卒論準備　最終更新日2022/01/19

**プログラム作成の目的**

人が数独を解く様子をプログラムで再現し、人が数独を解くうえで何か役立つことを発見したい。

例えば、簡単な方法だけでどのような難易度の問題も解くことが出来ることが分かれば、パズルを解いたことがない人が取り組みやすくなる。また、わざわざ難しい解き方を習得する必要がなくなる。

ここでの簡単な方法は数独を解くにあたって第一選択肢として考えられる基本的な解き方とする。

また、簡単な方法の繰り返しでどこまでパズルを解くことが出来るか、もしそれで答えにたどり着くことが出来なければ、どのような工夫をすれば最後まで解くことが出来るかを考察したい。

**用いたパズル、手段の説明**

数独の定義

　3×3のグループに区切られた9×9の正方形の枠内に1~9までの数字を入れるパズル。

縦方向の9マスを行、横方向の9マスを列と呼ぶ。

このパズルでは行、列、グループ、それぞれの中に同じ数字が複数入ってはいけないという制約がある。

盤面：パズルの表のこと

数独のルールを満たすとは以下のことを指す。

数独の問題例は整数 0, 1, … , 9を要素とする長さ9の列Rの9個組Qである。

各行は について,

各列は について,

9×9の盤面を3×3ごとに区切ったときに出来る計9個の部分をグループとすると、

各グループは について, (g mod3) としたとき、

となる。

また、各行、各列、各グループの9マスをここではエリアと呼ぶ。

以上の27つの9個組すべてにおいて、1から9までの数字が高々1回出現することが数独のルールとなる。

問題例の定義

　　整数0,1,…,9の9個の列の9個組ならびに

で数独のルールを満たすもの。

解の定義

　　整数1,2,…,9のいずれかの9個の列の9個組ならびにで、

　任意の に対して Q[i][j] > 0 のとき Q[i][j] = A[i][j]であり、かつ数独のルールを満たすもの。

基本テクニック1. エリア確定法

与えられた盤面の各エリアの空きマスの数に注目する。

それぞれのエリアの中で空きマスが1つだけである場合、他のマスで使われている数字から空きマスの数字を確定することが出来る。エリア内に存在する数字は制約によって空きマスに入れることが出来ないため、それらを除いた残り1つが空きマスに入る。このように盤面すべてのエリアの空きマスの数を調べ、空きマスが1つだけのエリアを完成させることで盤面を解き進めていく方法。

基本テクニック2. 可能性確定法

与えられた盤面の空きマスに入る可能性のある数字の数に注目する。空きマスが含まれるエリアを調べることで空きマスに入る可能性がある数字を特定することが出来る。この方法ははある空きマスを含むエリアが複数ある場合に使われる(1つしかない場合はエリア確定法が使われるため)。エリア内に存在する数字は制約によって空きマスに入れることが出来ないため、1~9の中でそれらを除いた数字が空きマスに入る可能性となる。このように空きマスの可能性を調べ、その可能性が1つだけであれば、その数字がマスに入る。この操作を全ての空きマスに対して行い盤面を解き進めていく方法。

基本テクニックの利点と欠点

基本テクニックは数独のルールから明らかで、誰でも簡単に理解出来るものである。つまり数独を解く第一選択肢の手段としてふさわしい。また、他の空きマスの可能性を記憶せずに使うことが出来る。つまり記憶領域の負担が少ない。ただ利点だけではなく欠点もある。それは難易度によるが、それだけでは解にたどり着けないことが多いことだ。そこでこれを幅優先探索、数独でいう総当たり法を使うことで解消した。総当たり法も基本テクニックと同様に誰でも簡単に理解できる数独の解法の1つである。これだけを使って数独を解くことは現実的ではないが基本テクニックが使えない場面に対して部分的に使うことで現実的に使えるような工夫をした。しかしこの方法で盤面に入れる数字は必ずしも正解ではなく、進めた結果不正解で盤面を戻して数字を入れなおす必要がある。これは利点である記憶領域の負担の少なさが失われてしまうことになるが、目的を達成するためには致し方無い。

幅優先探索

幅優先探索はグラフ理論においてグラフの探索に用いられるアルゴリズムである。アルゴリズムは特定のノードで探索を開始し、そこから隣接する未訪問のノード全てを探索する。そしてこれらの探索したノードのそれぞれに対して同様のことを繰り返す。始点から距離が近い順に探索をしていくため、探索するノードをキューを使って管理する。

