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８節：　二進法による「完全オイラー四方陣」の構成

０．研究目的

今回は四次の「完全オイラー方陣」を研究する。

四進法によって汎対角線四方陣(或いは，完全四方陣，正方連結完全四方陣)の全数を構成することは不可能であることが既にわかっているので，二進法によって構成する。

汎四方陣と言うと，良く調べられている研究対象で新味は無いが，方法を実際に実験し，どんなふうにやるのか体験的に熟知しておくことは決して無意味ではないと考えた。

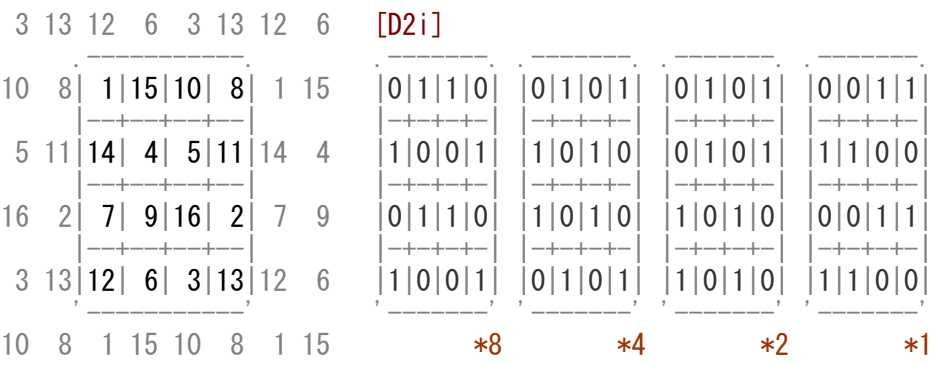
『ギリシャ・ラテン方陣』の構成法は，推奨・喧伝されている割には，実際にどうやるのか具体的で包括的な説明は，寡聞にして今まで見たことがない。たまたま説明があっても，再現ができない。何故にそうしなければならないのか，良くわからない点が多い。

かくなる上は，全部を自前で定義して方法的に開発するしかない，と心に決めた。

１．「完全オイラー方陣」の定義

下の図は，完全四方陣（もしくは汎対角線四方陣，正方連結汎四方陣）の解の一例だ。それに二進法分解図を添えてある。最左面が，８の位の層。次が，４の位の層，２の位の層。最右面が，１の位の層である。

［完全四方陣の解］　　［二進法分解図］



上の各図の対応する数値の間には，次のような等式が必ず成り立つ。

```
 1= 0*8 + 0*4 + 0*2 + 0*1 + 1;   15= 1*8 + 1*4 + 1*2 + 0*1 + 1;
10= 1*8 + 0*4 + 0*2 + 1*1 + 1;    8= 0*8 + 1*4 + 1*2 + 0*1 + 1;
14= 1*8 + 1*4 + 0*2 + 1*1 + 1;    4= 0*8 + 0*4 + 1*2 + 1*1 + 1;
 5= 0*8 + 1*4 + 0*2 + 0*1 + 1;   11= 1*8 + 0*4 + 1*2 + 0*1 + 1;
 . . . . . .
 3= 0*8 + 0*4 + 1*2 + 0*1 + 1;   13= 1*8 + 1*4 + 0*2 + 0*1 + 1;
Vn=An*8 + Bn*4 + Cn*2 + Dn*1 + 1 (n=1,2,3,4,...,15,16)
```

【条件１】分解図のどの層で見ても，各行・各列は {0, 0, 1, 1}，{0, 1, 0, 1}，{0, 1, 1, 0}，{1, 0, 0, 1}，{1, 0, 1, 0}，{1, 1, 0, 0} というように‘０’が２個，‘１’が２個ずつの組み合わせで構成されている。他のパターンが無い。

汎対角線の１つ１つを見ても，全てが同様に構成されている。

ゆえに各４数の和は，例えば次のように計算される。

1+15+10+8＝(0+1+1+0)*8+(0+1+0+1)*4+(0+1+0+1)*2+(0+0+1+1)*1＝2*8+2*4+2*2+2*1
　＝2*(8+4+2+1)＝2*15＝30
1+14+7+12＝(0+1+0+1)*8+(0+1+1+0)*4+(0+0+1+1)*2+(0+1+0+1)*1＝2*(8+4+2+1)＝2*15＝30
1+4+16+13＝(0+0+1+1)*8+(0+0+1+1)*4+(0+1+1+0)*2+(0+1+1+0)*1＝2*(8+4+2+1)＝2*15＝30
. . . . . .

定和となる所以である。なお，二進法分解図は，「数学的表記法」に基づいている。元々が０～15 の整数を素材にしているので，１～16 を素材にする「古典的表記」とは，定和の値が異なる。30(+) は，古典的表記の 34(+) に相当する。
【条件２】分解図の各層とも，‘０’が８個，‘１’が８個ずつの同数で構成されている。
【条件３】分解図から全体図を合成・再構成するには，次のような関係式を利用する。
対応する各ポジション {n, a, b, c, d} の数値の間には，

n1＝a1*8＋b1*4＋c1*2＋d1*1＋１
n2＝a2*8＋b2*4＋c2*2＋d2*1＋１
n3＝a3*8＋b3*4＋c3*2＋d3*1＋１
・・・・・・
n15＝a15*8＋b15*4＋c15*2＋d15*1＋１
n16＝a16*8＋b16*4＋c16*2＋d16*1＋１

という関係がある（全体図が古典的表記の場合）。
合成された全体図の n1～n16 の各数値は，無論，基本素材である自然数１～16 のいずれかでなければならない。その際‘１’が２個も３個もあったり，或いは‘11’が全然無かったりというような，特定の数値の重複や欠落があってはならない。どれもが，厳格に各１度ずつ現れていなければならない。

上の３つの条件を全て満たすものを，我々は特別に『完全オイラー方陣』と呼んでいる。全ての汎対角線でも【条件１】が成り立つことが，特に重要だ。
約 250 年くらい前にオイラー先生御自身が「オイラー方陣」を提唱した際には，汎対角線までは定義しなかったそうだが，我々はあえてそれを要求する。そして，その要求を満たす「オイラー方陣」にこの際「**完全**」の名を冠したい。祖師の意図に反するかもしれないが，こうした方が合理的でより美しい結果を得られることがわかっているからである。

２．完全オイラー四方陣の構成が目標

今回我々のやりたいことは，この定義と二進法分解図を用いて汎対角線四方陣の標準解の全数個を直接に再構成することである。そのために，
（１）先ず分解図の各層になるべき**単位面**を構成する。全部で幾つの単位面が必要か？
（２）出来た単位面を適当に選んで組み合わせ，４層の分解図を作る。それから計算で合成して各解を再構成する。はたして幾つ正解ができるか？
（３）合成した解が定義【条件３】に合うかどうかを，厳密にチェックする。またリスト出力時には，単純な反転・回転形を重複計数しないように，不等式を課してチェックする。

３．単位面の構成

二進法分解図の各層は，いずれも上述の「完全オイラー方陣」の定義・条件に従うものとする。また，汎対角線四方陣の定義にも従うものとする。四次の汎対角線魔方陣の場合は，それ自身が既に完全四方陣でもあり，正方連結陣でもあったから，各層でも補数の配置が全て汎対角線上にある。また内包二方陣の全てが {0, 0, 1, 1} で構成されるものとする。

上図で言うと，n1＋n1＝１； n6＋n6＝１； n4＋n4＝１； n7＋n7＝１； n2＋n2＝１； ...

二進法での「補数」は，全てが「１の補数」である。
また n1＋n2＋n5＋n6＝２； n2＋n3＋n6＋n7＝２； n3＋n4＋n7＋n8＝２； n4＋n1＋n8＋n5＝２； ...

では，実際に単位面を１つ作ってみよう。上の条件の全てに従うとすると，そんなに多数答があるわけではないことに気付く。

1/	2/	3/	4/
0 － － －	0 1 － －	0 1 1 0	0 1 0 1
－ 0 － －	1 0 － －	1 0 0 1	1 0 1 0
－ － 1 －	－ － 1 0	0 1 1 0	1 0 1 0
－ － － 1	－ － 0 1	1 0 0 1	0 1 0 1

主対角線の１本を上の 1/図のように仮定する。すると，補数条件と正方連結条件だけで，第 2/図まで決まる。そして，次に n4＝0 と決めると，後は次々と自動的に決まり，第 3/図が出来上がる。
もし，n4＝1 と決めると，第 4/図が出来上がる。
ところが，第 3/図と第 4/図は，主対角線に関して鏡映反転した形になっていて，厳密には異形とは言い難く，本質的には同じものである。果たして単位面は，１個しか無いのか。
あらためて調査したところ，目的方陣を作るためには，次のような８個の単位面が必要とわかった。しかし，この８個は，お互いにそっくりで，最初の１個から簡単に作れる。汎魔方陣特有の行列の移動と鏡映反転を組み合わせれば，８個はすぐに作れる。
しかも，この８個だけで十分なのである。
これらは，本質的には１個の基本型に帰せられるが，解を合成するための素材としては，別個に認識される。あえて１個にまとめず，このまま使うことにしよう。

＊＊　二進法分解図の「単位面」　＊＊

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

5/	6/	7/	8/
1 1 0 0	1 0 0 1	1 0 1 0	1 0 1 0
0 0 1 1	0 1 1 0	1 0 1 0	0 1 0 1
1 1 0 0	1 0 0 1	0 1 0 1	0 1 0 1
0 0 1 1	0 1 1 0	0 1 0 1	1 0 1 0

４．単位面を組み合わせて４層の分解図を作る

上の８つの単位面から任意に４面を選んで，二進法分解図の４層と見立てる。そして次の式によって合成して，目的方陣の各解を構成する。はたして出来るだろうか？

Ｖn＝Ａn*8＋Ｂn*4＋Ｃn*2＋Ｄn*1＋１ (n=1, 2, 3, ..., 16)

プログラムを組んでコンピュータに調べさせたところ，全部で 4096 個作った中に 48 個の正解が含まれていた。
最初は，{1/, 1/, 1/, 1/}, {1/, 1/, 1/, 2/}, {1/, 1/, 1/, 3/}, {1/, 1/, 1/, 4/}, ..., {1/, 1/, 2/, 1/}, {1/, 1/, 2/, 2/}, ..., {8/, 1/, 1/, 1/}, {8/, 1/, 1/, 2/}, ..., {8/, 8/, 8/, 7/}, {8/, 8/, 8/, 8/} と４層を組んで調べたから，全部で８×８×８×８＝ 4096 個も作って調べてしまった。
その中に求めている正解が 全48個含まれていた。狙いは悪くないようだが，しかし，4096個中正解が 48 個しか無いのである。後は，反転・回転形が 464 個。その他は誤答だった。まだまだ絞り方が甘いという気がする。手作業でやった場合，何時間もかかる。とても他人には選別を頼めない。
途中どんな誤答が出て来たのか。無理にリスト出力させた下の例を見てほしい。

＊＊　誤った答の例　＊＊

```
   1/   1/   1/   1/       1[ ??]     1/   1/   1/   2/       2[ ??]
 0101 0101 0101 0101   1 16  1 16   0101 0101 0101 0101   1 16  1 16
 1010 1010 1010 1010  16  1 16  1   1010 1010 1010 0101  15  2 15  2
 1010 1010 1010 1010  16  1 16  1   1010 1010 1010 1010  16  1 16  1
 0101 0101 0101 0101   1 16  1 16   0101 0101 0101 1010   2 15  2 15
   1/   1/   2/   3/      11[ ??]     1/   1/   2/   4/      12[ ??]
 0101 0101 0101 0110   1 16  2 15   0101 0101 0101 0011   1 15  2 16
 1010 1010 0101 1001  14  3 13  4   1010 1010 0101 1100  14  4 13  3
 1010 1010 1010 0110  15  2 16  1   1010 1010 1010 0011  15  1 16  2
 0101 0101 1010 1001   4 13  3 14   0101 0101 1010 1100   4 14  3 13

1/2/3/5/[ 13] 1/2/3/6/[ ??] 1/2/3/7/[ ??] 1/2/3/8/[ ??] 1/2/4/1/[ ??] 1/2/4/2/[ ??]
  2 16  3 13    2 15  3 14    2 15  4 13    2 15  4 13    1 14  3 16    1 14  3 16
 11  5 10  8   11  6 10  7   12  5 10  7   11  6  9  8   12  7 10  5   11  8  9  6
 14  4 15  1   14  3 15  2   13  4 15  2   13  4 15  2   14  1 16  3   14  1 16  3
  7  9  6 12    7 10  6 11    7 10  5 12    8  9  6 11    7 12  5 10    8 11  6  9
1/3/4/5/[ ??] 1/3/4/6/[ ??] 1/3/4/7/[ ??] 1/3/4/8/[ ??] 1/3/5/1/[ ??] 1/3/5/2/[ 27]
  2 14  7 11    2 13  7 12    2 13  8 11    2 13  8 11    3 16  5 10    3 16  5 10
 15  3 10  6   15  4 10  5   16  3 10  5   15  4  9  6   14  1 12  7   13  2 11  8
 10  6 15  3   10  5 15  4    9  6 15  4    9  6 15  4   12  7 14  1   12  7 14  1
  7 11  2 14    7 12  2 13    7 12  1 14    8 11  2 13    5 10  3 16    6  9  4 15

1/2/4/6/[ 19] 2/1/4/6/[ 20] 1/3/4/7/[ ??] 2/3/4/8/[ 21] 3/1/4/7/[ ??] 3/2/4/8/[ 22]
  2 13  3 16    2 13  3 16    2 13  8 11    2 13  8 11    2 13 12  7    2 13 12  7
 11  8 10  5    7 12  6  9   16  3 10  5    7 12  1 14   16  3  6  9   11  8  1 14
 14  1 15  4   14  1 15  4    9  6 15  4    9  6 15  4    5 10 15  4    5 10 15  4
  7 12  6  9   11  8 10  5    7 12  1 14   16  3 10  5   11  8  1 14   16  3  6  9

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

上図で[??]とあるのが，正解にならなかった例で，上段の解には良く見ると同じ数字が何回も登場している。無い数字もある。これらは，すべて特定の数値の重複・欠落を禁じる「完全オイラー方陣」の定義【条件３】に違反している。とてもこれを認めることはできない。

最後の２段は，誤答は誤答でも，正解の反転・回転形だ。誤ってはいないのだが，２個目，３個目としてゾロゾロ出て来られては困る解の形である。

プログラム中にチェック - プロセスを置いて，コンピューターに選ばせるのはどうか？

テストして定義【条件３】に違反するものを除き，さらに次の不等式を課して反転・回転形の出力を抑える。

