数独（シャドウ付き）の解法 (未定稿)

SHIMURA Masato
JCD02773@nifty.ne.jp

2011 年 1 月 19 日

目次

1 はじめに 1

2 入力と閲覧 2

3 sudoku の解法 3

4 解を求める 10

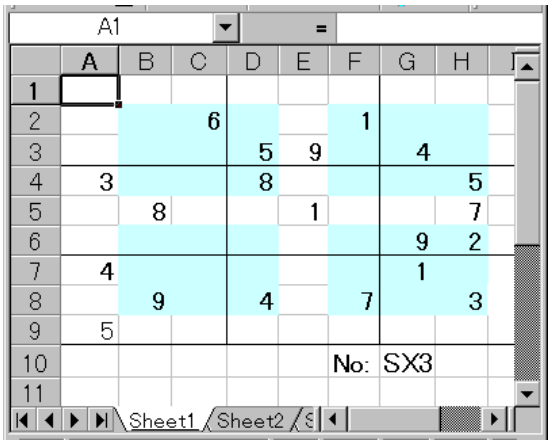
付録 A Hui の Script 12

1 はじめに

西川「数独 on EXCEL_J を楽しむ（J sudoku でどうやって数独の問題を解くか）」を熟読した。*Roger Hui* の凝縮された簡潔な解法の解説と優れたユーティリティ関数を添えたものである。

Herald Asahi に掲載されている数独はコントラクト・ブリッジと同じ欄にあるのでトライしたが次のように 4×4 のシャドウが *4* 個付加されておりそのままでは解けないので *Hui* と西川の関数群を少し改良することにした。

高度に凝縮されたゲームの *Script* は学習用の格好の素材であるので簡単な解説を加える




```
   cutmat SX3
+-----+-----+-----+
|0 0 0|0 0 0|0 0 0|
|0 0 6|0 0 1|0 0 0|
|0 0 0|5 9 0|4 0 0|
+-----+-----+-----+
|3 0 0|8 0 0|0 5 0|
|0 8 0|0 1 0|0 7 0|
|0 0 0|0 0 0|9 2 0|
+-----+-----+-----+
|4 0 0|0 0 0|1 0 0|
|0 9 0|4 0 7|0 3 2|
|5 0 0|0 0 0|8 0 0|
+-----+-----+-----+
```

cutmat 便利なユーティリティ

2　入力と閲覧

jis **ファイルへの記入**　次のマトリクスによる。9×9

```
NB. No: 113 2011/01/13 Herald
SX3=: 0". ] ;._2 (0 : 0)
0 0 0 0 0 0 0 0 0
0 0 6 0 0 1 0 0 0
0 0 0 5 9 0 4 0 0
3 0 0 8 0 0 0 5 0
0 8 0 0 1 0 0 7 0
0 0 0 0 0 0 9 2 0
4 0 0 0 0 0 1 0 0
0 9 0 4 0 7 0 3 2
5 0 0 0 0 0 8 0 0
)
```

J の関数への入力　1×81 のリスト（ベクトル）。(,) でリストにする. *ex.* ,SX3

閲覧　*1.*　*see* リストを 9×9 のマトリクスに直し、解を表示する

```
NB. Nishikawa's Modifications
see0 =: ({&'.123456789') @ (9 9&$) @ ,NB. modified by TN
see1 =: (3 3 ,: 3 3)&(<;.3) @ see0 NB. modified by TN
see =: <@see1"1`see1@.(1:=#@$)
```

2.　*cutmat 1* 参照

3.　*smat shadow* 部分の抜き出し

```
   smat SX3
+-----+-----+
|0 6 0|1 0 0|
|0 0 5|0 4 0|
|0 0 8|0 0 5|
+-----+-----+
|0 0 0|0 9 2|
|0 0 0|0 1 0|
|9 0 4|7 0 3|
+-----+-----+
```

3　sudoku の解法

3.1　free を計算する

9×9 の各ポイントでどの数を入れることができるか *(free)* を探す。*I* で *81* 行 ×27(=3×9) のインデクスを作成する。行、列、ボックスのインデクスを横に並べる。

