**数独システムのアルゴリズム**

**＊はじめに**

数独はパズルなので、自分の頭で解いて楽しむものです。

コンピュータに解いてもらっても全然楽しくありません。

本システムは数独問題を解くコードを持っていますが、このコードを利用して下記のものを提供するのが目的です。

(1) 新しい数独問題を作り出す

(2) 既存の問題に対して解くヒントを出す

(3) 自分で解いた答えを確認する

以下の説明は、一般的な数独の解法を、どのようなアルゴリズムでコード化しているのかを説明しています。

**1. 言葉の定義**

**\* 三次元化 \***

数独システムのコードでは数独問題をまず三次元化して考えます。

**５**

**列方向(j)**

**行方向(i)**

**カラム方向(k)**

**ブロック方向(b)**

**三次元化**

**カラム方向５番目のノードに該当**

（Fig.A 三次元化の概念）

図のように数独問題は9×9×9個=729個の箱からなる三次元立体に変換する事が出来ます。

例えば、左上のマスが"5"だとすると、三次元化すると、カラム方向の５番目の箱が”真”ということになります。（真の意味については後述）

**\* ノード \***

三次元化した立体（キューブ）にはi-j面、i-k面、j-k面、b-k面の四つの面があることになります。

四つの面はそれぞれ９面ずつありますから、全部で３６面あることになります。

b-k面は３×３の立体なので、９×９にひきのばして考えるとわかりやすいです。

ほとんどの数独問題はこの四つの面の中で解法を考える事が出来ます。

この立方体には9×9×9個=729個の箱があると言いましたが。私はそれぞれの箱を

**"ノード"**と呼んでいます。

**\* ノードの状態 \***

この箱の中身には三種類の状態が存在します。**"真：True"**の玉が入った箱,**"偽：False"**の玉が入った箱,**"空：Vacant"**つまり何も入っていない箱です。

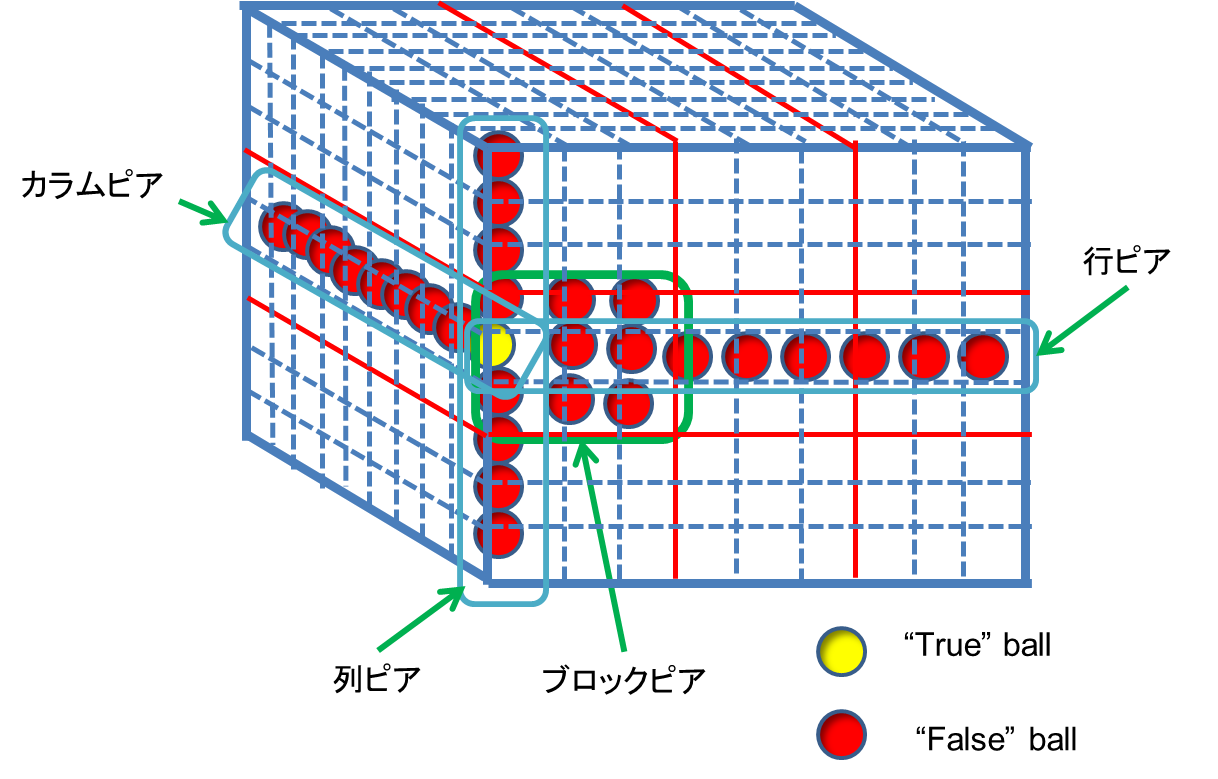
一つの箱に"真"が入ると、数独のルールにより、i-j面、i-k面、j-k面、ブロック面の全てで、この"真"の箱に影響される箱に"偽"が入ります。

同じピア内には”真”の玉はひとつしか入らないからです。

**\* ピア \***

一つの"真"に影響される９つの箱のグループを私はピアと呼んでいます。Fig.Bのように、一つの”真”の箱（ノード）には４つのピアが連動しています。

Fig.B ピア



便宜上、i方向のピアを"列ピア"、j方向のピアを"行ピア"、ブロック方向のピアを"ブロックピア"、k方向のピアを"カラムピア"と呼ぶことにします。

数独問題では通常9×9=81個のマスの内、２０数個から３０個のマスに数字が入っています。Fig.Bでは二次元で言うと５行目1列目に９が入った状態をあらわしています。

するとこのノードに関連する４つのピアの箱の全てに"偽"の玉が入ることになります。

言い換えると、最終的に各ピアには必ず一つの"真"の箱と、８つの"偽"の箱ができることになります。

ちょっと話がそれますが、数独で使用されている1~9の数字は数字としてではなく単に記号として使われているだけです。

ですから、１〜９の代わりにa〜iを使っても構わないわけです。

**\* マスク \***

"偽"の玉が箱（ノード）に入った状態のことを、**マスク**がかかっていると言います。

その箱には二度と"真"の玉が入ることはないからです。

各箱（ノード）に"真"が入るか"偽"が入るかを決めるのが数独問題を解く作業ということになります。

**\* コア \***

上に述べたように、あるピアで８つの箱に“偽”の玉がはいることによって、残りのノード（箱）に"真"の玉が入れば、そのノードにつながる残り３つのピア内の残り２４個の箱は全て"偽"の玉が入り、マスクがかかった状態になります。

しかし、あるノードが"真"である事が確定出来なくても、ある範囲のノードのどれかが"真"であるとわかれば、

その範囲のノード以外に、同じピア内にある空ノードに"偽"の玉を入れる（マスクする）事が出来ます。

この"ある範囲のノード"のことを"**コア**"と呼ぶことにします。

この説明だけではわかりにくいと思いますので、以降のセクションで実例を見ながら説明します。

**2. 数独の解法と三次元キューブとの関係**

(1) **単独未確定**

何度も言いますが、あるピアの９つのノード（箱）に"偽"の玉が８つ既に入っている場合、残りの一つの箱には必ず"真"が入って、元の9×9の二次元の問題に置き換えると、一つのマスの数字が確定します。すると、この”真”のノードに連動する残り三つのピアの全ノードにも”偽”の玉が入ることになります。ほとんどの数独問題はマスクノード(“偽”の玉が入っているノード)を増やしていって最終的に**単独未確定**の形に持っていって数字を決めていくという手順になります。

まれに違う方法が必要になる時もありますが。

(2) **座席予約**

単独未確定法で決まる数字が見つからなくなった時、次に易しいのがこの方法です。

前に説明した"コア"を見つける方法の一つです。

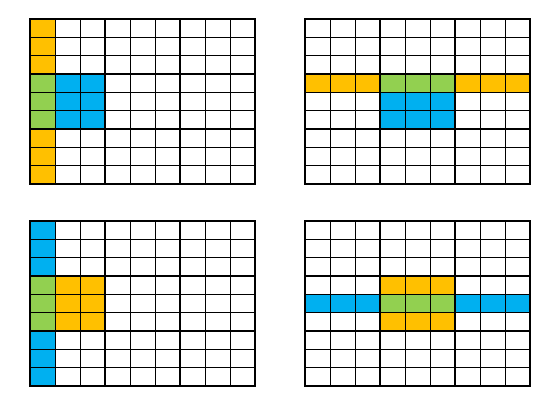
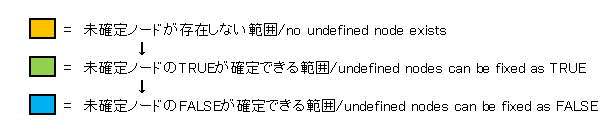
(Fig.C 座席予約)