```
n１＜n16; n１＜n４; n１＜n13; n２＞n５;
```

試しに次のようなプログラムを加えて，実行したら，正解だけが出て来た。

```
/**/
/* Procedure: Combine and Compose */
void cmbcmp(){
 short n,d,fc;
  /*. . . . .*/
        for(n=1;n<17;n++){uflg[n]=0;}
          for(n=1;n<17;n++){
            d=tlu[u1][n]*8+tlu[u2][n]*4+tlu[u3][n]*2+tlu[u4][n]+1;
            nm[n]=d; uflg[d]++;
          }
```

```
          fc=0;
          for(n=1;n<17;n++){if(uflg[n]==1){fc++;}else{break;}}
          if(fc==16){
            if((nm[1]<nm[16])&&(nm[1]<nm[4])&&(nm[1]<nm[13])&&(nm[2]>nm[5])){
              nextproc();}
          }
        }
  /*. . . . .*/
}
/**/
```

＊　再構成した汎四方陣の標準解のリスト：使用単位面//// 解番号:新#[旧] ＊

```
   1/   2/   3/   4/       #1[ 1]     1/   2/   3/   5/       #2[13]
0101 0101 0110 0011    1 15  4 14   0101 0101 0110 1100    2 16  3 13
1010 0101 1001 1100   12  6  9  7   1010 0101 1001 0011   11  5 10  8
1010 1010 0110 0011   13  3 16  2   1010 1010 0110 1100   14  4 15  1
0101 1010 1001 1100    8 10  5 11   0101 1010 1001 0011    7  9  6 12
   1/   2/   4/   3/       #3[ 7]     1/   2/   4/   6/       #4[19]
0101 0101 0011 0110    1 14  4 15   0101 0101 0011 1001    2 13  3 16
1010 0101 1100 1001   12  7  9  6   1010 0101 1100 0110   11  8 10  5
1010 1010 0011 0110   13  2 16  3   1010 1010 0011 1001   14  1 15  4
0101 1010 1100 1001    8 11  5 10   0101 1010 1100 0110    7 12  6  9
   1/   2/   5/   3/       #5[25]     1/   2/   5/   6/       #6[35]
0101 0101 1100 0110    3 16  2 13   0101 0101 1100 1001    4 15  1 14
1010 0101 0011 1001   10  5 11  8   1010 0101 0011 0110    9  6 12  7
1010 1010 1100 0110   15  4 14  1   1010 1010 1100 1001   16  3 13  2
0101 1010 0011 1001    6  9  7 12   0101 1010 0011 0110    5 10  8 11
   1/   2/   6/   4/       #7[31]     1/   2/   6/   5/       #8[41]
0101 0101 1001 0011    3 13  2 16   0101 0101 1001 1100    4 14  1 15
1010 0101 0110 1100   10  8 11  5   1010 0101 0110 0011    9  7 12  6
1010 1010 1001 0011   15  1 14  4   1010 1010 1001 1100   16  2 13  3
0101 1010 0110 1100    6 12  7  9   0101 1010 0110 0011    5 11  8 10
   1/   3/   2/   4/       #9[ 3]     1/   3/   2/   5/       #10[15]
0101 0110 0101 0011    1 15  6 12   0101 0110 0101 1100    2 16  5 11
1010 1001 0101 1100   14  4  9  7   1010 1001 0101 0011   13  3 10  8
1010 0110 1010 0011   11  5 16  2   1010 0110 1010 1100   12  6 15  1
0101 1001 1010 1100    8 10  3 13   0101 1001 1010 0011    7  9  4 14
   1/   3/   5/   2/       #11[27]    1/   3/   5/   7/       #12[37]
0101 0110 1100 0101    3 16  5 10   0101 0110 1100 1010    4 15  6  9
1010 1001 0011 0101   13  2 11  8   1010 1001 0011 1010   14  1 12  7
1010 0110 1100 1010   12  7 14  1   1010 0110 1100 0101   11  8 13  2
0101 1001 0011 1010    6  9  4 15   0101 1001 0011 0101    5 10  3 16

 2/1/3/4/[ 2]  2/1/3/5/[14]  2/1/4/3/[ 8]  2/1/4/6/[20]  2/1/5/3/[26]  2/1/5/6/[36]
  1 15  4 14    2 16  3 13    1 14  4 15    2 13  3 16    3 16  2 13    4 15  1 14
  8 10  5 11    7  9  6 12    8 11  5 10    7 12  6  9    6  9  7 12    5 10  8 11
 13  3 16  2   14  4 15  1   13  2 16  3   14  1 15  4   15  4 14  1   16  3 13  2
 12  6  9  7   11  5 10  8   12  7  9  6   11  8 10  5   10  5 11  8    9  6 12  7
 2/1/6/4/[32]  2/1/6/5/[42]  2/3/1/4/[ 4]  2/3/1/5/[16]  2/3/4/1/[ 9]  2/3/4/8/[21]
  3 13  2 16    4 14  1 15    1 15  6 12    2 16  5 11    1 14  7 12    2 13  8 11
  6 12  7  9    5 11  8 10    8 10  3 13    7  9  4 14    8 11  2 13    7 12  1 14
 15  1 14  4   16  2 13  3   11  5 16  2   12  6 15  1   10  5 16  3    9  6 15  4
 10  8 11  5    9  7 12  6   14  4  9  7   13  3 10  8   15  4  9  6   16  3 10  5
```

```
 2/3/5/1/[28]   2/3/5/8/[38]   2/3/8/4/[33]   2/3/8/5/[43]   2/4/1/3/[11]   2/4/1/6/[23]
  3 16  5 10     4 15  6  9     3 13  8 10     4 14  7  9     1 12  6 15     2 11  5 16
  6  9  4 15     5 10  3 16     6 12  1 15     5 11  2 16     8 13  3 10     7 14  4  9
 12  7 14  1    11  8 13  2     9  7 14  4    10  8 13  3    11  2 16  5    12  1 15  6
 13  2 11  8    14  1 12  7    16  2 11  5    15  1 12  6    14  7  9  4    13  8 10  3
 2/4/3/1/[12]   2/4/3/8/[24]   2/5/1/3/[45]   2/5/1/6/[47]   2/5/3/1/[46]   2/5/3/8/[48]
  1 12  7 14     2 11  8 13     5 16  2 11     6 15  1 12     5 16  3 10     6 15  4  9
  8 13  2 11     7 14  1 12     4  9  7 14     3 10  8 13     4  9  6 15     3 10  5 16
 10  3 16  5     9  4 15  6    15  6 12  1    16  5 11  2    14  7 12  1    13  8 11  2
 15  6  9  4    16  5 10  3    10  3 13  8     9  4 14  7    11  2 13  8    12  1 14  7
 3/1/2/4/[ 5]   3/1/2/5/[17]   3/1/5/2/[29]   3/1/5/7/[39]   3/2/1/4/[ 6]   3/2/1/5/[18]
  1 15 10  8     2 16  9  7     3 16  9  6     4 15 10  5     1 15 10  8     2 16  9  7
 14  4  5 11    13  3  6 12    13  2  7 12    14  1  8 11    12  6  3 13    11  5  4 14
  7  9 16  2     8 10 15  1     8 11 14  1     7 12 13  2     7  9 16  2     8 10 15  1
 12  6  3 13    11  5  4 14    10  5  4 15     9  6  3 16    14  4  5 11    13  3  6 12
 3/2/4/1/[10]   3/2/4/8/[22]   3/2/5/1/[30]   3/2/5/8/[40]   3/2/8/4/[34]   3/2/8/5/[44]
  1 14 11  8     2 13 12  7     3 16  9  6     4 15 10  5     3 13 12  6     4 14 11  5
 12  7  2 13    11  8  1 14    10  5  4 15     9  6  3 16    10  8  1 15     9  7  2 16
  6  9 16  3     5 10 15  4     8 11 14  1     7 12 13  2     5 11 14  4     6 12 13  3
 15  4  5 10    16  3  6  9    13  2  7 12    14  1  8 11    16  2  7  9    15  1  8 10
[Count of Solutions = 48]
```

＊　新旧のリストで解の対応関係をモニターした結果　＊

```
??: 0
 1: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
25: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   [OK!]
```

めでたし，めでたし。

しかし，これで終わりにするわけには行かない。正解が全部出て来たからと言って，まだ作業に無駄が多過ぎる。まだ「完全オイラー四方陣」の構造が良く見えて来ていない。

５．誤答を作らない工夫

途中で誤答が何故出て来るのか，それを分析しよう。誤答の例を示したリストに戻る。

（１）１つの解の中で同じ単位面を２度も３度も繰り返し使用してはいけないようだ。

つまり，８×８×８×８ではなくて，８個から４個を取る「順列」８×７×６×５でなくてはいけないようだ。そうすると，調べる場合の数が，4095 個から 1680 個に減る。

だが，これでもまだ無駄が多いように思われる。

（２）単位面の 1/面と 8/面は，どうも両立できないようだ。2/面と 7/面も，両立できない。何故か？

下図の 1/面と 8/面は，全ての数値が‘０’は‘１’に，‘１’は‘０’に入れ替わっている。お互いが相手の「補数面」になる関係だ。そういう例は，他にもあるか？

```
 T0:  1/       8/    T1:  2/       7/    T2:  3/       6/
 0 1 0 1   1 0 1 0   0 1 0 1   1 0 1 0   0 1 1 0   1 0 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 1 0   0 1 0 1   1 0 1 0   0 1 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   1 0 0 1   0 1 1 0
 T3:  4/       5/
 0 0 1 1   1 1 0 0
 1 1 0 0   0 0 1 1     もし[An + Bn == 1 (n=1,2,3,4, ... , 15,16)]
 0 0 1 1   1 1 0 0     ならば,  [単位面(A/, B/) は '補数面'である。]
 1 1 0 0   0 0 1 1
```

2/面と 7/面，3/面と 6/面，4/面と 5/面とが，お互いに「補数面」になっている。
これらの組合わせがあると，必ず誤答になる。重複や欠落が起きてしまう。
つまり，単位面を {1/, 8/}, {2/, 7/}, {3/, 6/}, {4/, 5/} と４つにグループ分けすると，１つの解の中では各組の２つの内のいずれか一方しか使えないことになる。
そして，目的の４層を作るには，この４組から１つづつ代表を出してもらえばよいことになる。その時の「場合の数」は，２×２×２×２＝ 16 通り。

後は，その４代表の並び方を考えてやれば良い。これは，４個から４個を取る順列で，
４×３×２×１＝ 24 通りだ。
全部で，16×24＝384 通りを調べれば良いことになる。随分と少なくなった。

では，リスト出力用の不等式を前もって課することができるか？　次に考えよう。
二進法分解図では，数値の大小判断に関しては，最上層が常に優先権を握っている。合成された解の n1 をいつも左上に持って来るには，最小の値を与えるために最上層に a1 ＝０のある面を選ぶしか無い。つまり，最上層は，{1/, 2/, 3/, 4/} から選ばれる。
他方，n２＞n５を満たすには，4/面を最上層に持って来ることはできない。4/面だと，必ず n２＜n５となってしまうから。
つまり，最上層は，{1/, 2/, 3/} の中から選ぶしかない。
２層，３層と順に調べて行くと，どうしても場合分けが複雑になってしまう。おぼえにくい。せっかく自分で見い出した規則を適用する時に，とかく混乱しがちだ。‥‥
ここは，いっそアフター・チェックに任せた方が簡単かもしれない。

不等式によって選ばれる解は，全体の８分の１だから，384÷８＝48 個が出力される。
これで総数は合うが，解の内容が問題だ。１つ１つを前に得た解のリストと参照・比較して，重複や欠落が無いか，一応これもアフター・チェックしなければならない。

６．四次の「完全オイラー方陣」構成法

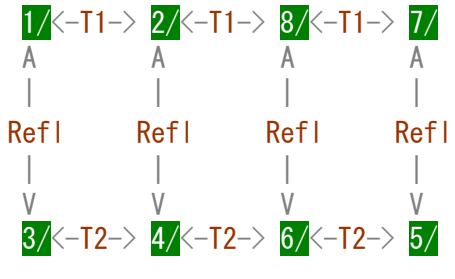
以上の考察から，新構成法は次のようにまとめられる。