Hui の *Script*　*I* が *free* 用の *index*

```
r =. 9#i.9 9
c =. 81$|:i.9 9
b =. (,j{9#i.9) { j
I =: ~."1 r,.c,.b
free  =: 0&= > (1+i.9) e."1 I&{
```

r	*raw*	行	81 個×9 行
c	*column*	列	81 個×9 列
b	*box*	ボックス	(3×3)×9 個
I	*index*	インデクス	81 個×27(行 + 列 + *Box*)

shadow を付加する *Hui* の *Script* に *shadow* 部分を付加する

1. *shadou* の抜き出し
   shadow の部分を *I* に付加すると (81×36) になるが *0* がインデクスになり最初のマス（左上　$0,0$*)* を取り出してしまうので、先に数値を取り出して同じ形に加工する

```
smat SX3 NB.numbers
+-----+-----+
|0 6 0|1 0 0|
|0 0 5|0 4 0|
|0 0 8|0 0 5|
+-----+-----+
|0 0 0|0 9 2|
|0 0 0|0 1 0|
|9 0 4|7 0 3|
+-----+-----+
```

```
 smat i.9 9 NB. index
+--------+--------+
|10 11 12|14 15 16|
|19 20 21|23 24 25|
|28 29 30|32 33 34|
+--------+--------+
|46 47 48|50 51 52|
|55 56 57|59 60 61|
|64 65 66|68 69 70|
+--------+--------+
```

2. *shadou* の部分を *expand* して加える。空白行には *0* が入るが次の *e.* ではカウントされない。

```
expind0=: (;i.9 9) e. ; smat0 i.9 9  NB. index for expand
NB.  take and expand
shadow=: 3 : 'expind0 expand (s0 y) { y'       NB. , SX1
```

   e. members of

3. *free_s* に変更

```
free0=: 3 : ' (1+i.9) e."1 (I {  y) ,. shadow y'  NB. ,SX3
```

4. *Hui* の *script* は `(1+i.9) e. "1` で *1* から *9* の数字を使っているかどうか照査する。左に *1-9* が来るので複数個の数値を *9* 個のベクトルを用いて *0/1* で表している

5. `0&= >`で 数字が既に埋まっている箇所を識別する
   ここを *free_s* に書き換えた

```
free_s=: 0&= > free0
```

3.2 ac と ar

free で作成したデータを解析して解を導く関数として *ac* と *ar* が用意されている。*ac* はスポット、*ar* は行や列単位で解読する。高難度では *ac,ar* で候補が見つからない場合に推測 *(guess)* を併用する。

ac .

1. >./ *Large of* 大きい方をとる

```
   0 5 >./   2 3
0|2 3
5|5 5
```

2. (1:=+/ "1)

   (1:=) で書き換えできるポイントを探る *(* (1=+/"1) でも同じ)

```
9 9 $ ( 1:= +/"1) free_s ,SX3
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

3. ac=: +/ .*&(1+i.9) * 1: = +/"1

   fork になっている。

```
                *
              /   \
+/ .*&(1+i.9)     1: = +/"1
```

```
                                                  see ac@free_s ,SX3
                                                +---+---+---+
          9 9 $ ac@free_s ,SX3                  |...|...|...|
       0 0 0 0 0 0 0 0 0                        |...|...|...|
       0 0 0 0 0 0 0 0 0                        |...|...|...|
       0 0 0 0 0 0 0 0 0                        +---+---+---+
       0 0 0 0 0 0 6 0 0                        |...|...|6..|
       0 0 0 0 0 0 0 0 0                        |...|...|...|
       0 0 0 0 0 0 0 0 0                        |...|...|...|
       0 0 0 0 0 0 0 6 0                        +---+---+---+
       0 0 0 0 0 0 0 0 0                        |...|...|.6.|
       0 0 0 0 0 0 0 0 0                        |...|...|...|
                                                |...|...|...|
                                                +---+---+---+
```

ar .

Hui のオリジナル

```
j =. (]/. i.@#) ,{;~3#i.3
R =: j,(,|:)i.9 9
ar  =: 3 : 0
 m=. 1=+/"2 R{y
 j=. I. +./"1 m
 k=. 1 i."1~ j{m
 i=. ,(k{"_1 |:"2 (j{R){y) #"1 j{R
 (1+k) i}81$0
)
```

次のように *shadow* 部分を追加した