ブロック対列

ブロック対行

列・行→ブロック

ブロック→列・行



i-j面で考える解法です（ある一つの数字で考えるということになります）。ブロックと行、またはブロックと列で、必ず重なるノードが三つありますが、この三つの重なるノードがコアになる可能性があります。（Fig.Cで緑のノード）

例えば、この重なっている三つのノードが２〜３個"空"（未確定）であるとします。

同じ行または列内の他のノードは全て"偽"だとすると、"真"の玉はこの２〜３個のノードのどれかに必ず入ることになります。

すると、この"コアノード"と同じブロックピアにある空（未確定）ノードには"偽"の玉が入ることになります。（Fig.Cで「列・行→ブロック」と表示）

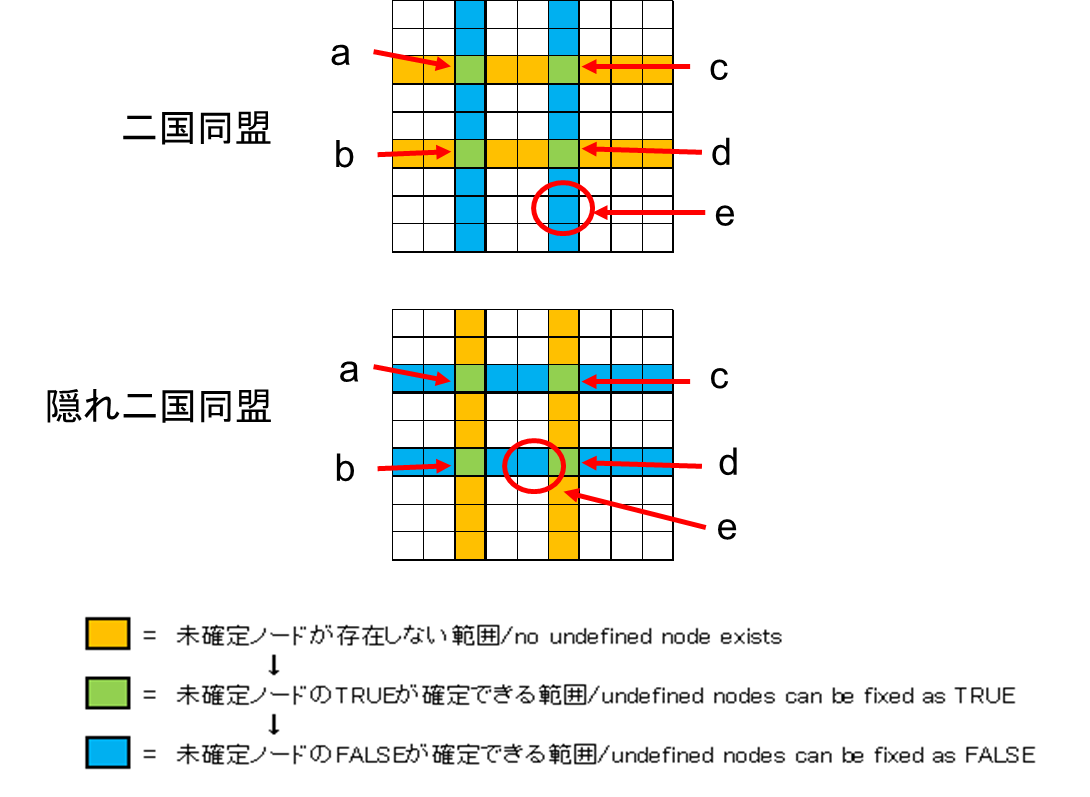
この緑のノードのグループを**座席予約**されていると表現します。

ブロックと行、ブロックと列を入れ替えて考えても同じ理屈が成立します。

つまり、あるブロックピア内で、ブロックピアと重なる三つのノード以外に空（未確定）ノードが存在しない時(座席予約されている)、この三つのノードと同じ行ピアまたは列ピア内の未確定ノードは全て"偽"の玉が入ります。（Fig.Cで「ブロック→列・行」と表示）

(3) **N国同盟と隠れN国同盟**

(Fig.D 二国同盟と隠れ二国同盟)



i-k面、またはj-k面、またはブロック-k面で考える解法です。N個の数字に着目します。

Nは２〜８まで可能性がありますが、考え方は同じですので、N=2（二国同盟）で説明します。

図のように四つの空ノード（緑）が配置されているとします。ノードa,b,c,d は必ず四角形に配置されています。

四本のピアが井桁状に重なっているわけです。

**＊二国同盟 ＊**

横二本のピア（青）で考えると、a,b,c,d以外に空ノードがないわけですから、ノードa が"真"ならノードd が"真"、ノードb が"真"ならノードc が"真"であるという論理が成立します。一つのピアには一つの"真"の玉しか入らないというルールを思い出して下さい。

この４つのノード以外には"真"が入らないということになり、ノードa~d が**"コア"**ということになります。

するとノードc,dと同じピアにあるノードe は"偽"ということになります。

通常これは"**二国同盟**"と呼ばれる解法になります。

**＊隠れ二国同盟 ＊**

逆にコアから見てタテ方向にマスクが発生する場合は”**隠れ二国同盟”**ということになります。（Fig.Dの下のほう）

**＊N 国同盟と隠れN国同盟 ＊**

Fig.E 三国同盟の場合



Nが３以上の場合も考え方は二国同盟または隠れ二国同盟と同じです。

ただ必ずしもコアノードの数がN×Nでなくても、Fig.Eの右側の図のように赤丸の部分が“空”でなく“偽”が入っていてもかまいません。”真”がN個配置できるような形であればよいということです。

ソースコードの中ではN国同盟も隠れN国同盟も四辺形の定理（行、列）も同じ関数を使っています。

基本の関数はi-k面での隠れN国同盟で作って、その他の面での隠れを含むN国同盟のコアを探すときは三次元キューブを回転してi-k面の隠れN国同盟になるようにし、コアが見つかってから三次元キューブを元に戻すという手法を使っています。