```
Step 1/    Step 2/    Step 3/    Step 4/
0 - - -    0 1 - -    0 1 - 1    0 1 0 1
- 0 - -    1 0 - -    1 0 - -    1 0 1 0
- - 1 -    - - 1 0    - 0 1 0    1 0 1 0
- - - 1    - - 0 1    - - 0 1    0 1 0 1
```

（１）単位面を上図のように１つ手作りし，それを変換して他の７つの単位面を構成する。例えば，下図のように列の移動・対称変換・行の移動を組み合わせて作る手がある。

＊　変換によって８つの単位面を作る　＊

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



{1/, 8/}, {2/, 7/}, {3/, 6/}, {4/, 5/}の４ペアがお互いに「補数面」になっていることを利用する変換法も可能だ。
（２）単位面を４つ選んで適切に組み合わせ，二進法分解図の４層と成す。

この時先ず {1/, 8/}, {2/, 7/}, {3/, 6/}, {4/, 5/} の各ペアから代表として１つの単位面を選び,それをいろいろな順序で並べ換えて,二進法分解図の４層とする。

（２×２×２×２）×（４×３×２×１）＝ 16×24 ＝ 384 通りの組み合せが出来る。

もし，第１層を {1/, 2/, 3/} の３つに絞れば，全体を 144 通りに絞れる。

（３）構成した二進法分解図の４層から合成して，各解を作って行く。

Ｖn＝Ａn*8＋Ｂn*4＋Ｃn*2＋Ｄn*1＋１ (n=1, 2, 3, ..., 16)

（４）その後で，不等式の試験を課し，標準解のみ 48 解を出力する。

ｎ１<n16; ｎ１<ｎ４; ｎ１<n13; ｎ２>ｎ５;

７．完全オイラー四方陣を再構成した結果

以上の手順に従って完全四方陣を再構成した結果を以下のリストで示す。定義【条件３】のチェックは，入れても入れなくても全く同じ結果を示した。ということは，事前に充分に絞り切ることができたことを意味する。

＊　二進法で再構成した完全四方陣の標準解のリスト：使用単位面//// 解番号旧[新] ＊

```
  1/   2/   3/   4/        1[ 1]    2/   1/   3/   4/        2[13]
0101 0101 0110 0011   1 15  4 14   0101 0101 0110 0011   1 15  4 14
1010 0101 1001 1100  12  6  9  7   0101 1010 1001 1100   8 10  5 11
1010 1010 0110 0011  13  3 16  2   1010 1010 0110 0011  13  3 16  2
0101 1010 1001 1100   8 10  5 11   1010 0101 1001 1100  12  6  9  7
  1/   3/   2/   4/        3[ 9]    2/   3/   1/   4/        4[21]
0101 0110 0101 0011   1 15  6 12   0101 0110 0101 0011   1 15  6 12
1010 1001 0101 1100  14  4  9  7   0101 1001 1010 1100   8 10  3 13
1010 0110 1010 0011  11  5 16  2   1010 0110 1010 0011  11  5 16  2
0101 1001 1010 1100   8 10  3 13   1010 1001 0101 1100  14  4  9  7
  3/   1/   2/   4/        5[37]    3/   2/   1/   4/        6[41]
0110 0101 0101 0011   1 15 10  8   0110 0101 0101 0011   1 15 10  8
1001 1010 0101 1100  14  4  5 11   1001 0101 1010 1100  12  6  3 13
0110 1010 1010 0011   7  9 16  2   0110 1010 1010 0011   7  9 16  2
1001 0101 1010 1100  12  6  3 13   1001 1010 0101 1100  14  4  5 11
  1/   2/   4/   3/        7[ 3]    2/   1/   4/   3/        8[15]
0101 0101 0011 0110   1 14  4 15   0101 0101 0011 0110   1 14  4 15
1010 0101 1100 1001  12  7  9  6   0101 1010 1100 1001   8 11  5 10
1010 1010 0011 0110  13  2 16  3   1010 1010 0011 0110  13  2 16  3
0101 1010 1100 1001   8 11  5 10   1010 0101 1100 1001  12  7  9  6
  2/   3/   4/   1/        9[23]    3/   2/   4/   1/       10[43]
0101 0110 0011 0101   1 14  7 12   0110 0101 0011 0101   1 14 11  8
0101 1001 1100 1010   8 11  2 13   1001 0101 1100 1010  12  7  2 13
1010 0110 0011 1010  10  5 16  3   0110 1010 0011 1010   6  9 16  3
1010 1001 1100 0101  15  4  9  6   1001 1010 1100 0101  15  4  5 10
  2/   4/   1/   3/       11[29]    2/   4/   3/   1/       12[31]
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
```

＊ 48 Standard Solutions(to be continued): Used Units////[List Number:New] ＊

```
 1/2/3/5/[ 2]  2/1/3/5/[14]  1/3/2/5/[10]  2/3/1/5/[22]  3/1/2/5/[38]  3/2/1/5/[42]
  2 16  3 13    2 16  3 13    2 16  5 11    2 16  5 11    2 16  9  7    2 16  9  7
 11  5 10  8    7  9  6 12   13  3 10  8    7  9  4 14   13  3  6 12   11  5  4 14
 14  4 15  1   14  4 15  1   12  6 15  1   12  6 15  1    8 10 15  1    8 10 15  1
  7  9  6 12   11  5 10  8    7  9  4 14   13  3 10  8   11  5  4 14   13  3  6 12
```

```
1/2/4/6/[ 4]  2/1/4/6/[16]  2/3/4/8/[24]  3/2/4/8/[44]  2/4/1/6/[30]  2/4/3/8/[32]
 2 13  3 16    2 13  3 16    2 13  8 11    2 13 12  7    2 11  5 16    2 11  8 13
11  8 10  5    7 12  6  9    7 12  1 14   11  8  1 14    7 14  4  9    7 14  1 12
14  1 15  4   14  1 15  4    9  6 15  4    5 10 15  4   12  1 15  6    9  4 15  6
 7 12  6  9   11  8 10  5   16  3 10  5   16  3  6  9   13  8 10  3   16  5 10  3
1/2/5/3/[ 5]  2/1/5/3/[17]  1/3/5/2/[11]  2/3/5/1/[25]  3/1/5/2/[39]  3/2/5/1/[45]
 3 16  2 13    3 16  2 13    3 16  5 10    3 16  5 10    3 16  9  6    3 16  9  6
10  5 11  8    6  9  7 12   13  2 11  8    6  9  4 15   13  2  7 12   10  5  4 15
15  4 14  1   15  4 14  1   12  7 14  1   12  7 14  1    8 11 14  1    8 11 14  1
 6  9  7 12   10  5 11  8    6  9  4 15   13  2 11  8   10  5  4 15   13  2  7 12
1/2/6/4/[ 7]  2/1/6/4/[19]  2/3/8/4/[27]  3/2/8/4/[47]  1/2/5/6/[ 6]  2/1/5/6/[18]
 3 13  2 16    3 13  2 16    3 13  8 10    3 13 12  6    4 15  1 14    4 15  1 14
10  8 11  5    6 12  7  9    6 12  1 15   10  8  1 15    9  6 12  7    5 10  8 11
15  1 14  4   15  1 14  4    9  7 14  4    5 11 14  4   16  3 13  2   16  3 13  2
 6 12  7  9   10  8 11  5   16  2 11  5   16  2  7  9    5 10  8 11    9  6 12  7
1/3/5/7/[12]  2/3/5/8/[26]  3/1/5/7/[40]  3/2/5/8/[46]  1/2/6/5/[ 8]  2/1/6/5/[20]
 4 15  6  9    4 15  6  9    4 15 10  5    4 15 10  5    4 14  1 15    4 14  1 15
14  1 12  7    5 10  3 16   14  1  8 11    9  6  3 16    9  7 12  6    5 11  8 10
11  8 13  2   11  8 13  2    7 12 13  2    7 12 13  2   16  2 13  3   16  2 13  3
 5 10  3 16   14  1 12  7    9  6  3 16   14  1  8 11    5 11  8 10    9  7 12  6
2/3/8/5/[28]  3/2/8/5/[48]  2/5/1/3/[33]  2/5/3/1/[35]  2/5/1/6/[34]  2/5/3/8/[36]
 4 14  7  9    4 14 11  5    5 16  2 11    5 16  3 10    6 15  1 12    6 15  4  9
 5 11  2 16    9  7  2 16    4  9  7 14    4  9  6 15    3 10  8 13    3 10  5 16
10  8 13  3    6 12 13  3   15  6 12  1   14  7 12  1   16  5 11  2   13  8 11  2
15  1 12  6   15  1  8 10   10  3 13  8   11  2 13  8    9  4 14  7   12  1 14  7
[Count of Solutions = 48]
```

＊　新旧のリストで解の対応関係をモニターした結果　＊

```
 ??: 0
  1: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 25: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   [OK!]
```

このリストは，前のリストと解の出力順序が違う。古いリストと同じ順序にしてある。その方が見やすいと思うからそうしたのだが，結果的にソートしたのと同じ結果になった。

各解の右肩にある番号は，新しいリストの解番号である。対応関係をモニターした結果を見ると，全 48 解を重複なく欠落なく再現・構成できていることがわかる。

二進法分解図と「ギリシャ・ラテン方陣」構成法によって完全オイラー四方陣の解を全数作る試みは，成功したと判断して良い。

８．この実験の意味

この実験の成功は，いったい何を意味するのだろうか？

（１）汎四方陣（或いは，完全四方陣，正方連結完全四方陣）が全て，二進法による「完全オイラー方陣」であることが，証明された。表からも裏からもそうであるから，集合論的に「同値」である。

（２）標準解の総数が何故 48 個なのかの説明もできた。

（２×２×２×２）×（４×３×２×１）÷８＝ 16×24÷８＝ 48

（３）単位面８個は，本質的には同一の大基本形の反転・回転形になっている。このことはいかにリソースが稀少であるかを示している。汎四方陣の「稀少価値」がこれでわかる。

（４）しかし，このままでは，汎四方陣の３つの基本形については，何も言えていない。

48 ＝３×16 の説明ができない。

（５）二進法でプログラムを組むのは，煩雑で手間がかかる。できれば，２度とやりたくな

い気がする（「同値」の正方連結汎魔方陣ならば，通常の方法でプログラムを組むのが，簡単で実行速度も速い。特に高次では，そちらを選びたい気がする）。
　しかし，当分の間は，この方法を試し続けなければならない。正方連結型の汎魔方陣だけが「完全オイラー方陣」であると言う予想が，高次ではまだ証明されていない。

次の課題は，八次で「全ての正方連結汎魔方陣が二進法による完全オイラー方陣である」ことを実証すること。そして，「二進法による完全オイラー方陣は，全てが正方連結汎魔方陣である」という逆の命題も実証しなければならない。前者は，実例がそれを証明しているが，後者の予想については，正直まだ何も実証されていない。

（初稿のまま: on July 11, 2003; by 摂田 寛二(Kanji Setsuda)）

９．改訂追記事項

その後の汎八方陣での研究成果を反映させて，次の２点を改訂したい。
（１）単位面８個の構成について，（２）無駄を省くチェック・プログラムについて。

単位面が８つ必要だと言うことは，実は，他の研究成果から借りて来た知識だった。これを自前で知るようにしたい。そのために単位面を作るプログラムを新たに採用したい。
完全オイラー方陣の定義に従い，‘0’と‘1’だけのラテン方陣を直に作るのである。単位面として二進法分解図にすぐに利用できるように。
それ専用のプログラムを組む。その解として，８つの単位面を出力するようにしたい。
どのポジションも，0 か 1 かの二つの値しか取らない。各行・各列・各汎対角線は {0, 0, 1, 1}, {0, 1, 0, 1}, {0, 1, 1, 0}, ... のように和が２になるような構成にする。
今は条件をそれだけに限定して，４×４の単位面を構築する。
詳細は，後で示すプログラム・リストを見ていただきたい。

＊　二進法分解図のための単位面　＊

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

[ラテン・ユニットの個数 = 8]

実験の結果，上図のような８つの単位面が解として確かに出て来た。これを組み合わせて，４層の二進法分解図とする。そして，目的方陣を合成する。
その手順は，既に説明した通りで良いのだが，途中の無駄を省くために，同一の単位面を重複利用せず，「補数面」の組み合わせも避けるようにしたい。前回はそれらの手順を２つ加えたのだったが，今回はこの手順を１つで済ますようにしたい。
そのために「参照表」を作る。単位面どうしを１個ずつ相互に比較して，対応する 16 個のポジションで値が一致する場合の数をカウントする。結果を次のようなテーブルに残す。
下表で，(1, 1), (2, 2), (3, 3), (4, 4), ... の値が 16 になるのは，同一の２面を較べれば常に全部が同じになるので，当然である。
(1, 8), (2, 7), (3, 6), (4, 5), (5, 4), ... の値が０になるのは，これらの組み合せが「補数面」であることを示している。一致するものが無いと言うことは，全ての値が，０は１に，１は０に「補数変換」されているからである。
他の組み合せは，全部８を示す。半数しか一致していないことを意味する。

[任意の２面の相似度を計測した参照表]

*	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
```