```
ar0=: 3 : 0
a0=. +/"2 R{y
a1=. +/"2 shadow_ind{y
m=: 1= a=. a0,a1
 jj=: I.+. /"1 m
 k =: 1 i."1~ jj{m
```

```
R0=: R,shadow_ind
 i =: ,(k{"_1 |:"2 (jj{R0){y) #"1 jj{R0
 (1+k) i}81$0
)
```

■経過と説明 *j*

1. *j* は *Box* を作成する巧妙な *script* である

```
   j =. (]/. i.@#) ,{;~3#i.3
```

2.

```
      (3#i.3);3#i.3
+-----------------+-----------------+
|0 0 0 1 1 1 2 2 2|0 0 0 1 1 1 2 2 2|
+-----------------+-----------------+
```

3. { カタログ 上の *Box* を縦横に配置して全ての組み合わせを作る

```
  { (3#i.3);3#i.3
+---+---+---+---+---+---+---+---+---+
|0 0|0 0|0 0|0 1|0 1|0 1|0 2|0 2|0 2|
+---+---+---+---+---+---+---+---+---+
|0 0|0 0|0 0|0 1|0 1|0 1|0 2|0 2|0 2|
+---+---+---+---+---+---+---+---+---+
|0 0|0 0|0 0|0 1|0 1|0 1|0 2|0 2|0 2|
+---+---+---+---+---+---+---+---+---+
|1 0|1 0|1 0|1 1|1 1|1 1|1 2|1 2|1 2|
+---+---+---+---+---+---+---+---+---+
|1 0|1 0|1 0|1 1|1 1|1 1|1 2|1 2|1 2|
+---+---+---+---+---+---+---+---+---+
|1 0|1 0|1 0|1 1|1 1|1 1|1 2|1 2|1 2|
+---+---+---+---+---+---+---+---+---+
|2 0|2 0|2 0|2 1|2 1|2 1|2 2|2 2|2 2|
+---+---+---+---+---+---+---+---+---+
|2 0|2 0|2 0|2 1|2 1|2 1|2 2|2 2|2 2|
+---+---+---+---+---+---+---+---+---+
|2 0|2 0|2 0|2 1|2 1|2 1|2 2|2 2|2 2|
+---+---+---+---+---+---+---+---+---+
```

```
 a=.,  { (3#i.3);3#i.3
```

4. */.* は *Key*

   ```
   ,. a </. i.# a
   ```

   参照

   ```
    (] /. i.@ #) a  NB. j for Box
    0  1  2  9 10 11 18 19 20
    3  4  5 12 13 14 21 22 23
    6  7  8 15 16 17 24 25 26
   27 28 29 36 37 38 45 46 47
   30 31 32 39 40 41 48 49 50
   33 34 35 42 43 44 51 52 53
   54 55 56 63 64 65 72 73 74
   57 58 59 66 67 68 75 76 77
   60 61 62 69 70 71 78 79 80
   ```

1. *R* は *hook* `(,|:)` を用いている

   ```
   R =: j,(,|:)i.9 9
   ```

   R は *Box*、行、列を (3×9 =)27×9 のマトリクスに並べたものである。

2. *ar* に *shadou* 部分のインデクスを付加する

   ```
   shadow_ind=: ;("1),. , ,L:0 smat i.9 9 NB. SX1
      shadow_ind
   10 11 12 19 20 21 28 29 30
   14 15 16 23 24 25 32 33 34
   46 47 48 55 56 57 64 65 66
   50 51 52 59 60 61 68 69 70
   ar0=: 3 : 0
   a0=. +/"2 R{y
   a1=. +/"2 shadow_ind{y
   m=: 1= a=. a0,a1
   ```

3. `I. +./ "1 m` *GCD* を利用して *9* 個の *0/1* のベクトルの行のどこかにに *1* が有れば *1*、全て *0* ならば *0* とする。そして *I.* で *1* の立った行番号を得る。

4. *1* の立った行の元のデータ *(m)* を取り直す

   ```
      jj{m
   0 0 0 0 0 0 0 1 0
   1 0 0 0 0 0 0 0 0
   ```