Fig.F 三次元キューブの回転と各解法の関係

Fig.F-1 N国同盟（列）

**基本形：隠れN国同（列）盟）**

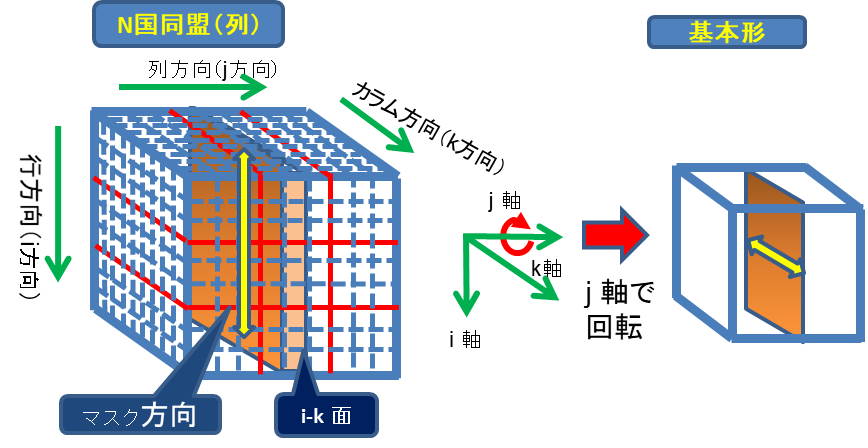
列方向（j方向）

カラム方向（k方向）

**i-k 面**

行方向（i方向）

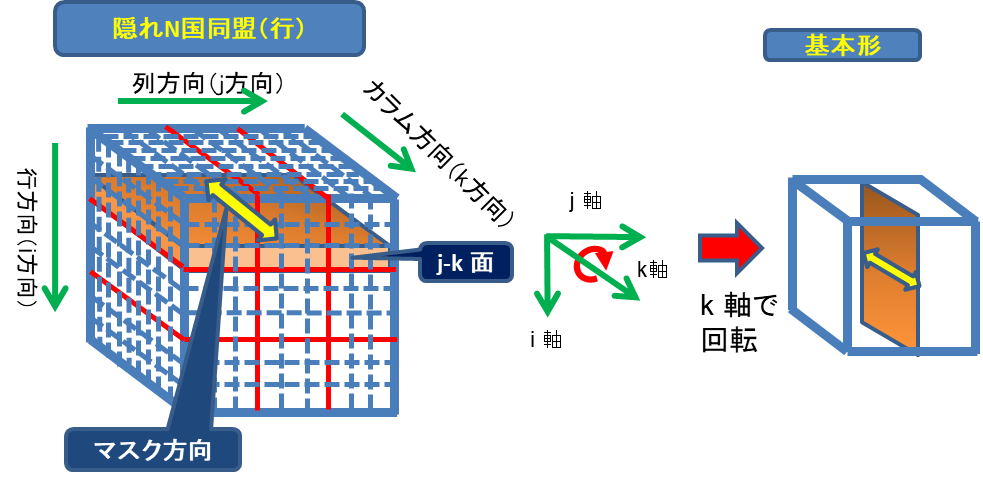
**マスク方向**



例えばFig.F－１のように、”N国同盟(列)”を基本形の”隠れN国同盟(列)”に変換するにはｊ軸で90度回転すれば良いことになります。

その他のケースは下記の通りです。

Fig.F-2　隠れN国同盟（行）　🡪 基本形



N国同盟（行）を基本形にするには、まずk軸で回転してN国同盟(列)にする必要があります。

Fig.F-3 N国同盟（行）🡪 基本形

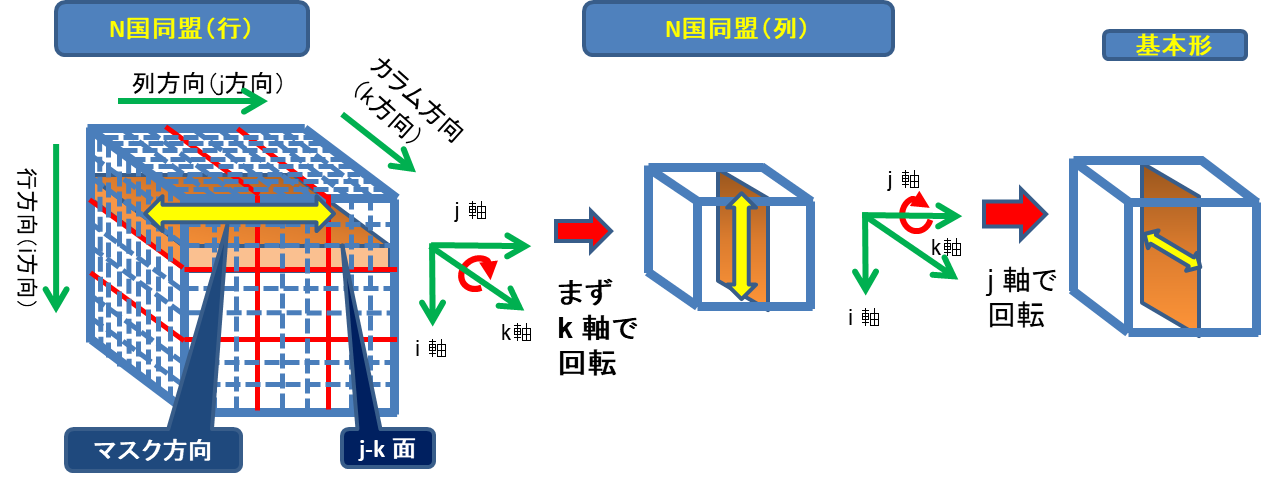
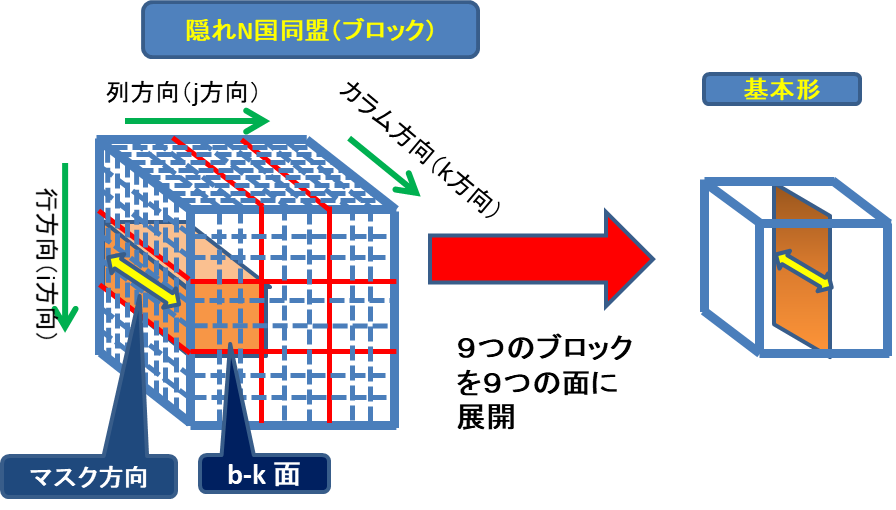