このテーブルを後で利用する。ペアの値が８である場合だけを選んで，組み合わせる。

そして，以下のリストが，わが最新**プログラム**の全貌である。全て“Ｃ言語”で書いたが，中身は，ほとんど「ミニ‘PASCAL’」である。なるべく簡潔に書いたつもりだ。

なお，ここでは「正方連結型汎対角線四方陣」を求めるプログラムを紹介している。定義条件の数の一番多いタイプだが，出来上がる解集合は今までのものと全く同じだ。

```
/** 'Composite & Pan-diagonal' Magic Squares of Order 4: Composing 'Complete Euler **/
/**  Squares' of Order 4 by the "New Euler's Method" with Binary Number System  **/
/** 'CES44PD.c': Dictated by Kanji Setsuda on Oct.22,'04; Sep.15,'06; **/
/**  Aug.10,'10; and Mar. 9, 2011; with MacOSX 10.5.8 and Xcode 3.1.2 **/
/**/
#include <stdio.h>
/**/
short lcnt, cnt;
short LSM;
short nm[17], uflg[17];
short tlu[9][17];
short mtc[9][9];
short ta[49][22];
short d[2][22];
short lnsm[2][9], pdsm[2][9];
/**/
short bcf[9][2], pdf[9][2], ccf[17][2];
short u1, u2, u3, u4;
/**/
 void stp01(void), stp02(void), stp03(void), stp04(void);
 void stp05(void), stp06(void), stp07(void), stp08(void);
 void stp09(void), stp10(void), stp11(void), stp12(void);
 void stp13(void), stp14(void), stp15(void), stp16(void);
 void lurecord(void);
 void prlunit(void);
 void cmbcmp(void), recordsol(void);
 void sortsol(void), exc(short x);
 void chksum(void);
 void prpat2(void);
 void pr2sol(short x), pr6sol(short x, short y);
/**/
int main(){
 short m,n;
  printf("\n** Making 'Composite & Pan-diagonal' Magic Squares of Order 4 **\n");
  printf("  ** by the 'New Euler's Method' with Binary Number System **\n");
  for(n=0;n<17;n++){nm[n]=0;}
  for(m=0;m<9;m++){for(n=0;n<2;n++){bcf[m][n]=0; pdf[m][n]=0;}}
  for(m=0;m<17;m++){for(n=0;n<2;n++){ccf[m][n]=0;}}
  lcnt=0;
  stp01();        // Make Latin Units
  printf(" [List of Latin Units with Decompositions by Binary System]\n");
  prlunit();      // Print Latin Units
  cnt=0;
  cmbcmp();       // Combine and Compose
  sortsol();      // Sort the Solutions
  printf("\n ** 'Composite & Pan-diagonal' Magic Squares of Order 4: **\n");
  printf(" ** Composing the 'Complete Euler Squares' of Order 4 by **\n");
  printf(" **  the 'New Euler's Method' with Binary Number System  **\n");
  printf(" [List of Standard Solutions(Sorted): Used Units//Sol.Number#|Sum-Checks/]\n");
  pr2sol(12);
```

```
  pr6sol(13,48);
  printf(" [Solution Counts = %d]\n",cnt);
  printf("\n [OK!]\n");
  printf("** Calculated and Listed by Kanji Setsuda on Mar. 8, 2011 with MAC Xcode 3 **\n");
 return 0;
}
/**/
/*
** Basic Forms and Basic Conditions for the 'Composite **
** & Pan-diagonal' Magic Squares of Order 4: **
 14 15 16 13 14 15 16 13 14 15          Mode1/C1
        .--.--.--.--.                .--.--.--.--.
  2  3  4| 1| 2| 3| 4| 1  2  3     | 1|15| 4|14|
        |--+--+--+--|              |--+--+--+--|
  6  7  8| 5| 6| 7| 8| 5  6  7     |12| 6| 9| 7|
        |--+--+--+--|              |--+--+--+--|
 10 11 12| 9|10|11|12| 9 10 11     |13| 3|16| 2|
        |--+--+--+--|              |--+--+--+--|
 14 15 16|13|14|15|16|13 14 15     | 8|10| 5|11|
        '--'--'--'--'              '--'--'--'--'
  2  3  4  1  2  3  4  1  2  3
** Basic Conditions for Complete PMS44: **
  n1+n2+n3+n4=S     ... rw1; | n1+n5+n9+n13=S    ... cl1;
  n5+n6+n7+n8=S     ... rw2; | n2+n6+n10+n14=S   ... cl2;
  n9+n10+n11+n12=S  ... rw3; | n3+n7+n11+n15=S   ... cl3;
  n13+n14+n15+n16=S ... rw4; | n4+n8+n12+n16=S   ... cl4;
** 'Pan-diagonal' Conditions: **
  n1+n6+n11+n16=S   ... pd1; | n1+n8+n11+n14=S   ... pb1;
  n2+n7+n12+n13=S   ... pd2; | n2+n5+n12+n15=S   ... pb2;
  n3+n8+n9+n14=S    ... pd3; | n3+n6+n9+n16=S    ... pb3;
  n4+n5+n10+n15=S   ... pd4; | n4+n7+n10+n13=S   ... pb4;
** 'Composite' Conditions: **
  n1+n2+n5+n6=S     ...cc01; | n2+n3+n6+n7=S     ...cc02;
  n3+n4+n7+n8=S     ...cc03; | n4+n1+n8+n5=S     ...cc04;
  n5+n6+n9+n10=S    ...cc05; | n6+n7+n10+n11=S   ...cc06;
  n7+n8+n11+n12=S   ...cc07; | n8+n5+n12+n9=S    ...cc08;
  n9+n10+n13+n14=S  ...cc09; | n10+n11+n14+n15=S ...cc10;
  n11+n12+n15+n16=S ...cc11; | n12+n9+n16+n13=S  ...cc12;
  n13+n14+n1+n2=S   ...cc13; | n14+n15+n2+n3=S   ...cc14;
  n15+n16+n3+n4=S   ...cc15; | n16+n13+n4+n1=S   ...cc16;
** 'Complete' Conditions: **
  n1+n11=n2+n12=n3+n9=n4+n10=n5+n15=n6+n16=n7+n13=n8+n14=CC ... cc
** List-forming Inequality Conditions: **
  n1<n16, n1<n4, n1<n13, and n2>n5;
*/
/**/
/** Program Modules for Latin Units of 'Complete Euler Squares' **/
/** of Order 4 for the 'Composite & Pan-diagonal' Magic Squares **/
/**  Dictated by Kanji Setsuda on Mar. 8, '11 with MAC Xcode 3  **/
/**/
/* Level 1: */
/* Set n1 */
void stp01(){
 short a;
  for(a=0;a<2;a++){
    if((bcf[1][a]<2)&&(bcf[5][a]<2)&&(pdf[1][a]<2)&&(pdf[5][a]<2)){
      if((ccf[1][a]<2)&&(ccf[4][a]<2)&&(ccf[13][a]<2)&&(ccf[16][a]<2)){
        bcf[1][a]++; bcf[5][a]++; pdf[1][a]++; pdf[5][a]++;
        ccf[1][a]++; ccf[4][a]++; ccf[13][a]++; ccf[16][a]++;
        nm[1]=a; stp02();
        ccf[16][a]--; ccf[13][a]--; ccf[4][a]--; ccf[1][a]--;
        pdf[5][a]--; pdf[1][a]--; bcf[5][a]--; bcf[1][a]--;
    }}
  }
```

```
}
/**/
/* Set n11 */
void stp02(){
 short a;
  for(a=1;a>=0;a--){
    if((bcf[3][a]<2)&&(bcf[7][a]<2)&&(pdf[1][a]<2)&&(pdf[5][a]<2)){
      if((ccf[6][a]<2)&&(ccf[7][a]<2)&&(ccf[10][a]<2)&&(ccf[11][a]<2)){
        bcf[3][a]++; bcf[7][a]++; pdf[1][a]++; pdf[5][a]++;
        ccf[6][a]++; ccf[7][a]++; ccf[10][a]++; ccf[11][a]++;
        nm[11]=a; stp03();
        ccf[11][a]--; ccf[10][a]--; ccf[7][a]--; ccf[6][a]--;
        pdf[5][a]--; pdf[1][a]--; bcf[7][a]--; bcf[3][a]--;
    }}
  }
}
/**/
/* Set n2 */
void stp03(){
 short a;
  for(a=1;a>=0;a--){
    if((bcf[1][a]<2)&&(bcf[6][a]<2)&&(pdf[2][a]<2)&&(pdf[6][a]<2)){
      if((ccf[1][a]<2)&&(ccf[2][a]<2)&&(ccf[13][a]<2)&&(ccf[14][a]<2)){
        bcf[1][a]++; bcf[6][a]++; pdf[2][a]++; pdf[6][a]++;
        ccf[1][a]++; ccf[2][a]++; ccf[13][a]++; ccf[14][a]++;
        nm[2]=a; stp04();
        ccf[14][a]--; ccf[13][a]--; ccf[2][a]--; ccf[1][a]--;
        pdf[6][a]--; pdf[2][a]--; bcf[6][a]--; bcf[1][a]--;
    }}
  }
}
/**/
/* Set n12 */
void stp04(){
 short a;
  for(a=0;a<2;a++){
    if((bcf[3][a]<2)&&(bcf[8][a]<2)&&(pdf[2][a]<2)&&(pdf[6][a]<2)){
      if((ccf[7][a]<2)&&(ccf[8][a]<2)&&(ccf[11][a]<2)&&(ccf[12][a]<2)){
        bcf[3][a]++; bcf[8][a]++; pdf[2][a]++; pdf[6][a]++;
        ccf[7][a]++; ccf[8][a]++; ccf[11][a]++; ccf[12][a]++;
        nm[12]=a; stp05();
        ccf[12][a]--; ccf[11][a]--; ccf[8][a]--; ccf[7][a]--;
        pdf[6][a]--; pdf[2][a]--; bcf[8][a]--; bcf[3][a]--;
    }}
  }
}
/**/
/* Set n4 */
void stp05(){
 short a;
  for(a=1;a>=0;a--){
    if((bcf[1][a]<2)&&(bcf[8][a]<2)&&(pdf[4][a]<2)&&(pdf[8][a]<2)){
      if((ccf[3][a]<2)&&(ccf[4][a]<2)&&(ccf[15][a]<2)&&(ccf[16][a]<2)){
        bcf[1][a]++; bcf[8][a]++; pdf[4][a]++; pdf[8][a]++;
        ccf[3][a]++; ccf[4][a]++; ccf[15][a]++; ccf[16][a]++;
        nm[4]=a; stp06();
        ccf[16][a]--; ccf[15][a]--; ccf[4][a]--; ccf[3][a]--;
        pdf[8][a]--; pdf[4][a]--; bcf[8][a]--; bcf[1][a]--;
    }}
  }
}
/**/
/* Set n10 */
void stp06(){
```

```
 short a;
  for(a=0;a<2;a++){
    if((bcf[3][a]<2)&&(bcf[6][a]<2)&&(pdf[4][a]<2)&&(pdf[8][a]<2)){
      if((ccf[5][a]<2)&&(ccf[6][a]<2)&&(ccf[9][a]<2)&&(ccf[10][a]<2)){
        bcf[3][a]++; bcf[6][a]++; pdf[4][a]++; pdf[8][a]++;
        ccf[5][a]++; ccf[6][a]++; ccf[9][a]++; ccf[10][a]++;
        nm[10]=a; stp07();
        ccf[10][a]--; ccf[9][a]--; ccf[6][a]--; ccf[5][a]--;
        pdf[8][a]--; pdf[4][a]--; bcf[6][a]--; bcf[3][a]--;
    }}
  }
}
/**/
/* Set n3 */
void stp07(){
 short a;
  for(a=0;a<2;a++){
    if((bcf[1][a]<2)&&(bcf[7][a]<2)&&(pdf[3][a]<2)&&(pdf[7][a]<2)){
      if((ccf[2][a]<2)&&(ccf[3][a]<2)&&(ccf[14][a]<2)&&(ccf[15][a]<2)){
        bcf[1][a]++; bcf[7][a]++; pdf[3][a]++; pdf[7][a]++;
        ccf[2][a]++; ccf[3][a]++; ccf[14][a]++; ccf[15][a]++;
        nm[3]=a; stp08();
        ccf[15][a]--; ccf[14][a]--; ccf[3][a]--; ccf[2][a]--;
        pdf[7][a]--; pdf[3][a]--; bcf[7][a]--; bcf[1][a]--;
    }}
  }
}
/**/
/* Set n9 */
void stp08(){
 short a;
  for(a=1;a>=0;a--){
    if((bcf[3][a]<2)&&(bcf[5][a]<2)&&(pdf[3][a]<2)&&(pdf[7][a]<2)){
      if((ccf[5][a]<2)&&(ccf[8][a]<2)&&(ccf[9][a]<2)&&(ccf[12][a]<2)){
        bcf[3][a]++; bcf[5][a]++; pdf[3][a]++; pdf[7][a]++;
        ccf[5][a]++; ccf[8][a]++; ccf[9][a]++; ccf[12][a]++;
        nm[9]=a; stp09();
        ccf[12][a]--; ccf[9][a]--; ccf[8][a]--; ccf[5][a]--;
        pdf[7][a]--; pdf[3][a]--; bcf[5][a]--; bcf[3][a]--;
    }}
  }
}
/**/
/* Level 2: */
/* Set n5 */
void stp09(){
 short a;
  for(a=1;a>=0;a--){
    if((bcf[2][a]<2)&&(bcf[5][a]<2)&&(pdf[4][a]<2)&&(pdf[6][a]<2)){
      if((ccf[1][a]<2)&&(ccf[4][a]<2)&&(ccf[5][a]<2)&&(ccf[8][a]<2)){
        bcf[2][a]++; bcf[5][a]++; pdf[4][a]++; pdf[6][a]++;
        ccf[1][a]++; ccf[4][a]++; ccf[5][a]++; ccf[8][a]++;
        nm[5]=a; stp10();
        ccf[8][a]--; ccf[5][a]--; ccf[4][a]--; ccf[1][a]--;
