { 7,11,18,18}, { 7,12,15,18}, { 7,12,16,15}, { 7,12,17,12},
 { 7,12,18, 9}, { 7,13,12,18}, { 7,13,13,15}, { 7,13,14,12}, { 7,13,15, 9},
 { 7,13,16, 6}, { 7,13,17, 3}, { 7,13,18, 0}, { 7,14, 9,18}, { 7,14,10,15},
 { 7,14,11,12}, { 7,14,12, 9}, { 7,14,13, 6}, { 7,14,14, 3}, { 7,14,15, 0},
 { 7,15, 6,18}, { 7,15, 7,15}, { 7,15, 8,12}, { 7,15, 9, 9}, { 7,15,10, 6},
 { 7,15,11, 3}, { 7,15,12, 0}, { 7,16, 3,18}, { 7,16, 4,15}, { 7,16, 5,12},
 { 7,16, 6, 9}, { 7,16, 7, 6}, { 7,16, 8, 3}, { 7,16, 9, 0}, { 7,17, 0,18},
 { 7,17, 1,15}, { 7,17, 2,12}, { 7,17, 3, 9}, { 7,17, 4, 6}, { 7,17, 5, 3},
 { 7,17, 6, 0}, { 7,18, 0, 9}, { 7,18, 1, 6}, { 7,18, 2, 3}, { 7,18, 3, 0},
 { 8, 8,18,18}, { 8, 9,15,18}, { 8, 9,16,15}, { 8, 9,17,12}, { 8, 9,18, 9},
 { 8,10,12,18}, { 8,10,13,15}, { 8,10,14,12}, { 8,10,15, 9}, { 8,10,16, 6},
 { 8,10,17, 3}, { 8,10,18, 0}, { 8,11, 9,18}, { 8,11,10,15}, { 8,11,11,12},
 { 8,11,12, 9}, { 8,11,13, 6}, { 8,11,14, 3}, { 8,11,15, 0}, { 8,12, 6,18},
 { 8,12, 7,15}, { 8,12, 8,12}, { 8,12, 9, 9}, { 8,12,10, 6}, { 8,12,11, 3},
 { 8,12,12, 0}, { 8,13, 3,18}, { 8,13, 4,15}, { 8,13, 5,12}, { 8,13, 6, 9},
 { 8,13, 7, 6}, { 8,13, 8, 3}, { 8,13, 9, 0}, { 8,14, 0,18}, { 8,14, 1,15},
 { 8,14, 2,12}, { 8,14, 3, 9}, { 8,14, 4, 6}, { 8,14, 5, 3}, { 8,14, 6, 0},
 { 8,15, 0, 9}, { 8,15, 1, 6}, { 8,15, 2, 3}, { 8,15, 3, 0}, { 8,16, 0, 0},
 { 9, 5,18,18}, { 9, 6,15,18}, { 9, 6,16,15}, { 9, 6,17,12}, { 9, 6,18, 9},
 { 9, 7,12,18}, { 9, 7,13,15}, { 9, 7,14,12}, { 9, 7,15, 9}, { 9, 7,16, 6},
 { 9, 7,17, 3}, { 9, 7,18, 0}, { 9, 8, 9,18}, { 9, 8,10,15}, { 9, 8,11,12},
 { 9, 8,12, 9}, { 9, 8,13, 6}, { 9, 8,14, 3}, { 9, 8,15, 0}, { 9, 9, 6,18},
 { 9, 9, 7,15}, { 9, 9, 8,12}, { 9, 9, 9, 9}, { 9, 9,10, 6}, { 9, 9,11, 3},
 { 9, 9,12, 0}, { 9,10, 3,18}, { 9,10, 4,15}, { 9,10, 5,12}, { 9,10, 6, 9},
 { 9,10, 7, 6}, { 9,10, 8, 3}, { 9,10, 9, 0}, { 9,11, 0,18}, { 9,11, 1,15},
 { 9,11, 2,12}, { 9,11, 3, 9}, { 9,11, 4, 6}, { 9,11, 5, 3}, { 9,11, 6, 0},
 { 9,12, 0, 9}, { 9,12, 1, 6}, { 9,12, 2, 3}, { 9,12, 3, 0}, { 9,13, 0, 0},
 {10, 2,18,18}, {10, 3,15,18}, {10, 3,16,15}, {10, 3,17,12}, {10, 3,18, 9},
 {10, 4,12,18}, {10, 4,13,15}, {10, 4,14,12}, {10, 4,15, 9}, {10, 4,16, 6},
 {10, 4,17, 3}, {10, 4,18, 0}, {10, 5, 9,18}, {10, 5,10,15}, {10, 5,11,12},
 {10, 5,12, 9}, {10, 5,13, 6}, {10, 5,14, 3}, {10, 5,15, 0}, {10, 6, 6,18},
 {10, 6, 7,15}, {10, 6, 8,12}, {10, 6, 9, 9}, {10, 6,10, 6}, {10, 6,11, 3},
 {10, 6,12, 0}, {10, 7, 3,18}, {10, 7, 4,15}, {10, 7, 5,12}, {10, 7, 6, 9},
 {10, 7, 7, 6}, {10, 7, 8, 3}, {10, 7, 9, 0}, {10, 8, 0,18}, {10, 8, 1,15},