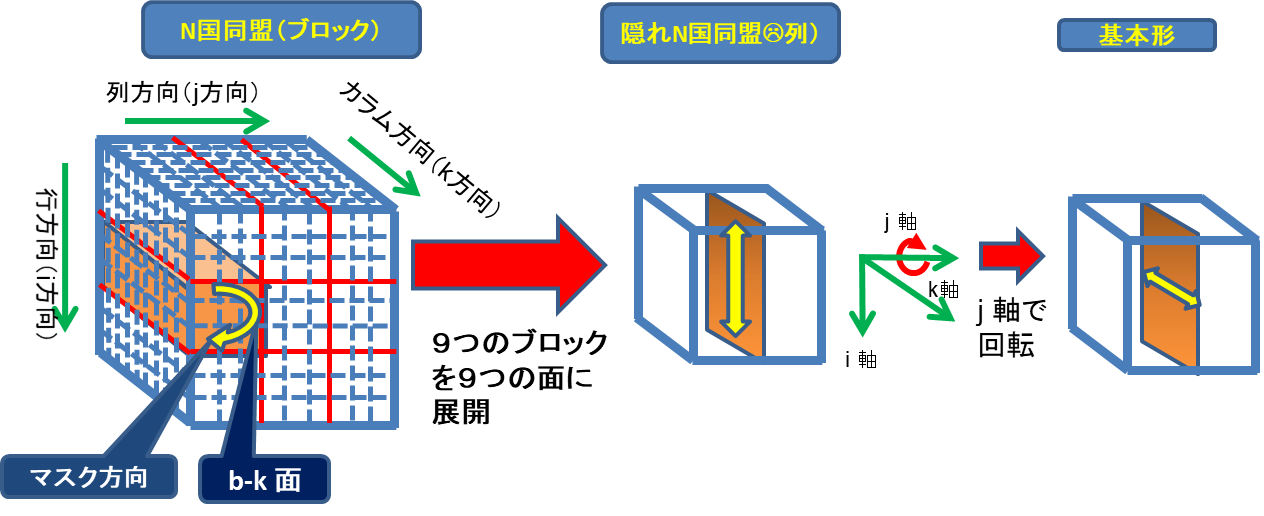
Fig.F-４ 隠れN国同盟（ブロック） 🡪 基本形



３×３の９つのブロックを１×９の面に展開します。水平に展開してもさしつかえありませんが、基本形にするにはFig.F-2のように回転が必要になります。

面の順番はどうやっても構いません。

Fig.F-5 N国同盟(ブロック) 🡪 基本形

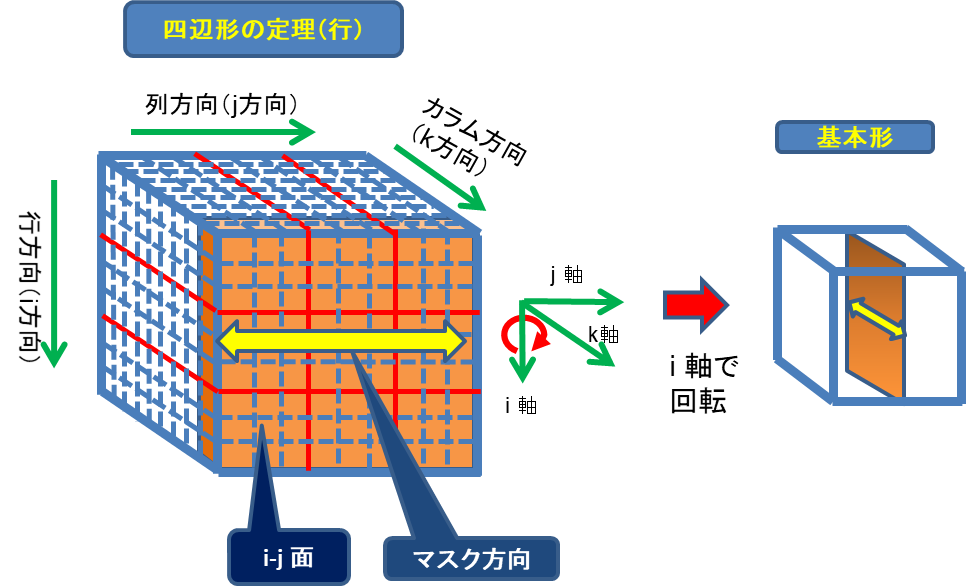


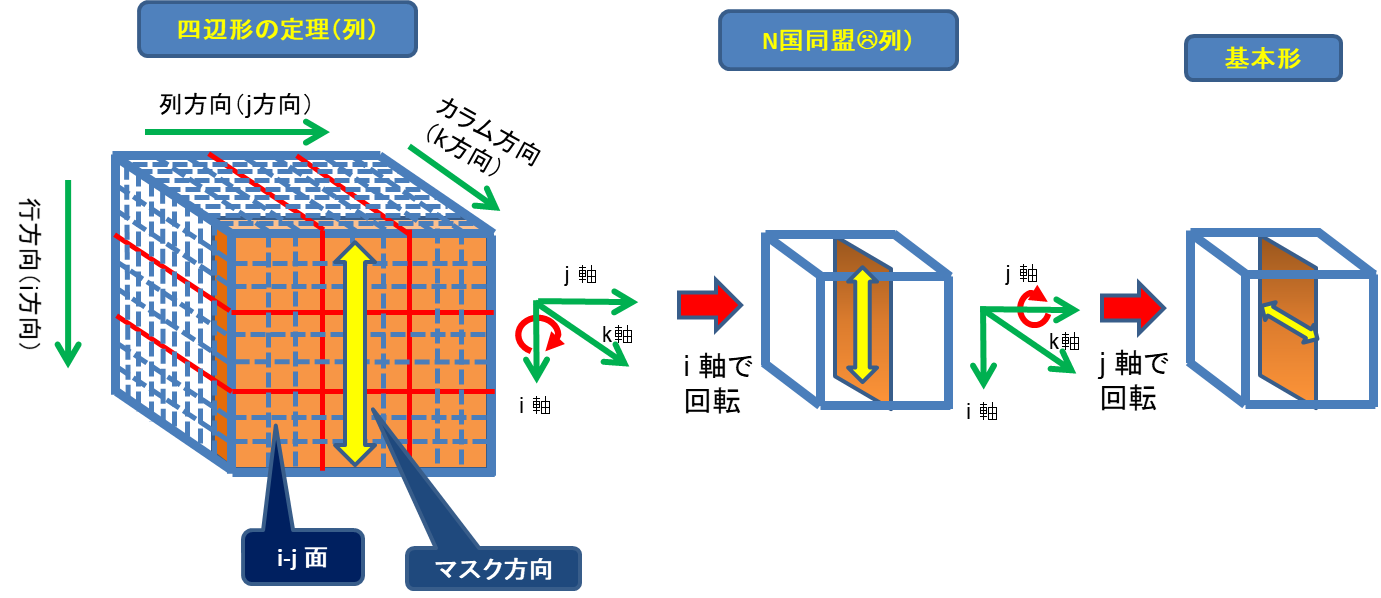
N国同盟（ブロック）を基本形にするには、ブロックを面に展開した後J 軸で回転する必要があります。

(4) 四辺形の定理（行と列）

Fig.F-6のように三次元キューブを回転させると上記の各N国同盟、隠れ二国同盟と同じ解法になります。

Fig.F-6 四辺形の定理（行、列）



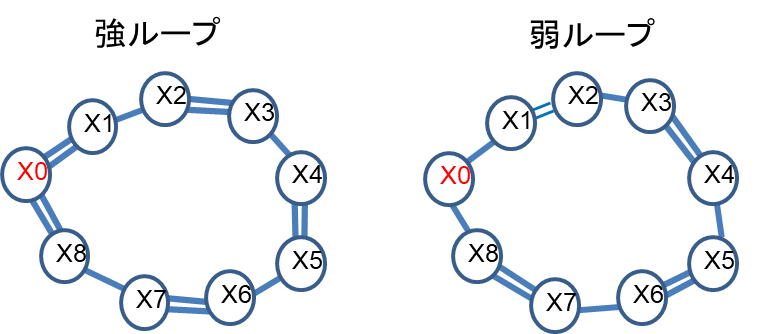


四辺形の定理もN×Nのコアによる解法で説明できます。

Nが３のときは”ソードフィッシュ”、４以上になると”ホエール”とか”リバイアサン”とか呼ばれることがあります。コアの形から来ていると思いますが、よくわかりません。いずれも海洋生物の名前です。リバイアサンは海に住む空想上の怪物ですね。

(5) 強ループ法と弱ループ法

Fig.G 強ループと弱ループ



今までの解法は三次元キューブに存在する３６面の内、一つだけを使う解法でした。

強ループ法と弱ループ法は複数の面にまたがる解法になります。

図でマル（○ ）は一つの空ノード、二本線（＝）は両端のノードが強リンクであるということを、一本線（ー）は両端のノードが弱リンクであるということを表しています。

当然、二本線または一本線の両端のノードは同じピアにあリます。

強リンクとは、一方のノードが"真"だと、他方のノードは"偽"になり、一方のノードが"偽"だと他方のノードは"真"になる

という関係のことです。言い換えると、この二つのノードがあるピアには他に空ノードが無いということです。

弱リンクは逆に一本線の両側のノード以外に空ノードが同じピア内にあるということです。

つまり、一方のノードが"真"だと他方のノードは"偽"になりますが、一方のノードが"偽"のとき、他方のノードは必ずしも"真"にはならないということです。具体的には同じピア内に空ノードが三つ以上あるケースです。

私はX0のノードをターゲットノードと呼んでいます。

このターゲットノードの両端が強リンクになっているループを強ループ、両端が弱リンクになっているループを弱ループと呼んでいます。

このターゲットノードにつながる二つのノード以降は強リンクと弱リンクが交互に現れています。

なお弱リンクは強リンクと置き換え可能ですが、強リンクの代わりに弱リンクを持ってくることはできません。

強ループのターゲットノードは必ず”真”になり、数字が確定できます。

弱ループのターゲットノードは必ず”偽”になります。数字の確定はできませんが、マスクノードとすることができます。

この理屈は以降で説明します。そういうものだと思ってもらえる方は読み飛ばしてください。

**\* 強ループの性質**

Fig.H 強ループの例

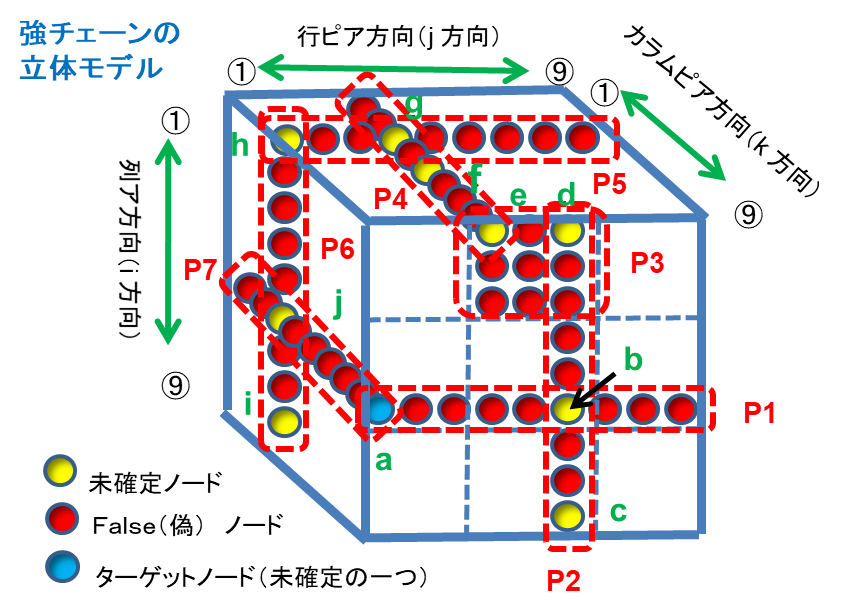
****

Fig.Hの例では連動するピアがP1からP7ばであり、ピアの交差するノードが強ループを形成しています。”真”となるノードを”1”､他のノード（“偽”）を”0”に置き換えて考えると、数独のルールから一つのピアの合計は必ず”1”になります。

するとFig.Hの場合、下記の式が成り立ちます。

P1 a + b = 1

P2 b + c + d = 1

P3 d + e = 1

P4 e + f + g = 1

P5 g + h = 1

P6 h + i + j = 1

P7 a + j = 1

合計 2a +2b +c +2d +2e + f+2g+2h + i + 2j = 7

↓

**2(a+b+d+e+g+h+j)+c+f+i = 7**

ここで、a+b=1,d+e=1,g+h=1は確定していますから、この式は下記のようになります。

**2 ( 3 + j ) + e + f + i = 7**

そうすると、もしj が”1”なら2(3+j)だけで8 になってしまい、合計を 7 にすることができなくなってしまいます。したがって、j は”0”でなくてはならず、P7の式から a は”1”（真）という理論が成り立ちます。また上の式の右辺を７にするには、e,f,iのどれか一つは必ず”1”である必要があります。この点は今のところ重要ではありませんが、数独問題を解くヒントになることがあるかもしれません。