        pdf[6][a]--; pdf[4][a]--; bcf[5][a]--; bcf[2][a]--;
    }}
  }
}
/**/
/* Set n15 */
void stp10(){
 short a;
  for(a=0;a<2;a++){
    if((bcf[4][a]<2)&&(bcf[7][a]<2)&&(pdf[4][a]<2)&&(pdf[6][a]<2)){
```

```
      if((ccf[10][a]<2)&&(ccf[11][a]<2)&&(ccf[14][a]<2)&&(ccf[15][a]<2)){
        bcf[4][a]++; bcf[7][a]++; pdf[4][a]++; pdf[6][a]++;
        ccf[10][a]++; ccf[11][a]++; ccf[14][a]++; ccf[15][a]++;
        nm[15]=a; stp11();
        ccf[15][a]--; ccf[14][a]--; ccf[11][a]--; ccf[10][a]--;
        pdf[6][a]--; pdf[4][a]--; bcf[7][a]--; bcf[4][a]--;
    }}
  }
}
/**/
/* Set n6 */
void stp11(){
 short a;
  for(a=0;a<2;a++){
    if((bcf[2][a]<2)&&(bcf[6][a]<2)&&(pdf[1][a]<2)&&(pdf[7][a]<2)){
      if((ccf[1][a]<2)&&(ccf[2][a]<2)&&(ccf[5][a]<2)&&(ccf[6][a]<2)){
        bcf[2][a]++; bcf[6][a]++; pdf[1][a]++; pdf[7][a]++;
        ccf[1][a]++; ccf[2][a]++; ccf[5][a]++; ccf[6][a]++;
        nm[6]=a; stp12();
        ccf[6][a]--; ccf[5][a]--; ccf[2][a]--; ccf[1][a]--;
        pdf[7][a]--; pdf[1][a]--; bcf[6][a]--; bcf[2][a]--;
    }}
  }
}
/**/
/* Set n16 */
void stp12(){
 short a;
  for(a=1;a>=0;a--){
    if((bcf[4][a]<2)&&(bcf[8][a]<2)&&(pdf[1][a]<2)&&(pdf[7][a]<2)){
      if((ccf[11][a]<2)&&(ccf[12][a]<2)&&(ccf[15][a]<2)&&(ccf[16][a]<2)){
        bcf[4][a]++; bcf[8][a]++; pdf[1][a]++; pdf[7][a]++;
        ccf[11][a]++; ccf[12][a]++; ccf[15][a]++; ccf[16][a]++;
        nm[16]=a; stp13();
        ccf[16][a]--; ccf[15][a]--; ccf[12][a]--; ccf[11][a]--;
        pdf[7][a]--; pdf[1][a]--; bcf[8][a]--; bcf[4][a]--;
    }}
  }
}
/**/
/* Set n7 */
void stp13(){
 short a;
  for(a=1;a>=0;a--){
    if((bcf[2][a]<2)&&(bcf[7][a]<2)&&(pdf[2][a]<2)&&(pdf[8][a]<2)){
      if((ccf[2][a]<2)&&(ccf[3][a]<2)&&(ccf[6][a]<2)&&(ccf[7][a]<2)){
        bcf[2][a]++; bcf[7][a]++; pdf[2][a]++; pdf[8][a]++;
        ccf[2][a]++; ccf[3][a]++; ccf[6][a]++; ccf[7][a]++;
        nm[7]=a; stp14();
        ccf[7][a]--; ccf[6][a]--; ccf[3][a]--; ccf[2][a]--;
        pdf[8][a]--; pdf[2][a]--; bcf[7][a]--; bcf[2][a]--;
    }}
  }
}
/**/
/* Set n13 */
void stp14(){
 short a;
  for(a=0;a<2;a++){
    if((bcf[4][a]<2)&&(bcf[5][a]<2)&&(pdf[2][a]<2)&&(pdf[8][a]<2)){
      if((ccf[9][a]<2)&&(ccf[12][a]<2)&&(ccf[13][a]<2)&&(ccf[16][a]<2)){
        bcf[4][a]++; bcf[5][a]++; pdf[2][a]++; pdf[8][a]++;
        ccf[9][a]++; ccf[12][a]++; ccf[13][a]++; ccf[16][a]++;
        nm[13]=a; stp15();
```

```
          ccf[16][a]--; ccf[13][a]--; ccf[12][a]--; ccf[9][a]--;
          pdf[8][a]--; pdf[2][a]--; bcf[5][a]--; bcf[4][a]--;
      }}
    }
}
/**/
/* Set n8 */
void stp15(){
 short a;
   for(a=0;a<2;a++){
     if((bcf[2][a]<2)&&(bcf[8][a]<2)&&(pdf[3][a]<2)&&(pdf[5][a]<2)){
       if((ccf[3][a]<2)&&(ccf[4][a]<2)&&(ccf[7][a]<2)&&(ccf[8][a]<2)){
         bcf[2][a]++; bcf[8][a]++; pdf[3][a]++; pdf[5][a]++;
         ccf[3][a]++; ccf[4][a]++; ccf[7][a]++; ccf[8][a]++;
         nm[8]=a; stp16();
         ccf[8][a]--; ccf[7][a]--; ccf[4][a]--; ccf[3][a]--;
         pdf[5][a]--; pdf[3][a]--; bcf[8][a]--; bcf[2][a]--;
     }}
   }
}
/**/
/* Set n14 */
void stp16(){
 short a;
   for(a=1;a>=0;a--){
     if((bcf[4][a]<2)&&(bcf[6][a]<2)&&(pdf[3][a]<2)&&(pdf[5][a]<2)){
       if((ccf[9][a]<2)&&(ccf[10][a]<2)&&(ccf[13][a]<2)&&(ccf[14][a]<2)){
         bcf[4][a]++; bcf[6][a]++; pdf[3][a]++; pdf[5][a]++;
         ccf[9][a]++; ccf[10][a]++; ccf[13][a]++; ccf[14][a]++;
         nm[14]=a; lurecord();
         ccf[14][a]--; ccf[13][a]--; ccf[10][a]--; ccf[9][a]--;
         pdf[5][a]--; pdf[3][a]--; bcf[6][a]--; bcf[4][a]--;
     }}
   }
}
/**/
/* Record the Answers */
void lurecord(){
 short n;
   tlu[lcnt][0]=lcnt+1;
   for(n=1;n<17;n++){tlu[lcnt][n]=nm[n];}
   lcnt++;
}
/**/
/* Make Matching Table and Print the Latin Units */
void prlunit(){
 short t,l,l4,m,n;
   for(m=0;m<lcnt;m++){
     for(n=0;n<lcnt;n++){
       t=0;
       for(l=1;l<17;l++){if(tlu[m][l]==tlu[n][l]){t++;}}
       mtc[m][n]=t;
     }
   }
   for(t=0;t<lcnt;t=t+8){
     printf("%8d/%9d/%9d/%9d/%9d/%9d/%9d/%9d/\n",t+1,t+2,t+3,t+4,t+5,t+6,t+7,t+8);
     for(l=0;l<4;l++){l4=l*4;
       for(m=t;m<(t+8);m++){
         printf(" ");
         for(n=1;n<5;n++){printf("%2d",tlu[m][l4+n]);}
         if(m<(t+7)){printf(" ");}
       }
       printf("\n");
     }
```

```
  }
  printf(" [Count of Latin Units = %d]\n",lcnt);
  printf("\n[Reference Table for the Best Combination]\n");
  printf("  *|");
  for(n=0;n<lcnt;n++){printf("%3d",n+1);}
  printf("\n");
  printf(" --+------------------------\n");
  for(m=0;m<lcnt;m++){
    printf("%3d|",m+1);
    for(n=0;n<lcnt;n++){t=mtc[m][n];
      if(t>=0){printf("%3d",t);}else{printf("  -");}}
    printf("\n");
  }
}
/**/
/* Combine and Compose */
void cmbcmp(){
 short lu,luh,md,n,d,fc;
  lu=8; luh=lu/2; md=8;
  for(u1=0;u1<luh;u1++){
    for(u2=0;u2<lu;u2++){
      if(mtc[u2][u1]==md){
      for(u3=0;u3<lu;u3++){
        if((mtc[u3][u1]==md)&&(mtc[u3][u2]==md)){
        for(u4=0;u4<lu;u4++){
          if((mtc[u4][u1]==md)&&(mtc[u4][u2]==md)&&(mtc[u4][u3]==md)){
          for(n=1;n<17;n++){uflg[n]=0;}
            for(n=1;n<17;n++){
              d=tlu[u1][n]*8+tlu[u2][n]*4+tlu[u3][n]*2+tlu[u4][n]+1;
              nm[n]=d; uflg[d]++;
            }
            fc=0;
            for(n=1;n<17;n++){if(uflg[n]==1){fc++;}else{break;}}
            if(fc==16){
              if((nm[1]<nm[16])&&(nm[1]<nm[4])&&(nm[1]<nm[13])&&(nm[2]>nm[5])){
                recordsol();}
            }
    }}}}}}
  }
}
/**/
/* Record the Answers to the Table */
void recordsol(){
 short n;
  ta[cnt][0]=cnt+1;
  for(n=1;n<17;n++){ta[cnt][n]=nm[n];}
  ta[cnt][18]=u1; ta[cnt][19]=u2;
  ta[cnt][20]=u3; ta[cnt][21]=u4;
  cnt++;
}
/**/
/* Solution Sorting for the Smart List */
void sortsol(){
 short mx,m,f;
 short d1,d2,d3,d4,d5,d6;
  mx=cnt-1;
  do{f=0;
    for(m=0;m<mx;m++){d1=ta[m][1]; d2=ta[m+1][1];
      if(d1>d2){exc(m); f=1;}
      else{
        d3=(ta[m][2]*17+ta[m][4])*17+ta[m][5];
        d4=(ta[m+1][2]*17+ta[m+1][4])*17+ta[m+1][5];
        if((d1==d2)&&(d3<d4)){exc(m); f=1;}
        else{
```

```
            d5=(ta[m][3]*17+ta[m][6])*17+ta[m][13];
            d6=(ta[m+1][3]*17+ta[m+1][6])*17+ta[m+1][13];
            if((d1==d2)&&(d3==d4)&&(d5>d6)){exc(m); f=1;}
        }}
      }
      mx--;
    }while(f>0);
}
/**/
void exc(short x){
 short n,d;
   for(n=1;n<21;n++){d=ta[x][n]; ta[x][n]=ta[x+1][n]; ta[x+1][n]=d;}
}
/**/
/* Check every sums of rows and columns */
void chksum(){
 short m,m4,n,s;
   for(m=0;m<4;m++){m4=m*4; s=0;
     for(n=1;n<5;n++){s=s+d[0][m4+n];}
     lnsm[0][m]=s; s=0;
     for(n=0;n<4;n++){s=s+d[0][n*4+m+1];}
     lnsm[0][m+4]=s; s=0;
     for(n=1;n<5;n++){s=s+d[1][m4+n];}
     lnsm[1][m]=s; s=0;
     for(n=0;n<4;n++){s=s+d[1][n*4+m+1];}
     lnsm[1][m+4]=s;
   }
   pdsm[0][0]=d[0][1]+d[0][6]+d[0][11]+d[0][16];
   pdsm[0][1]=d[0][2]+d[0][7]+d[0][12]+d[0][13];
   pdsm[0][2]=d[0][3]+d[0][8]+d[0][9]+d[0][14];
   pdsm[0][3]=d[0][4]+d[0][5]+d[0][10]+d[0][15];
   pdsm[0][4]=d[0][1]+d[0][8]+d[0][11]+d[0][14];
   pdsm[0][5]=d[0][2]+d[0][5]+d[0][12]+d[0][15];
   pdsm[0][6]=d[0][3]+d[0][6]+d[0][9]+d[0][16];
   pdsm[0][7]=d[0][4]+d[0][7]+d[0][10]+d[0][13];
/**/
   pdsm[1][0]=d[1][1]+d[1][6]+d[1][11]+d[1][16];
   pdsm[1][1]=d[1][2]+d[1][7]+d[1][12]+d[1][13];
   pdsm[1][2]=d[1][3]+d[1][8]+d[1][9]+d[1][14];
   pdsm[1][3]=d[1][4]+d[1][5]+d[1][10]+d[1][15];
   pdsm[1][4]=d[1][1]+d[1][8]+d[1][11]+d[1][14];
   pdsm[1][5]=d[1][2]+d[1][5]+d[1][12]+d[1][15];
   pdsm[1][6]=d[1][3]+d[1][6]+d[1][9]+d[1][16];
   pdsm[1][7]=d[1][4]+d[1][7]+d[1][10]+d[1][13];
}
/**/
/* Print 2 Answers */
void pr2sol(short x){
 short t,l,l4,m,n,s,v;
   for(t=0;t<x;t=t+2){
     for(n=1;n<22;n++){d[0][n]=ta[t][n]; d[1][n]=ta[t+1][n];}
     chksum();
     printf("%4d/%4d/%4d/%4d/%12d#|RW|CL|P1\\P2/%5d/%4d/%4d/%4d/%12d#|RW|CL|P1\\P2/\n",
       d[0][18]+1,d[0][19]+1,d[0][20]+1,d[0][21]+1,t+1,
       d[1][18]+1,d[1][19]+1,d[1][20]+1,d[1][21]+1,t+2);
    /**/
     for(l=0;l<4;l++){l4=l*4;
       for(s=0;s<2;s++){
         for(m=1;m<5;m++){
           printf(" ");
           for(n=1;n<5;n++){v=tlu[d[s][m+17]][l4+n];
             printf("%d",v);}
         }
         printf(" ");
```

```
        for(n=1;n<5;n++){printf("%3d",d[s][l4+n]);}
        printf("|%2d|%2d|%2d\\%2d/",lnsm[s][l],lnsm[s][l+4],pdsm[s][l],pdsm[s][l+4]);
        if(s==0){printf(" ");}
      }
      printf("\n");
    }
  }
}
/**/
/* Print 6 Answers */
void pr6sol(short x, short y){
 short t,l,l4,m,n;
  for(t=(x-1);t<y;t=t+6){
    for(m=t;m<(t+6);m++){printf("  ");
      printf("%2d/%d/%d/%d/%2d#",
        ta[m][18]+1,ta[m][19]+1,ta[m][20]+1,ta[m][21]+1,m+1);
    }
    printf("\n");
    for(l=0;l<4;l++){l4=l*4;
      for(m=t;m<(t+6);m++){printf("  ");
        for(n=1;n<5;n++){printf("%3d",ta[m][l4+n]);}
      }
      printf("\n");
    }
  }
}
/**/
```