```
0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0
1 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0 0
0 0 0 1 0 0 0 1 0
```

5. *ar* を一回実行してで得た値（*k)*（*1* を加えて解とする）

```
 k =: 1 i."1~ jj{m
   1 i."1~(jj{m)
7 0 4 7 8 0 0 8 6 3
```

両項の *i* は *y* の *x* での位置を示す

```
   'abcde' i. 'ac'
0 2
```

x を *1* とすると次となる

```
  1 i. 0 1 1 0
NB 0 is Yes 1 in No
1 0 0 1


   1 e.~ 0 1 1 0
0 1 1 0
```

6. *R* に *shadou* 部分の書き換えインデクスを加える

```
R0=: R,shadow_ind
 i =: ,(k{"_1 |:"2 (jj{R0){y) #"1 jj{R0
   k,.i
7 53
0 75
4 17
7 53
8 62
0 75
0  7
8 29
6 15
3 50
```

4　解を求める

assign　*assign guess* の *free* を *free_s* に差し替える

```
assign_s0 =: (+ (ac >. ar0)@free_s)
```

(ac >. ar)@free_s 大きい方を取るが片方は必ず *0* なので *ac,ar* で選んだ数値を取っている。
ここで手で解いていて次の一手が分からないときのアシストができる

```
assign_s0 =: (+ (ac >. ar0)@free_s)
assign_s=: assign_s0 ^:_ "1
```

^:_ "1 は *tacit* のループ
ループをかけると数回で殆どの問題は解が得られる

guess　一位にポイントが埋まらない場合は *guess* を用いる

```
guess_as =: 3 : 0
 if. -. 0 e. y do. ,:y return. end.
 b =. free_s y
 i =. (i.<./) (+/"1 b){10,}.i.10
 y +"1 (1+ Ip i{b)*/i=i.81
)
```

sudoku　フルセット

```
guess_s =: ; @: (<@guess_as"1)
sudoku_s =: guess_s @: (ok #]) @: assign_s ^:_ @ ,
```

```
 ; see  sudoku_s ,SX3
+---+---+---+
|954|786|213|
|836|241|795|
|271|593|486|
+---+---+---+
|349|872|651|
```

```
|682|915|374|
|715|634|928|
+---+---+---+
|427|358|169|
|198|467|532|
|563|129|847|
+---+---+---+
```

付録 A　Hui の Script

http://www.jsoftware.com *showcase Essay Sudoku*

```
j      =. (]/. i.@#) ,{;~3#i.3
r      =. 9#i.9 9
c      =. 81$|:i.9 9
b      =. (,j{9#i.9) { j

I      =: ~."1 r,.c,.b
R      =: j,(,|:)i.9 9

regions=: R {"_ 1 ]
free   =: 0&= > (1+i.9) e."1 I&{
ok     =: (27 9$1) -:"2 (0&= +. ~:"1)@regions

ac     =: +/ .*&(1+i.9) * 1 = +/"1

ar     =: 3 : 0
 m=. 1=+/"2 R{y
 j=. I. +./"1 m
 k=. 1 i."1~ j{m
 i=. ,(k{"_1 |:"2 (j{R){y) #"1 j{R
 (1+k) i}81$0
)

assign =: (+ (ac >. ar)@free)^:_"1

guessa =: 3 : 0
 if. -. 0 e. y do. ,:y return. end.
 b=. free y
 i=. (i.<./) (+/"1 b){10,}.i.10
```

```
 y +"1 (1+I.i{b)*/i=i.81
)

guess  =: ; @: (<@guessa"1)
sudoku =: guess @: (ok # ]) @: assign ^:_ @ ,

see1   =: (;~9$1 0 0)&(<;.1) @ ({&'.123456789') @ (9 9&$) @ ,
see    =: <@see1"1`see1@.(1:=#@$)
diff   =: * 0&=@}:@(0&,)
```

References

西川「数独 *on EXCEL_J* を楽しむ（*J sudoku* でどうやって数独の問題を解くか） *JAPLA symposium Dec./2010*

`http://japla.sakura.ne.jp` *symposium 2010*

Miscellance

J602 is download available (No charge)

`http://www.jsoftware.com`

Scripts are accessible

`http://japla.sakura.ne.jp`