以上のことは下記のように考えてもOKです。

**ノードaが"偽"**と仮定すると、

↓

ノードbは"真"

↓

ノードcと d は"偽"

↓

ノードeは"真"

↓

ノードfと g は"偽"

↓

ノードhは"真"

↓

ノードiと j は"偽"

↓

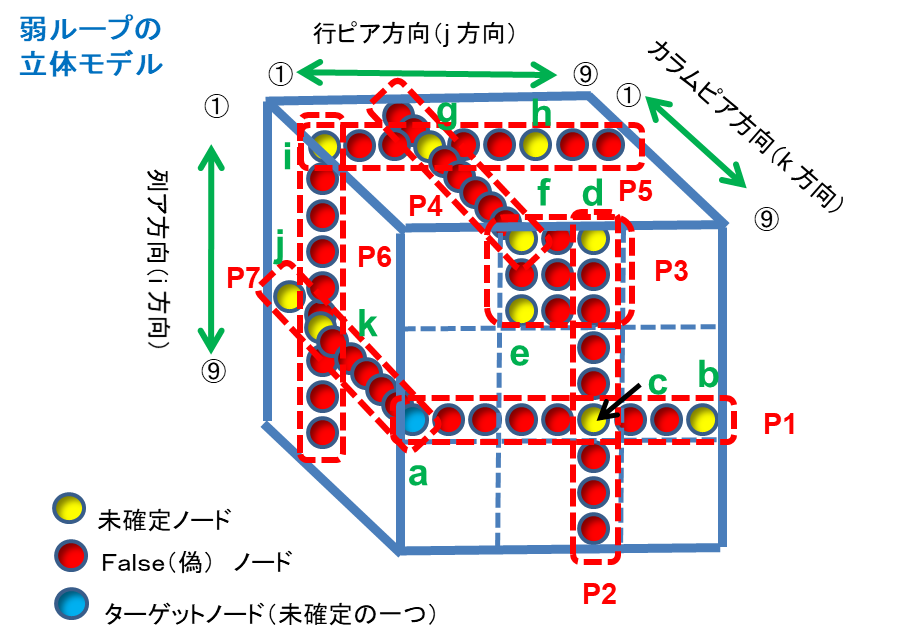
ノード a は”真”

となり、最初の仮定であるノード a は”偽”という仮定が間違っていることになり、

ノードaは"真"ということになります。

\* **弱ループの性質**

Fig.H 弱ループの例



弱ループの場合も下記のように同じやり方でターゲットノードが”偽”であることが証明できます。

P1 a + b + c = 1

P2 c + d = 1

P3 d + e + f = 1

P4 f + g = 1

P5 g + h + i = 1

P6 i + j = 1

P7 a + j + k = 1

合計 2a +b +2c +2d +e +2f+2g+ h +2i +2j + k = 7

↓

**2(a+c+d+f+g+i+j)+b+e+h＋k = 7**

ここで、c+d=1,f+g=1,i+j=1は確定していますから、この式は下記のようになります。

**2 ( a + 3 ) + b + e + h + k = 7**

そうすると、もしa が”1”なら2(a+3)だけで8 になってしまい、合計を 7 にすることができなくなってしまいます。したがって、a は”0”となります。また、b,e,h,k のどれか一つはかならず”1”になります。

これは強ループと同様下記のように考えてもOKです。

**ノードaが"真"**と仮定すると、

↓

ノードbと c は"偽"

↓

ノード d は"真"

↓

ノードeと f は"偽"

↓

ノード g は"真"

↓

ノードhと iは"偽"

↓

ノード j は"真"

↓

ノード a は”偽”

となり、最初の仮定と矛盾します。

従ってノードaは"偽"ということになります。

強ループでも弱ループでも、ループを構成するノードが何個あっても同じ理論が適用できます。ループを構成するノード数は必ず奇数になります。

ただし現実的にループが大きくなると人間の頭ではループの発見が難しくなるので、

本システムのソースコードはノード数が７つのループまでしか検出できません。

ただしコードを改造すればノード数を増やすことは可能です。

その場合ノード数は２つずつ増えていきます。3 🡪 5 🡪 7 🡪 9 🡪 11 ....といった具合です。

**3. 数独システムのヒント画面の見方**

1. 基本画面とヒント画面の関係

　数独システムのヒント画面は三次元キューブでのコアノード情報を二次元に置き換えたものになっています。

Fig.I-1 数独問題画面

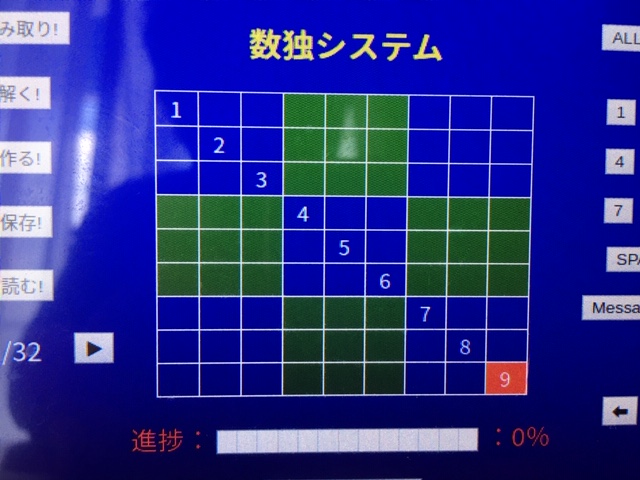
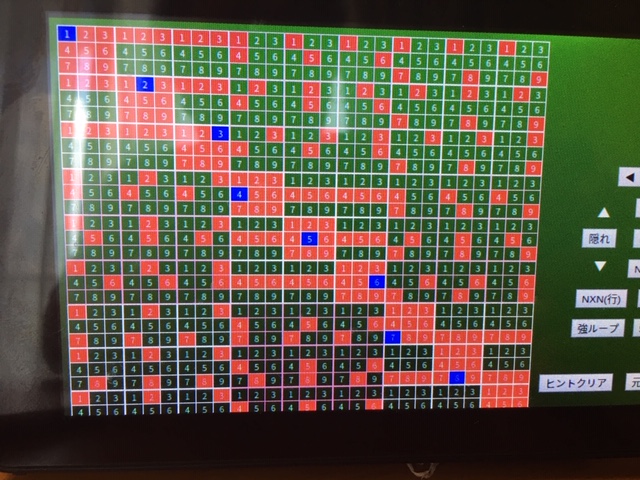


Fig.I-1のような問題はありませんが、わかりやすくするために、このように数字が入るとすると、Fig.I-2のようなヒント画面になります。

Fig.I-2 Fig.I-1のヒント画面



三次元情報を二次元で表現しているのでわかりにくいですが、数独問題の9×9のマスの一つ一つに3×3=9個のノードが表現されています。青のノードは”真”のノードを表しています。Fig.I-1で左上の画面の数字は”1”ですから、Fig.I-2では左上の9個のマスのうち、”1”が青になっているわけです。２～９まで同様であることを確認してください。”9”は画面からはずれていて見えませんが、実際の画面ではスクロールして見ることができます。