[正方連結型汎対角線四方陣の標準解のリスト： 使用単位面//解番号#|和のチェック/]

```
  1/   2/   3/   4/          1#|RW|CL|P1\P2/   2/   1/   3/   5/          2#|RW|CL|P1\P2/
0101 0101 0110 0011   1 15  4 14|34|34|34\34/ 0101 0101 0110 1100   1 15  4 14|34|34|34\34/
1010 0101 1001 1100  12  6  9  7|34|34|34\34/ 0101 1010 1001 0011   8 10  5 11|34|34|34\34/
1010 1010 0110 0011  13  3 16  2|34|34|34\34/ 1010 1010 0110 1100  13  3 16  2|34|34|34\34/
0101 1010 1001 1100   8 10  5 11|34|34|34\34/ 1010 0101 1001 0011  12  6  9  7|34|34|34\34/

  1/   3/   2/   3/          3#|RW|CL|P1\P2/   2/   3/   1/   6/          4#|RW|CL|P1\P2/
0101 0110 0101 0110   1 15  6 12|34|34|34\34/ 0101 0110 0101 1001   1 15  6 12|34|34|34\34/
1010 1001 0101 1001  14  4  9  7|34|34|34\34/ 0101 1001 1010 0110   8 10  3 13|34|34|34\34/
1010 0110 1010 0110  11  5 16  2|34|34|34\34/ 1010 0110 1010 1001  11  5 16  2|34|34|34\34/
0101 1001 1010 1001   8 10  3 13|34|34|34\34/ 1010 1001 0101 0110  14  4  9  7|34|34|34\34/

  3/   1/   2/   3/          5#|RW|CL|P1\P2/   3/   2/   1/   6/          6#|RW|CL|P1\P2/
0110 0101 0101 0110   1 15 10  8|34|34|34\34/ 0110 0101 0101 1001   1 15 10  8|34|34|34\34/
1001 1010 0101 1001  14  4  5 11|34|34|34\34/ 1001 0101 1010 0110  12  6  3 13|34|34|34\34/
0110 1010 1010 0110   7  9 16  2|34|34|34\34/ 0110 1010 1010 1001   7  9 16  2|34|34|34\34/
1001 0101 1010 1001  12  6  3 13|34|34|34\34/ 1001 1010 0101 0110  14  4  5 11|34|34|34\34/

  1/   2/   4/   4/          7#|RW|CL|P1\P2/   2/   1/   4/   5/          8#|RW|CL|P1\P2/
0101 0101 0011 0011   1 14  4 15|34|34|34\34/ 0101 0101 0011 1100   1 14  4 15|34|34|34\34/
1010 0101 1100 1100  12  7  9  6|34|34|34\34/ 0101 1010 1100 0011   8 11  5 10|34|34|34\34/
1010 1010 0011 0011  13  2 16  3|34|34|34\34/ 1010 1010 0011 1100  13  2 16  3|34|34|34\34/
0101 1010 1100 1100   8 11  5 10|34|34|34\34/ 1010 0101 1100 0011  12  7  9  6|34|34|34\34/

  2/   3/   4/   4/          9#|RW|CL|P1\P2/   3/   2/   4/   5/         10#|RW|CL|P1\P2/
0101 0110 0011 0011   1 14  7 12|34|34|34\34/ 0110 0101 0011 1100   1 14 11  8|34|34|34\34/
0101 1001 1100 1100   8 11  2 13|34|34|34\34/ 1001 0101 1100 0011  12  7  2 13|34|34|34\34/
1010 0110 0011 0011  10  5 16  3|34|34|34\34/ 0110 1010 0011 1100   6  9 16  3|34|34|34\34/
1010 1001 1100 1100  15  4  9  6|34|34|34\34/ 1001 1010 1100 0011  15  4  5 10|34|34|34\34/

  2/   4/   1/   2/         11#|RW|CL|P1\P2/   2/   4/   3/   7/         12#|RW|CL|P1\P2/
0101 0011 0101 0101   1 12  6 15|34|34|34\34/ 0101 0011 0110 1010   1 12  7 14|34|34|34\34/
0101 1100 1010 0101   8 13  3 10|34|34|34\34/ 0101 1100 1001 1010   8 13  2 11|34|34|34\34/
1010 0011 1010 1010  11  2 16  5|34|34|34\34/ 1010 0011 0110 0101  10  3 16  5|34|34|34\34/
1010 1100 0101 1010  14  7  9  4|34|34|34\34/ 1010 1100 1001 0101  15  6  9  4|34|34|34\34/
```

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

 1/2/4/4/19#    2/1/4/5/20#    2/3/4/4/21#    3/2/4/5/22#    2/4/1/1/23#    2/4/3/8/24#
 2 13  3 16     2 13  3 16     2 13  8 11     2 13 12  7     2 11  5 16     2 11  8 13
11  8 10  5     7 12  6  9     7 12  1 14    11  8  1 14     7 14  4  9     7 14  1 12
14  1 15  4    14  1 15  4     9  6 15  4     5 10 15  4    12  1 15  6     9  4 15  6
 7 12  6  9    11  8 10  5    16  3 10  5    16  3  6  9    13  8 10  3    16  5 10  3

 1/2/5/1/25#    2/1/5/8/26#    1/3/5/4/27#    2/3/5/5/28#    3/1/5/3/29#    3/2/5/6/30#
 3 16  2 13     3 16  2 13     3 16  5 10     3 16  5 10     3 16  9  6     3 16  9  6
10  5 11  8     6  9  7 12    13  2 11  8     6  9  4 15    13  2  7 12    10  5  4 15
15  4 14  1    15  4 14  1    12  7 14  1    12  7 14  1     8 11 14  1     8 11 14  1
 6  9  7 12    10  5 11  8     6  9  4 15    13  2 11  8    10  5  4 15    13  2  7 12

 1/2/6/1/31#    2/1/6/8/32#    2/3/8/3/33#    3/2/8/6/34#    1/2/5/1/35#    2/1/5/8/36#
 3 13  2 16     3 13  2 16     3 13  8 10     3 13 12  6     4 15  1 14     4 15  1 14
10  8 11  5     6 12  7  9     6 12  1 15    10  8  1 15     9  6 12  7     5 10  8 11
15  1 14  4    15  1 14  4     9  7 14  4     5 11 14  4    16  3 13  2    16  3 13  2
 6 12  7  9    10  8 11  5    16  2 11  5    16  2  7  9     5 10  8 11     9  6 12  7

 1/3/5/4/37#    2/3/5/5/38#    3/1/5/2/39#    3/2/5/7/40#    1/2/6/4/41#    2/1/6/5/42#
 4 15  6  9     4 15  6  9     4 15 10  5     4 15 10  5     4 14  1 15     4 14  1 15
14  1 12  7     5 10  3 16    14  1  8 11     9  6  3 16     9  7 12  6     5 11  8 10
11  8 13  2    11  8 13  2     7 12 13  2     7 12 13  2    16  2 13  3    16  2 13  3
 5 10  3 16    14  1 12  7     9  6  3 16    14  1  8 11     5 11  8 10     9  7 12  6

 2/3/8/1/43#    3/2/8/8/44#    2/5/1/1/45#    2/5/3/8/46#    2/5/1/4/47#    2/5/3/5/48#
 4 14  7  9     4 14 11  5     5 16  2 11     5 16  3 10     6 15  1 12     6 15  4  9
 5 11  2 16     9  7  2 16     4  9  7 14     4  9  6 15     3 10  8 13     3 10  5 16
10  8 13  3     6 12 13  3    15  6 12  1    14  7 12  1    16  5 11  2    13  8 11  2
15  1 12  6    15  1  8 10    10  3 13  8    11  2 13  8     9  4 14  7    12  1 14  7
```

[Solution Counts = 48] [OK!]

** Calculated and Listed by Kanji Setsuda on Mar. 8, 2011 with MacOSX & Xcode 3 **

四次の汎魔方陣では，「汎型」で定義しようと「完全型」で定義しようと，或いは「正方連結完全型」で定義しようと，結果は全て同じ答が出て来る。それを，二進法による「完全オイラー四方陣」でも，全てテストして確かめられたことを報告する。

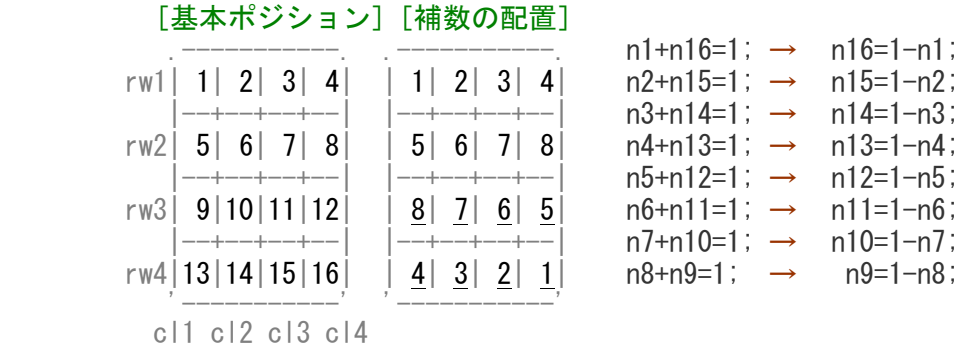
同じ解集合を得られるのだから，いずれも「同値」であると言って差し支えあるまい。

10. 二進法による対称四方陣の構成

この構成法を対称四方陣でも試してみた。汎対角線の定和は追求しないから，もはや「完全オイラー方陣」ではなくなるが，それでも二進法分解図で対称型を構成できるだろうか。

問題は，単位面の「設計」である。設計さえうまく行けば，できるのではないか？

単位面に与える条件を次のようにする。