 {10, 8, 2,12}, {10, 8, 3, 9}, {10, 8, 4, 6}, {10, 8, 5, 3}, {10, 8, 6, 0},
 {10, 9, 0, 9}, {10, 9, 1, 6}, {10, 9, 2, 3}, {10, 9, 3, 0}, {10,10, 0, 0},
 {11, 0,15,18}, {11, 0,16,15}, {11, 0,17,12}, {11, 0,18, 9}, {11, 1,12,18},
 {11, 1,13,15}, {11, 1,14,12}, {11, 1,15, 9}, {11, 1,16, 6}, {11, 1,17, 3},
 {11, 1,18, 0}, {11, 2, 9,18}, {11, 2,10,15}, {11, 2,11,12}, {11, 2,12, 9},
 {11, 2,13, 6}, {11, 2,14, 3}, {11, 2,15, 0}, {11, 3, 6,18}, {11, 3, 7,15},
 {11, 3, 8,12}, {11, 3, 9, 9}, {11, 3,10, 6}, {11, 3,11, 3}, {11, 3,12, 0},
 {11, 4, 3,18}, {11, 4, 4,15}, {11, 4, 5,12}, {11, 4, 6, 9}, {11, 4, 7, 6},
 {11, 4, 8, 3}, {11, 4, 9, 0}, {11, 5, 0,18}, {11, 5, 1,15}, {11, 5, 2,12},
 {11, 5, 3, 9}, {11, 5, 4, 6}, {11, 5, 5, 3}, {11, 5, 6, 0}, {11, 6, 0, 9},
 {11, 6, 1, 6}, {11, 6, 2, 3}, {11, 6, 3, 0}, {11, 7, 0, 0}, {12, 0, 6,18},
 {12, 0, 7,15}, {12, 0, 8,12}, {12, 0, 9, 9}, {12, 0,10, 6}, {12, 0,11, 3},
 {12, 0,12, 0}, {12, 1, 3,18}, {12, 1, 4,15}, {12, 1, 5,12}, {12, 1, 6, 9},
 {12, 1, 7, 6}, {12, 1, 8, 3}, {12, 1, 9, 0}, {12, 2, 0,18}, {12, 2, 1,15},
 {12, 2, 2,12}, {12, 2, 3, 9}, {12, 2, 4, 6}, {12, 2, 5, 3}, {12, 2, 6, 0},
 {12, 3, 0, 9}, {12, 3, 1, 6}, {12, 3, 2, 3}, {12, 3, 3, 0}, {12, 4, 0, 0},
 {13, 0, 0, 9}, {13, 0, 1, 6}, {13, 0, 2, 3}, {13, 0, 3, 0}, {13, 1, 0, 0}]
* Count = 285
```

いずれのケースでも，オイラー型になるのは，各面で定和が成り立つ場合のただ１通りしかない。

八次の解は，２千個を超える。それぞれの解のそれぞれの構成数について，０と８以外ではまた何通りもの数字の並び方が考えられるから，ノン - オイラー型の解が最後には気が遠くなるほど多数になる。反対にオイラー型になる確率は，どんどん小さくなる。

「ノン - オイラー方陣」が多い理由の１つが，これで説明がつく。

ノン - オイラー方陣は，作るのがかえって難しい。条件が少なくて緩いので，絞り切れない。コンピュータに探索・計数させても，時間ばかり食う。数もすぐに何億個となる。
解の数が多過ぎるのは，リスト出力にも検索にも，構造解析にも適さない。メモリー上にデータを置けない。たとえ解の総数を何らかの積の形で表わし得るとしても，その約数が重要な「要素」であることを意味する保証はどこにも無い。第一，それをテストすることも確認することもできない。
事実上不可能なことをあえて求める必要は無いと，小生は思う。ｎ１＝１の標準解や基本解だけを調べても埒（らち）があかないケースは，小生はいつもあっさり諦めることにしている。
だから，七次以上の魔方陣では，ノン - オイラー方陣を具体的に調べる必要は無い。オイラー方陣だけを調べるだけで良い，と思う。小生の直観で極言すれば，ノン - オイラー方陣に「大したもの」は無い。
しかし，オイラー方陣の中には，どうしても解の総数を知りたいタイプがある。たとえ何億個になろうとも，数え切りたい魔方陣がある。両立型の対称汎魔方陣のような特別な魔方陣が既に見つかっている場合，それらを部分集合として含むような母集合のスケールはぜひ知りたいものだ。
だが，汎魔方陣や対称魔方陣の総数は，七次以上の高次の場合はなかなか決定できない。たとえオイラー型でも，数が多くて，計数はいつでも大事業になる。単位面の総数だけでも千個を超えるような八方陣は，もう小生の旧型パソコンには手が負えない。
ハード／ソフトの両面で技術が進歩して，数え切れる日が早く来てくれることを願ってやまない。

そんな時に「新オイラー法」が，解の総数は無理としても単位面の総数くらいなら数え切れることを示してくれたのは，大いなる救いだった。概算で全体のスケールが想像できるからである。
一歩譲っても，２つの内どちらの解集合の方が大きいかぐらいは想像することができる。
例えば，単位面が 20 万個以上ある八方陣は，４千個しかない八方陣よりは解の総数が多いに決まっている。…
そんなことがわかるだけでも刺激になって，想像力が大いに膨らむてぇものだ。

（初稿：November 6, 2003；清書稿：March 9, 2013; 執筆: 摂田 寛二;
with MacOSX 10.8.2 and Xcode 4.6; Dictated by Kanji Setsuda）

*** E-Mail Address:<jag12001@nifty.com> ***