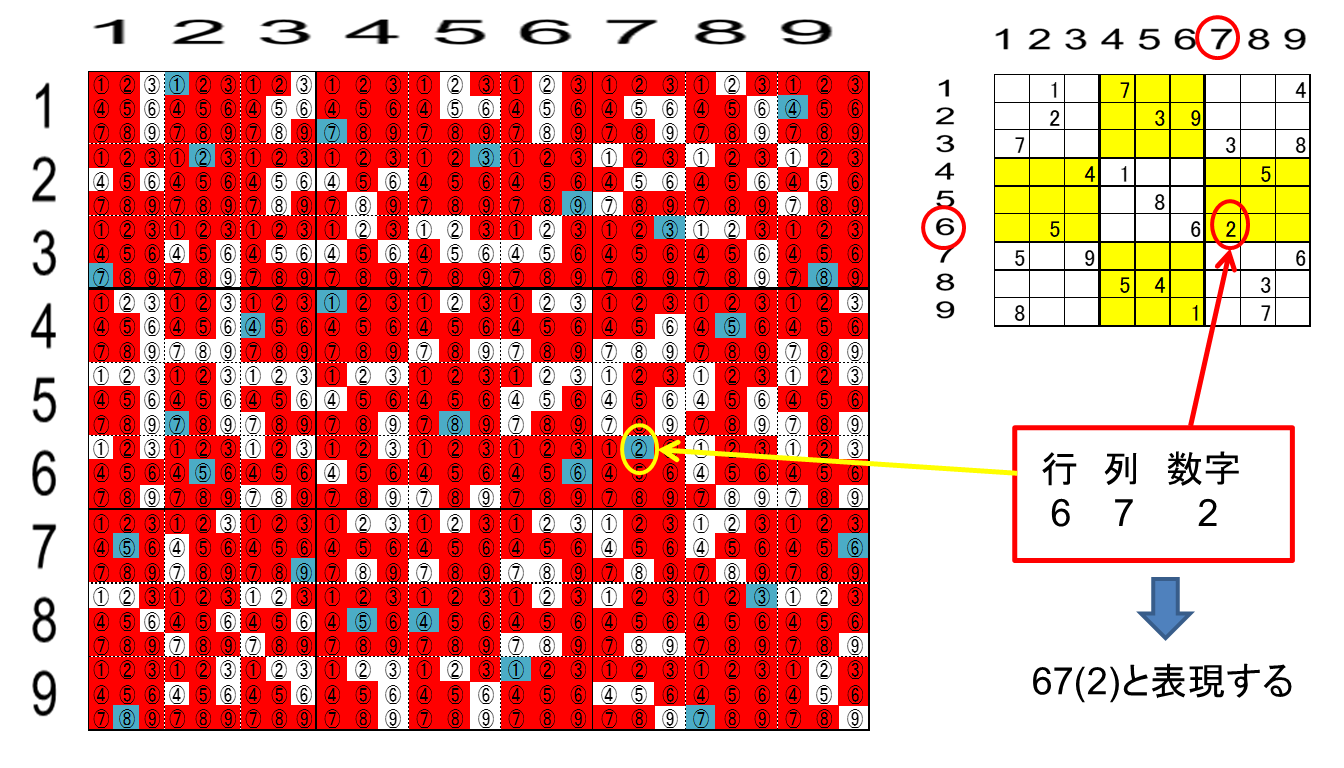
赤のノードはマスクがかかっている”偽”のノードです。”真”のノードによって連動する３つのピアのすべてのノードが赤になっています。

背景色（緑）のままのノードはまだ”真”か”偽“か不確定ということです。数独の問題のヒントはすべてこの緑の不確定ノードの配置から導き出されます。

1. ノードの位置の表し方

まず各マス（ノード）の位置を表現する方法を定義します。このテキストでは下記のように定義することにします。

Fig.J 各マスの位置の表現



1. ヒント画面の使い方

Fig.Jの問題を例に説明します。まず基本画面で右側の数独問題を入力してください。

入力したら[問題を保存する!]を押して保存しておいてください。保存した番号が表示されますので覚えておいてください。途中行き詰まったら番号を選んで[問題を読む!]を押せば最初からやり直すことができます。

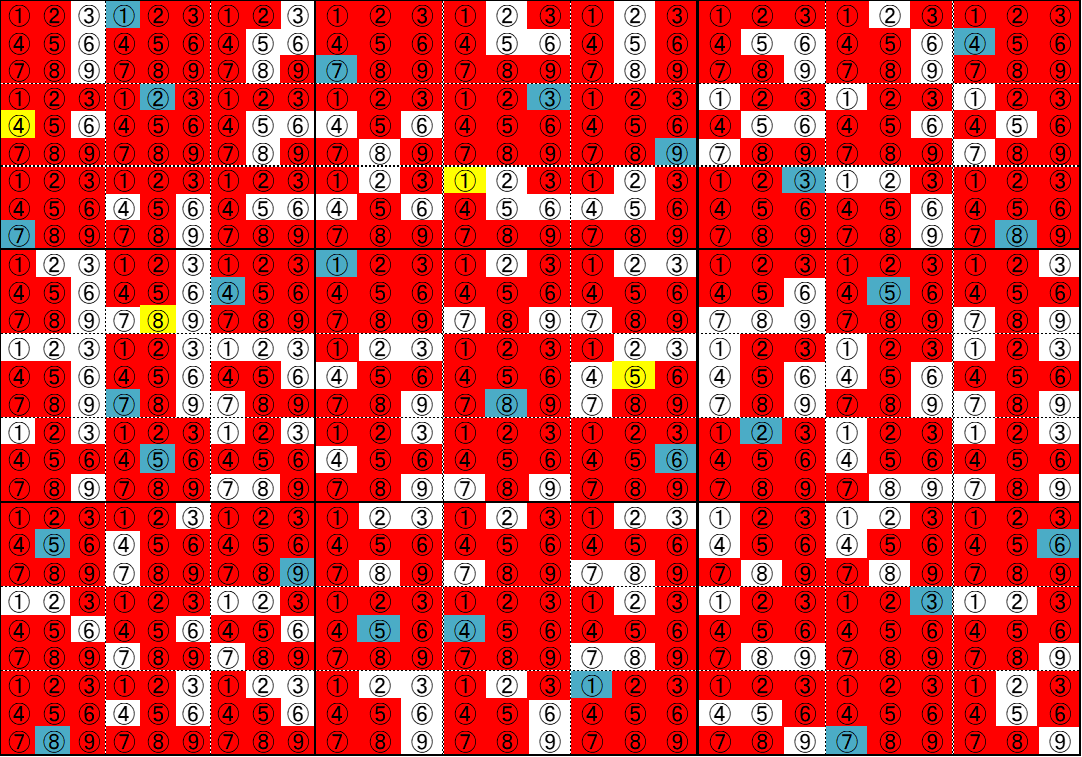
上級者ならFig.Jの情報から最後まで解き進めることは出来ますが、ここではヒント画面を使いながら説明します。

<step-1> 単独不確定ノードを見つける

ヒント画面で[単独不確定]のボタンを押してください。すると、ボタンが赤くなって

Fig.K-1のような画面が表示されるはずです。

Fig.K-1 単独不確定ノードのヒント画面(1)



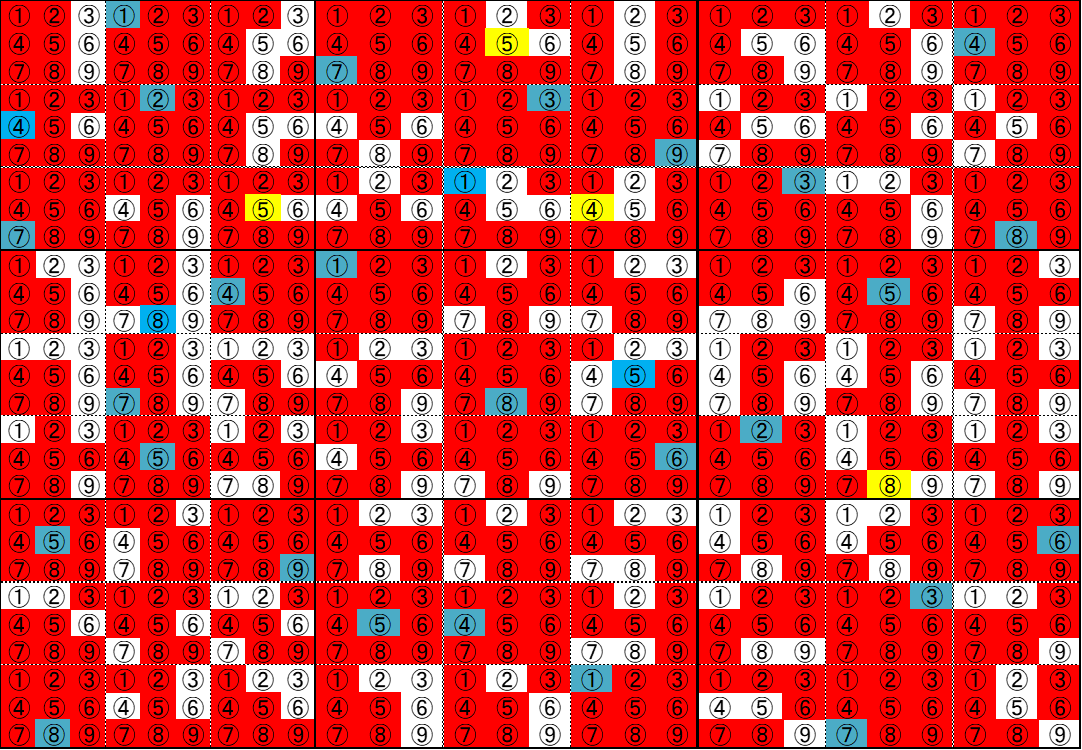
次の４つのノードが黄色で表示されるはずです。

21(4) 35(1) 42(8) 56(5)