```
    n1+n2+n3+n4=2      ... rw1;  n1+n5+n9+n13=2  ... cl1;
    n5+n6+n7+n8=2      ... rw2;  n2+n6+n10+n14=2 ... cl2;
    n9+n10+n11+n12=2   ... rw3;  n3+n7+n11+n15=2 ... cl3;
    n13+n14+n15+n16=2  ... rw4;  n4+n8+n12+n16=2 ... cl4;
    n1+n6+n11+n16=2    ... pd1;  n4+n7+n10+n13=2 ... pb4;
```

そして，プログラムを次のように組んだ。大部分は共通なので，違った部分だけを示す。

```
/**/
/* Make Latin Units */
/* Set n1 & n16 */
void sstp01(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((bcf[1][a]<2)&&(bcf[5][a]<2)){
      if((bcf[4][b]<2)&&(bcf[8][b]<2)){
        bcf[1][a]++; bcf[5][a]++;
        bcf[4][b]++; bcf[8][b]++;
        nm[1]=a; nm[16]=b;
        sstp02();
        bcf[8][b]--; bcf[4][b]--;
        bcf[5][a]--; bcf[1][a]--;
    }}
  }
}
/**/
/* Set n2 & n15 */
void sstp02(){
 short a,b;
  for(a=1;a>=0;a--){b=CC-a;
    if((bcf[1][a]<2)&&(bcf[6][a]<2)){
      if((bcf[4][b]<2)&&(bcf[7][b]<2)){
        bcf[1][a]++; bcf[6][a]++;
        bcf[4][b]++; bcf[7][b]++;
        nm[2]=a; nm[15]=b;
        sstp03();
        bcf[7][b]--; bcf[4][b]--;
        bcf[6][a]--; bcf[1][a]--;
    }}
  }
}
/**/
/* Set n3 & n14 */
void sstp03(){
 short a,b;
  for(a=1;a>=0;a--){b=CC-a;
    if((bcf[1][a]<2)&&(bcf[7][a]<2)){
      if((bcf[4][b]<2)&&(bcf[6][b]<2)){
        bcf[1][a]++; bcf[7][a]++;
        bcf[4][b]++; bcf[6][b]++;
        nm[3]=a; nm[14]=b;
        sstp04();
        bcf[6][b]--; bcf[4][b]--;
        bcf[7][a]--; bcf[1][a]--;
    }}
  }
}
/**/
/* Set n4 & n13 */
void sstp04(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((bcf[1][a]<2)&&(bcf[8][a]<2)){
```

```
      if((bcf[4][b]<2)&&(bcf[5][b]<2)){
        bcf[1][a]++; bcf[8][a]++;
        bcf[4][b]++; bcf[5][b]++;
        nm[4]=a; nm[13]=b;
        sstp05();
        bcf[5][b]--; bcf[4][b]--;
        bcf[8][a]--; bcf[1][a]--;
    }}
  }
}
/**/
/* Level 2: */
/* Set n5 & n12 */
void sstp05(){
 short a,b;
  for(a=1;a>=0;a--){b=CC-a;
    if((bcf[2][a]<2)&&(bcf[5][a]<2)){
      if((bcf[3][b]<2)&&(bcf[8][b]<2)){
        bcf[2][a]++; bcf[5][a]++;
        bcf[3][b]++; bcf[8][b]++;
        nm[5]=a; nm[12]=b;
        sstp06();
        bcf[8][b]--; bcf[3][b]--;
        bcf[5][a]--; bcf[2][a]--;
    }}
  }
}
/**/
/* Set n6 & n11 */
void sstp06(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((bcf[2][a]<2)&&(bcf[6][a]<2)){
      if((bcf[3][b]<2)&&(bcf[7][b]<2)){
        bcf[2][a]++; bcf[6][a]++;
        bcf[3][b]++; bcf[7][b]++;
        nm[6]=a; nm[11]=b;
        sstp07();
        bcf[7][b]--; bcf[3][b]--;
        bcf[6][a]--; bcf[2][a]--;
    }}
  }
}
/**/
/* Set n7 & n10 */
void sstp07(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((bcf[2][a]<2)&&(bcf[7][a]<2)){
      if((bcf[3][b]<2)&&(bcf[6][b]<2)){
        bcf[2][a]++; bcf[7][a]++;
        bcf[3][b]++; bcf[6][b]++;
        nm[7]=a; nm[10]=b;
        sstp08();
        bcf[6][b]--; bcf[3][b]--;
        bcf[7][a]--; bcf[2][a]--;
    }}
  }
}
/**/
/* Set n8 & n9 */
void sstp08(){
 short a,b;
  for(a=1;a>=0;a--){b=CC-a;
```

```
    if((bcf[2][a]<2)&&(bcf[8][a]<2)){
      if((bcf[3][b]<2)&&(bcf[5][b]<2)){
        bcf[2][a]++; bcf[8][a]++;
        bcf[3][b]++; bcf[5][b]++;
        nm[8]=a; nm[9]=b;
        clurecord();
        bcf[5][b]--; bcf[3][b]--;
        bcf[8][a]--; bcf[2][a]--;
    }}
  }
}
/**/
/* . . . */
```

＊　二進法による構成法で補数対称型四方陣の標準解を作る　＊

[二進法分解図の単位面となるラテン方陣]

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

[ラテン方陣の個数 = 8]

[任意の２面の相似度を計測した結果]

```
 *|  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
```

[補数対称四方陣の標準解のリスト：使用単位面////　解番号#|和のチェック|]

```
   1/   2/   3/   4/          1#|RW|CL|     2/   1/   3/   4/          2#|RW|CL|
0110 0110 0101 0011   1 15 14  4|34|34|  0110 0110 0101 0011   1 15 14  4|34|34|
1001 0110 1010 1100  12  6  7  9|34|34|  0110 1001 1010 1100   8 10 11  5|34|34|
0110 1001 1010 1100   8 10 11  5|34|34|  1001 0110 1010 1100  12  6  7  9|34|34|
1001 1001 0101 0011  13  3  2 16|34|34|  1001 1001 0101 0011  13  3  2 16|34|34|

   1/   3/   2/   4/          3#|RW|CL|     2/   3/   1/   4/          4#|RW|CL|
0110 0101 0110 0011   1 15 12  6|34|34|  0110 0101 0110 0011   1 15 12  6|34|34|
1001 1010 0110 1100  14  4  7  9|34|34|  0110 1010 1001 1100   8 10 13  3|34|34|
0110 1010 1001 1100   8 10 13  3|34|34|  1001 1010 0110 1100  14  4  7  9|34|34|
1001 0101 1001 0011  11  5  2 16|34|34|  1001 0101 1001 0011  11  5  2 16|34|34|

   3/   1/   2/   4/          5#|RW|CL|     3/   2/   1/   4/          6#|RW|CL|
0101 0110 0110 0011   1 15  8 10|34|34|  0101 0110 0110 0011   1 15  8 10|34|34|
1010 1001 0110 1100  14  4 11  5|34|34|  1010 0110 1001 1100  12  6 13  3|34|34|
1010 0110 1001 1100  12  6 13  3|34|34|  1010 1001 0110 1100  14  4 11  5|34|34|
0101 1001 1001 0011   7  9  2 16|34|34|  0101 1001 1001 0011   7  9  2 16|34|34|

   1/   2/   4/   3/          7#|RW|CL|     2/   1/   4/   3/          8#|RW|CL|
0110 0110 0011 0101   1 14 15  4|34|34|  0110 0110 0011 0101   1 14 15  4|34|34|
1001 0110 1100 1010  12  7  6  9|34|34|  0110 1001 1100 1010   8 11 10  5|34|34|
0110 1001 1100 1010   8 11 10  5|34|34|  1001 0110 1100 1010  12  7  6  9|34|34|
1001 1001 0011 0101  13  2  3 16|34|34|  1001 1001 0011 0101  13  2  3 16|34|34|
```

```
  2/  3/  4/  1/          9#|RW|CL|    3/  2/  4/  1/         10#|RW|CL|
0110 0101 0011 0110   1 14 12  7|34|34| 0101 0110 0011 0110   1 14  8 11|34|34|
0110 1010 1100 1001   8 11 13  2|34|34| 1010 0110 1100 1001  12  7 13  2|34|34|
1001 1010 1100 0110  15  4  6  9|34|34| 1010 1001 1100 0110  15  4 10  5|34|34|
1001 0101 0011 1001  10  5  3 16|34|34| 0101 1001 0011 1001   6  9  3 16|34|34|
  2/  4/  1/  3/          11#|RW|CL|    2/  4/  3/  1/         12#|RW|CL|
0110 0011 0110 0101   1 12 15  6|34|34| 0110 0011 0101 0110   1 12 14  7|34|34|
0110 1100 1001 1010   8 13 10  3|34|34| 0110 1100 1010 1001   8 13 11  2|34|34|
1001 1100 0110 1010  14  7  4  9|34|34| 1001 1100 1010 0110  15  6  4  9|34|34|
1001 0011 1001 0101  11  2  5 16|34|34| 1001 0011 0101 1001  10  3  5 16|34|34|

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

1/2/4/6/19#   2/1/4/6/20#   2/3/4/8/21#   3/2/4/8/22#   2/4/1/6/23#   2/4/3/8/24#
 2 13 16  3    2 13 16  3    2 13 11  8    2 13  7 12    2 11 16  5    2 11 13  8
11  8  5 10    7 12  9  6    7 12 14  1   11  8 14  1    7 14  9  4    7 14 12  1
 7 12  9  6   11  8  5 10   16  3  5 10   16  3  9  6   13  8  3 10   16  5  3 10
14  1  4 15   14  1  4 15    9  6  4 15    5 10  4 15   12  1  6 15    9  4  6 15

1/3/5/2/25#   2/3/5/1/26#   3/1/5/2/27#   3/2/5/1/28#   2/3/8/4/29#   3/2/8/4/30#
 3 16 10  5    3 16 10  5    3 16  6  9    3 16  6  9    3 13 10  8    3 13  6 12
13  2  8 11    6  9 15  4   13  2 12  7   10  5 15  4    6 12 15  1   10  8 15  1
 6  9 15  4   13  2  8 11   10  5 15  4   13  2 12  7   16  2  5 11   16  2  9  7
12  7  1 14   12  7  1 14    8 11  1 14    8 11  1 14    9  7  4 14    5 11  4 14

2/4/6/1/31#   2/4/8/3/32#   1/3/5/7/33#   2/3/5/8/34#   3/1/5/7/35#   3/2/5/8/36#
 3 10 16  5    3 10 13  8    4 15  9  6    4 15  9  6    4 15  5 10    4 15  5 10
 6 15  9  4    6 15 12  1   14  1  7 12    5 10 16  3   14  1 11  8    9  6 16  3
13  8  2 11   16  5  2 11    5 10 16  3   14  1  7 12    9  6 16  3   14  1 11  8
12  1  7 14    9  4  7 14   11  8  2 13   11  8  2 13    7 12  2 13    7 12  2 13

2/3/8/5/37#   3/2/8/5/38#   2/4/6/8/39#   2/4/8/6/40#   3/5/1/2/41#   3/5/2/1/42#
 4 14  9  7    4 14  5 11    4  9 15  6    4  9 14  7    5 16  4  9    5 16  4  9
 5 11 16  2    9  7 16  2    5 16 10  3    5 16 11  2   11  2 14  7   10  3 15  6
15  1  6 12   15  1 10  8   14  7  1 12   15  6  1 12   10  3 15  6   11  2 14  7
10  8  3 13    6 12  3 13   11  2  8 13   10  3  8 13    8 13  1 12    8 13  1 12

2/8/3/4/43#   2/8/4/3/44#   3/5/1/7/45#   3/5/2/8/46#   2/8/3/5/47#   2/8/4/6/48#
 5 11 10  8    5 10 11  8    6 15  3 10    6 15  3 10    6 12  9  7    6  9 12  7
 4 14 15  1    4 15 14  1   12  1 13  8    9  4 16  5    3 13 16  2    3 16 13  2
16  2  3 13   16  3  2 13    9  4 16  5   12  1 13  8   15  1  4 14   15  4  1 14
 9  7  6 12    9  6  7 12    7 14  2 11    7 14  2 11   10  8  5 11   10  5  8 11
[標準解の総数 = 48]

* 新旧の解のリストを付き合わせて対応関係をモニタとした結果 *
 ??:  0
  1:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 25:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1

 [OK!]
** Calculated and Listed by Kanji Setsuda on Mar.20, 2011 with MacOSX & Xcode 3 **
```

単位面は，８形出て来た。汎四方陣の場合と同じである。

それを組み合わせて解を合成すると，上のリストのように 48 個の解が得られた。これは，前に得ていた解と内容が全く同じものである。

対称四方陣も，二進法で構成できるのであった。

今後は，必ずしも「完全」にこだわらずに「オイラー方陣」を組んで良いことになる。

ただし，二進法によるこの構成法は，四次，八次，十六次，‥‥の場合にしか使えない。二進法分解図が満足に作れるのは，ｎの２乗が２の累乗値に等しい場合に限られるからである。従って，六次や十次はダメである。

$32 < 6^2 < 64$ ； $64 < 10^2 < 128$ ； $128 < 12^2 < 256$ ； ...

五次，七次は既に見ての通り，ｎ進法を使う。九次は，九進法のかわりに三進法が使える。

高次になればなるほど，この構成法は威力を発揮する。ただし，作れるのは「オイラー方陣」に限られる。より多くあるはずの「ノン・オイラー型」については，まだ良い構成法が何も見つかっていない。それは，もう仕方が無いのではないかと，小生は殆ど諦めている。

それにしても，四次魔方陣の「奇跡の一致」については，いつ出逢っても不思議に思う。

汎対角線四方陣，**完全四方陣**，**正方連結完全四方陣**の３つのタイプが，全て同じ解集合を持ち，論理学的に「同値」であること。そして，**補数対称四方陣**は，解の総数が 48 個という点が同じで，しかも完全四方陣の解とは一対一の対応関係があること。比喩的に言えば，これらの全てが構造的に「表裏一体」なのである。

高次なると，それが分化しつつ別個の道をたどって進化して行く。

四次が何故同一なのかは，それらが同じ構造を持つことで，説明が付く。が，それでも，不思議の感情はおさまらない。

この辺りで，小生はいつも宇宙の進化を連想してしまう。四方陣・五方陣は，ビッグバン宇宙の初期の「対称性の高い」時期の全体の構造に似ていないか。

或いは，数の世界にも特有の「創造と進化」の神話があるのか，と思う。空想が膨らんで妄想にさえひたる。

11. 完全型と対称型の間の変換について

４次の魔方陣では，次のような**型変換**の方法が可能である。縦も横も同時に３行/列目と４行/列目を相互に入れ替える。縦から始めても良いし，横から始めても良いが，必ず両方向を入れ替える。それだけで，対称型が完全型に，完全型は対称型に変換できる。

```
        1#S   1#C                 2#S  25#C                 3#S   7#C
 1 15 14  4    1 15  4 14    2 16 13  3    2 16  3 13    1 14 15  4    1 14  4 15
12  6  7  9   12  6  9  7   11  5  8 10   11  5 10  8   12  7  6  9   12  7  9  6
 8 10 11  5   13  3 16  2    7  9 12  6   14  4 15  1    8 11 10  5   13  2 16  3
13  3  2 16    8 10  5 11   14  4  1 15    7  9  6 12   13  2  3 16    8 11  5 10

        21#S  ?#C                 41#S  ?#C
 5 16  4  9    5 16  9  4    3 10 16  5    3 10  5 16
11  2 14  7   11  2  7 14    6 15  9  4    6 15  4  9
10  3 15  6    8 13 12  1   13  8  2 11   12  1 14  7
 8 13  1 12   10  3  6 15   12  1  7 14   13  8 11  2
```

```
/**/
/* 型変換のプログラム */
   s[1]=t[1];  s[2]=t[2];   s[3]=t[4];   s[4]=t[3];
   s[5]=t[5];  s[6]=t[6];   s[7]=t[8];   s[8]=t[7];
   s[9]=t[13]; s[10]=t[14]; s[11]=t[16]; s[12]=t[15];
   s[13]=t[9]; s[14]=t[10]; s[15]=t[12]; s[16]=t[11];
/**/
```

ただし，48 個の標準解全数について「一括変換」を行うことはできない。リストの後半で必ず「一対一対応」の関係が破れてしまうからである。

実例の ?#C のように前のリストに乗っていないような解が出来てしまう。何故か？

変換した結果が，リスト出力時に課した不等式の条件に違反した形になってしまうからである。我々は，いつでも n1<n16; n1<n4; n1<n13; n2>n5; の条件を付けて解を整理する習慣になっている。リストの後半に限って，変換した解がこの約束事とコンフリクトを起こしてしまうのである。これは，避けようが無い。

どうしても標準解全部を型変換したければ，最初に戻って，リスト整形用の不等式が働く以前の「原始解」384 個を作った段階で，型変換を行うしか無い。そして，その後で 48 個の標準解をあらためて選ぶようにする。そうすれば，正しい解集合を得ることができる。

[原始解 384 個のリスト（部分）：補数対称型 S・汎対角線型 C：使用単位面////　解番号#]

1/	2/	3/	4/	1#S	1#C	1/	2/	3/	5/	2#S	25#C
0110	0110	0101	0011	1 15 14 4	1 15 4 14	0110	0110	0101	1100	2 16 13 3	2 16 3 13
1001	0110	1010	1100	12 6 7 9	12 6 9 7	1001	0110	1010	0011	11 5 8 10	11 5 10 8
0110	1001	1010	1100	8 10 11 5	13 3 16 2	0110	1001	1010	0011	7 9 12 6	14 4 15 1
1001	1001	0101	0011	13 3 2 16	8 10 5 11	1001	1001	0101	1100	14 4 1 15	7 9 6 12
1/	2/	4/	3/	3#S	7#C	1/	2/	4/	6/	4#S	31#C
0110	0110	0011	0101	1 14 15 4	1 14 4 15	0110	0110	0011	1010	2 13 16 3	2 13 3 16
1001	0110	1100	1010	12 7 6 9	12 7 9 6	1001	0110	1100	0101	11 8 5 10	11 8 10 5
0110	1001	1100	1010	8 11 10 5	13 2 16 3	0110	1001	1100	0101	7 12 9 6	14 1 15 4
1001	1001	0011	0101	13 2 3 16	8 11 5 10	1001	1001	0011	1010	14 1 4 15	7 12 6 9
1/	2/	5/	3/	5#S	49#C	1/	2/	5/	6/	6#S	73#C
0110	0110	1100	0101	3 16 13 2	3 16 2 13	0110	0110	1100	1010	4 15 14 1	4 15 1 14
1001	0110	0011	1010	10 5 8 11	10 5 11 8	1001	0110	0011	0101	9 6 7 12	9 6 12 7
0110	1001	0011	1010	6 9 12 7	15 4 14 1	0110	1001	0011	0101	5 10 11 8	16 3 13 2
1001	1001	1100	0101	15 4 1 14	6 9 7 12	1001	1001	1100	1010	16 3 2 13	5 10 8 11
1/	2/	6/	4/	7#S	55#C	1/	2/	6/	5/	8#S	79#C
0110	0110	1010	0011	3 13 16 2	3 13 2 16	0110	0110	1010	1100	4 14 15 1	4 14 1 15
1001	0110	0101	1100	10 8 5 11	10 8 11 5	1001	0110	0101	0011	9 7 6 12	9 7 12 6
0110	1001	0101	1100	6 12 9 7	15 1 14 4	0110	1001	0101	0011	5 11 10 8	16 2 13 3
1001	1001	1010	0011	15 1 4 14	6 12 7 9	1001	1001	1010	1100	16 2 3 13	5 11 8 10
1/	3/	2/	4/	9#S	3#C	1/	3/	2/	5/	10#S	27#C
0110	0101	0110	0011	1 15 12 6	1 15 6 12	0110	0101	0110	1100	2 16 11 5	2 16 5 11
1001	1010	0110	1100	14 4 7 9	14 4 9 7	1001	1010	0110	0011	13 3 8 10	13 3 10 8
0110	1010	1001	1100	8 10 13 3	11 5 16 2	0110	1010	1001	0011	7 9 14 4	12 6 15 1
1001	0101	1001	0011	11 5 2 16	8 10 3 13	1001	0101	1001	1100	12 6 1 15	7 9 4 14
1/	3/	4/	2/	11#S	9#C	1/	3/	4/	7/	12#S	33#C
0110	0101	0011	0110	1 14 12 7	1 14 7 12	0110	0101	0011	1001	2 13 11 8	2 13 8 11
1001	1010	1100	0110	15 4 6 9	15 4 9 6	1001	1010	1100	1001	16 3 5 10	16 3 10 5
0110	1010	1100	1001	8 11 13 2	10 5 16 3	0110	1010	1100	0110	7 12 14 1	9 6 15 4
1001	0101	0011	1001	10 5 3 16	8 11 2 13	1001	0101	0011	0110	9 6 4 15	7 12 1 14
1/	3/	5/	2/	13#S	51#C	1/	3/	5/	7/	14#S	75#C
0110	0101	1100	0110	3 16 10 5	3 16 5 10	0110	0101	1100	1001	4 15 9 6	4 15 6 9
1001	1010	0011	0110	13 2 8 11	13 2 11 8	1001	1010	0011	1001	14 1 7 12	14 1 12 7
0110	1010	0011	1001	6 9 15 4	12 7 14 1	0110	1010	0011	0110	5 10 16 3	11 8 13 2
1001	0101	1100	1001	12 7 1 14	6 9 4 15	1001	0101	1100	0110	11 8 2 13	5 10 3 16
1/	3/	7/	4/	15#S	57#C	1/	3/	7/	5/	16#S	81#C
0110	0101	1001	0011	3 13 10 8	3 13 8 10	0110	0101	1001	1100	4 14 9 7	4 14 7 9
1001	1010	1001	1100	16 2 5 11	16 2 11 5	1001	1010	1001	0011	15 1 6 12	15 1 12 6
0110	1010	0110	1100	6 12 15 1	9 7 14 4	0110	1010	0110	0011	5 11 16 2	10 8 13 3
1001	0101	0110	0011	9 7 4 14	6 12 1 15	1001	0101	0110	1100	10 8 3 13	5 11 2 16
1/	4/	2/	3/	17#S	13#C	1/	4/	2/	6/	18#S	37#C
0110	0011	0110	0101	1 12 15 6	1 12 6 15	0110	0011	0110	1010	2 11 16 5	2 11 5 16
1001	1100	0110	1010	14 7 4 9	14 7 9 4	1001	1100	0110	0101	13 8 3 10	13 8 10 3
0110	1100	1001	1010	8 13 10 3	11 2 16 5	0110	1100	1001	0101	7 14 9 4	12 1 15 6
1001	0011	1001	0101	11 2 5 16	8 13 3 10	1001	0011	1001	1010	12 1 6 15	7 14 4 9
1/	4/	3/	2/	19#S	15#C	1/	4/	3/	7/	20#S	39#C
0110	0011	0101	0110	1 12 14 7	1 12 7 14	0110	0011	0101	1001	2 11 13 8	2 11 8 13
1001	1100	1010	0110	15 6 4 9	15 6 9 4	1001	1100	1010	1001	16 5 3 10	16 5 10 3
0110	1100	1010	1001	8 13 11 2	10 3 16 5	0110	1100	1010	0110	7 14 12 1	9 4 15 6
1001	0011	0101	1001	10 3 5 16	8 13 2 11	1001	0011	0101	0110	9 4 6 15	7 14 1 12

```
  1/   4/   6/   2/         21#S         61#C    1/   4/   6/   7/         22#S         85#C
0110 0011 1010 0110   3 10 16  5   3 10  5 16  0110 0011 1010 1001   4  9 15  6   4  9  6 15
1001 1100 0101 0110  13  8  2 11  13  8 11  2  1001 1100 0101 1001  14  7  1 12  14  7 12  1
0110 1100 0101 1001   6 15  9  4  12  1 14  7  0110 1100 0101 0110   5 16 10  3  11  2 13  8
1001 0011 1010 1001  12  1  7 14   6 15  4  9  1001 0011 1010 0110  11  2  8 13   5 16  3 10
  1/   4/   7/   3/         23#S         63#C    1/   4/   7/   6/         24#S         87#C
0110 0011 1001 0101   3 10 13  8   3 10  8 13  0110 0011 1001 1010   4  9 14  7   4  9  7 14
1001 1100 1001 1010  16  5  2 11  16  5 11  2  1001 1100 1001 0101  15  6  1 12  15  6 12  1
0110 1100 0110 1010   6 15 12  1   9  4 14  7  0110 1100 0110 0101   5 16 11  2  10  3 13  8
1001 0011 0110 0101   9  4  7 14   6 15  1 12  1001 0011 0110 1010  10  3  8 13   5 16  2 11

 . . . . . .
```

＊ 新旧のリストで解の対応関係をモニターした結果 ＊

```
 ??: 0
  1: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 41: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 81: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
121: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
161: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
201: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
241: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
281: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
321: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
361: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   [OK!]
```

上のリストは，原始解 384 個のレベルで「型変換」を行った結果を表している。全数が，見事に**一対一対応**している。

試みに二進法の単位面に直接当たって，完全型（汎対角線型）と補数対称型がどう違うかを具体的に調べておこう。

[二進法分解図の単位面８個：補数対称四方陣用]

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

[二進法分解図の単位面８個：完全四方陣用]

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

上のリストを良く調べてみると，それぞれの内容が少しずつ違っているが，同じ番号どうしは，例外無くお互いの第３行/列と第４行/列の内容を交換した形になっている。つまり，上の「型変換」の概念がそのまま単位面でも成り立つことがわかる。

こうして根本のところで「型変換」が成り立つのだから，汎対角線型（完全型）と補数対称型の間には，この変換法によって「一対一対応」が本質的に成り立つと，言える。

この２つのタイプの魔方陣は，１つの複合体の言わば「表と裏」の関係にある。

（初稿：October 5, 2003; 改訂稿：April 7, 2006 and March 20, 2011;
清書稿：May 24, 2017 with macOS Sierra & Xcode7.3 by 摂田 寛二(Kanji Setsuda)）

E-mail Address:<jag12001@nifty.com>