基本画面に戻って上の４つの数字を入力後、またヒント画面に戻ってください。

Fig.K-2 のようなヒント画面に変わっているはずです。[単独不確定]ボタンが白くなっていたらもう一度押して赤くしてください。

Fig.K-2 単独不確定ノードのヒント画面(2)



今度は次の４つのノードが黄色で表示されるはずです。

15(5) 33(5) 36(4) 68(8)

基本画面に戻って上の４つの数字を入力後、またヒント画面に戻ってください。

Fig.K-3 のようなヒント画面に変わっているはずです。[単独不確定]ボタンが白くなっていたらもう一度押して赤くしてください。

今度は次の２つのノードが黄色で表示されるはずです。

64(4) 95(6)

同様に基本画面に戻って上の２つの数字を入力後、またヒント画面に戻ってください。

Fig.K-4 のようなヒント画面に変わっているはずです。[単独不確定]ボタンが白くなっていたらもう一度押して赤くしてください。

今度は次の1つのノードだけが黄色で表示されるはずです。

94(9)

また基本画面に戻って上の数字を入力後、またヒント画面に戻ってください。

今度は[単独不確定]ボタンが赤くなっていてもヒントは表示されません。

現段階では単独不確定ノードは存在しないということです。

Fig.K-3 単独不確定ノードのヒント画面(3)

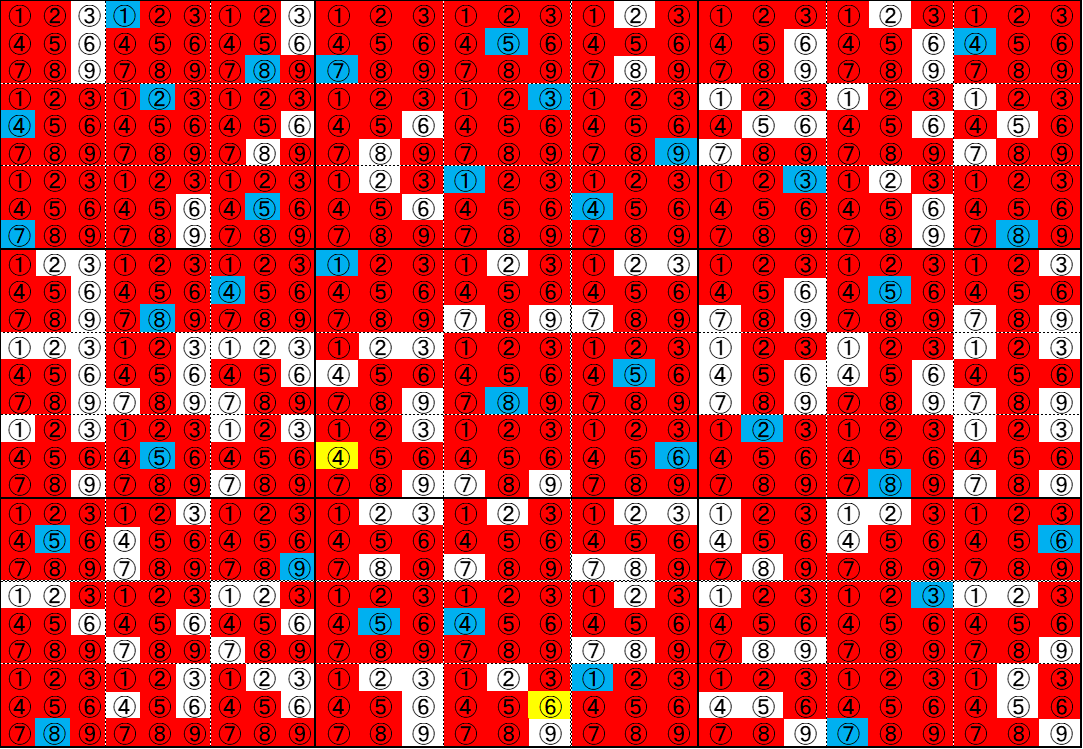
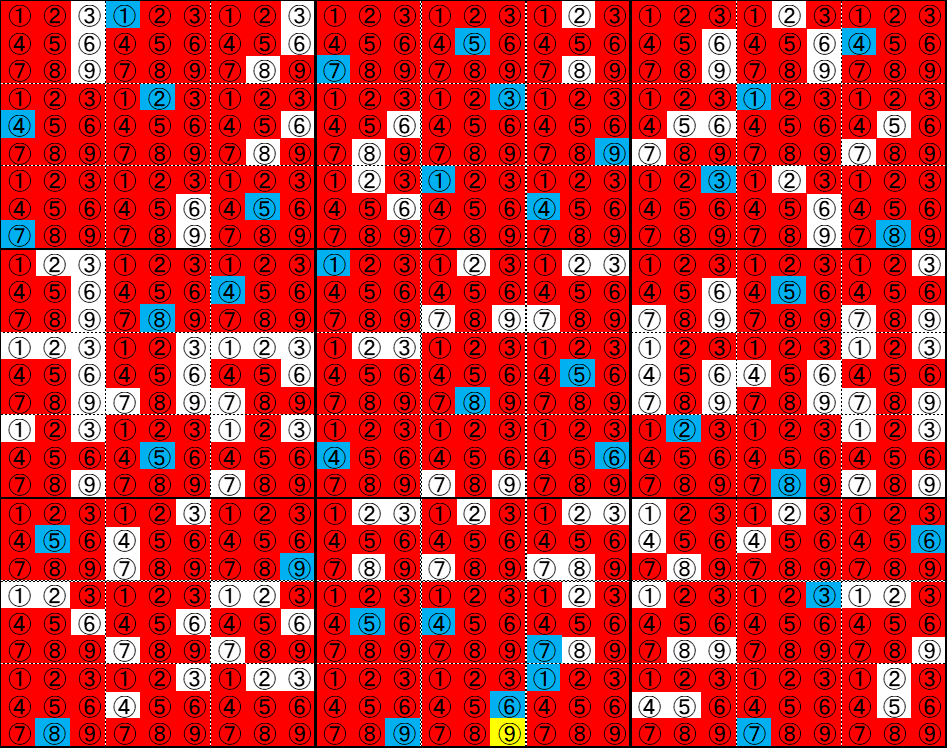


Fig.K-4 単独不確定ノードのヒント画面(4)

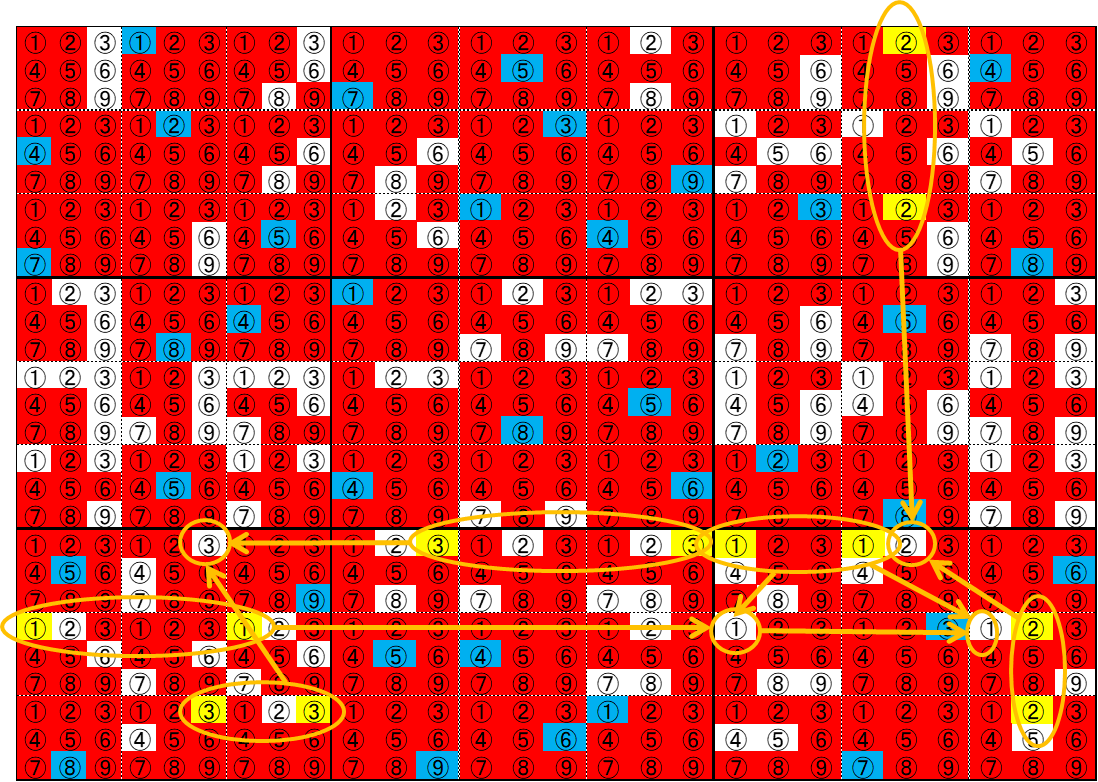


<step-2> 座席予約によるマスクノードを見つける

Step-1が終わった状態のヒント画面で[座席予約]のボタンを押してください。すると、ボタンが赤くなってFig.L-1のような画面が表示されるはずです。

座席予約によるコアノードによって下記のマスクノードが確定します。

Fig.L-1 座席予約のヒント画面(1)



＊座席予約によるコアノード ＊コアノードによるマスクノード

18(2) 38(2) 78(2)

74(3) 76(3) 72(3)

92(3) 93(3) 72(3)

77(1) 78(1) 87(1) 89(1)

81(1) 83(1) 87(1) 89(1)

89(2) 99(2) 78(2)

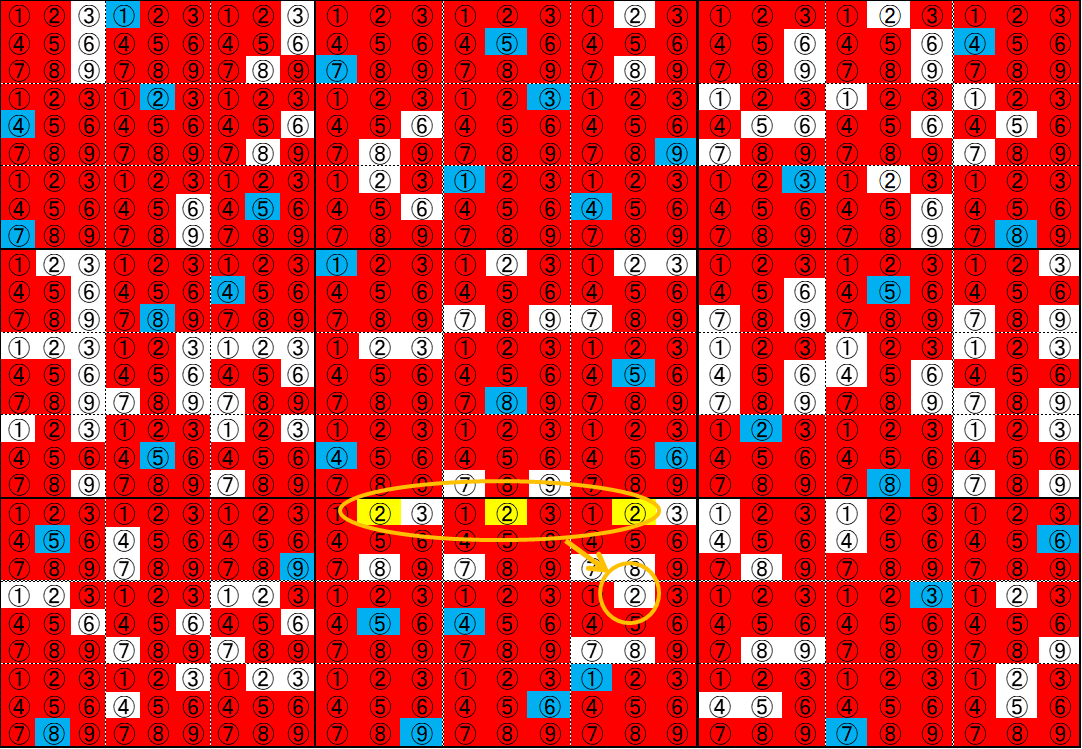
ダブっているマスクもあるので、結局下記４つのノードがマスク確定します。

87(1) 89(1) 78(2) 72(3)

上のマスク情報だけでは新たな単独不確定ノードは発生しませんので、次のステップ以降は単独不確定ノードが発生するまで紙の上でマスク追加を進めることになります。この点は次回の数独システムのバージョンアップで解決したいと考えています。

上記のマスクが確定すると、今度はFig.L-2のように別の座席予約ノードが現れてきます。

Fig.L-2 座席予約のヒント画面(2)



＊座席予約によるコアノード ＊コアノードによるマスクノード

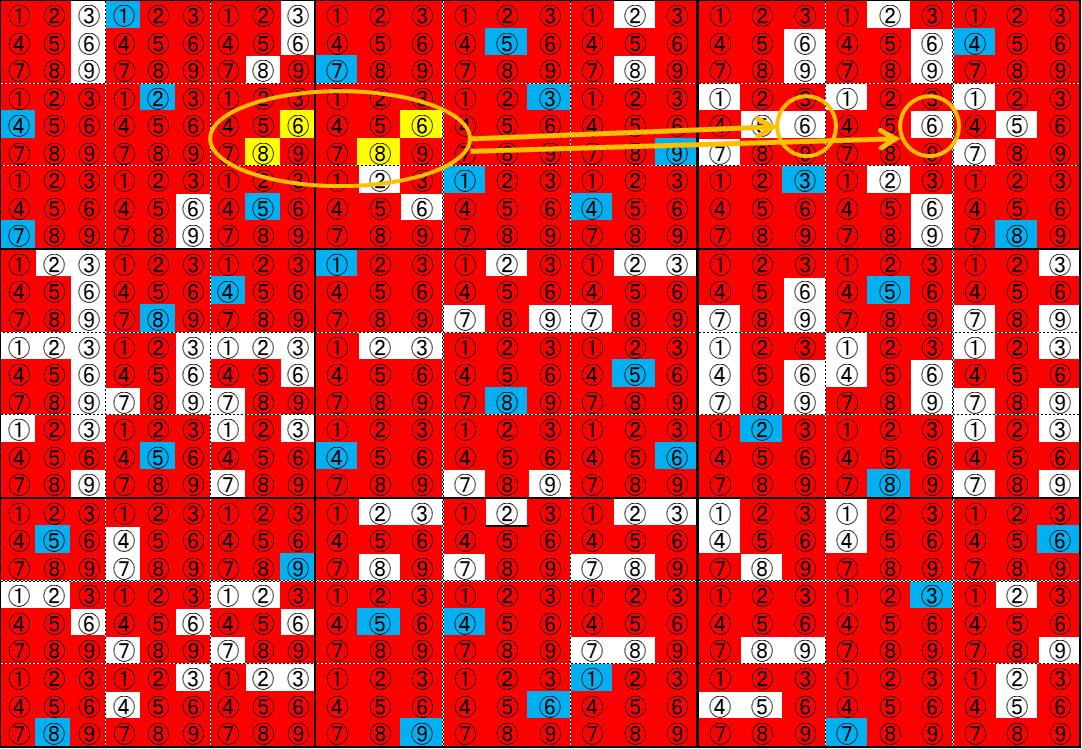
74(2) 75(2) 76(2) 86(2)

これで座席予約は無くなりましたので、次の解法を探します。

<step-3> 二国同盟によるマスクノードを見つける

今までのマスクノードをすべて適用すると、Fig.-Mのような二国同盟のコアが確認できると思います。

Fig.M二国同盟によるマスクノード



＊二国同盟によるコアノード ＊コアノードによるマスクノード

23(6) 23(8) 24(6) 24(8) 27(6) 28(6)

するとノード28(1)が単独未確定ノードになることがわかります。

ここからまた数独システムに戻ります。

<step-4> 再び単独未確定ノードを探す

今後は下記のようにドミノ式に単独未確定ノードが次々に現れて最後まで到達できます。

最初の数独システムのヒント画面の使い方を参考に解答を進めてください。

ドミノ回数 単独未確定ノード

1. 28(1)
2. 77(1)
3. 87(8)
4. 89(9)
5. 99(2)
6. 29(5) 78(4) 93(3) 97(5)
7. 11(3) 27(7) 52(3) 57(4) 72(7) 92(4)
8. 32(9) 46(3) 54(2) 69(3) 74(3) 75(2) 82(6) 86(7)
9. 16(2) 24(8) 34(6) 38(2) 41(2) 49(7) 59(1) 76(8) 83(2)
10. 13(8) 23(6) 45(9) 47(6) 51(6) 53(7) 63(1) 65(7) 81(1)
11. 17(9) 18(6) 58(9) 61(9)

以上のように一つの解法のコアノードによるマスクだけでは単独未確定ノードが発見できない場合があり、複数の解法のドミノ倒しが必要な場合があります。

ただ新聞や雑誌に掲載されている問題は単独未確定を探していけば解ける問題がほとんどです。座席予約や二国同盟が必要な場合もありますが、上級者向けの雑誌に限られます。

でも数独システムを活用していただければ９９％の問題は解けるはずです。

以上