幅優先探索の疑似コードは以下のようになる。

1. 始点のノードを探索待ちキューに追加する。
2. 探索待ちキューにノードがあれば取り出す。なければ探索終了。
3. 取り出したノードが目的ノードであれば探索終了。
4. 取り出したノードに隣接するノードの内、未探索のノードを探索待ちキューに追加する。
5. 2の処理に戻る。

数独における幅優先探索では木としてすべてのノードを走査する。与えられた問題例を根ノードとし、各ノードが表す盤面の空きマスを1つ埋めた状態をその子ノードとする。

探索では各ノードに対して、作成可能な子ノードすべてを探索対象にすることで盤面を埋めていき、最終的にすべてのマスに数字が入って、かつ制約を満たすノードを見つければ探索は終了となる。

”盤面を埋める”は幅優先探索の中で過程として空きマスに数字を入れて行くことを指す。結果的に間違った数字を埋める操作に対しても使う。”盤面を解き進める”は解となる数字を盤面に入れることを指す。

幅優先探索や深さ優先探索などの全探索のみを使用する方法では問題に対してすべての数字のパターンを入れていくため、数独では総当たり法という解き方にあたる。総当たり法は最もシンプルだが、計算量が非常に大きくなるため、これのみを使って解くことは現実的ではない。そこで今回は、基本テクニックでは解き進めることが出来ない盤面を進めるために総当たり法を用いる。ここで1つの盤面から多くのパターンの子ノードを得るために深さ優先探索でなく、幅優先探索を使用している。

幅優先探索をこのまま用いると空きマスに数字を入れる段階で制約を満たさないものまで入れてしまうと無駄な探索の枝が生じるので、制約を満たす数字のみ入れることで計算量を減らし使いやすくする工夫をした。

探索の中ですべての空きマスに対して制約を満たす数字が見つからない場合が出てくるが、それは葉ノードとなり、バックトラックをして探索を進めていく。

**プログラムの説明**

数独のsolver　solver()

ここから出てくる変数は以下の意味を表す。

　child : 子、幅優先探索を進めていくなかで注目される盤面

　children : 探索されていないchildを保存するリスト

1. 入力として盤面を与える.
2. まず入力の盤面に対してtechnique関数を使いパズルを進める。その後はこの進めた盤面を幅優先探索の根として探索を進めていく。(進めた盤面をchildrenに追加し、ループを始める。)
3. childrenの先頭から子を1つ取り出しこれをchildとする。
4. childにtechnique関数を使い盤面を解き進める。
5. 進めた盤面が正解であるかをanswer関数でチェックする。

(i)盤面が正解であれば探索を終了する。

1. childに対して1つ先までの幅優先探索を行う。(childの子を作成する。)

childの空きマスに入りうるすべての数字を1つだけいれる。

(childから1マスだけすすめた盤面を制約を満たす限りで全パターン作る)

数字を入れる際にchildの空きマスの部分を書き換えるが、1つの盤面に対して何度も書き換えを行う必要があるため、childのコピーを作成し、コピーを書き換えていくことで不具合なく次の子を作成することが出来る。

このとき、最終的に答えではなくなるようなものでもその時点で制約を満たしていれば子とする。作った子をchildrenに追加する。

1. childは探索済となるので、childrenから削除する。
2. aに戻る。

チェック関数 check()

行、列、グループ、それぞれに対してチェック関数を作成し、それをひとまとめにしたものをエリアのチェック関数とする。チェック関数は以下の手順で動く。

1.　入力として盤面、チェックしたい行、チェックしたい列、マスに入れたい数字を与える。

2.　入力した行、列から調べたいマスを特定する。

3.　そのマスに入れたい数字を入れたとき、数独の制約を満たすかどうかを真偽で返す。

　行でのチェック関数 row()

1. 入力として盤面、チェックしたい行、マスに入れたい数字を与える。
2. 調べたい行の全ての列の数字と盤面に入れたい数字を比較する
3. 同じ数字がなければ制約を満たすためTrueを返す。
4. 同じ数字があれば入れたい数字は制約を満たさないためFalseを返す。

　列でのチェック関数column()

1. 入力として盤面、チェックしたい列、マスに入れたい数字を与える。
2. 調べたい列の全ての行の数字と盤面に入れたい数字を比較する。
3. 同じ数字がなけば制約を満たすためTrueを返す。
4. 同じ数字があれば入れたい数字は制約を満たさないためFalseを返す。

　グループでのチェック関数 block()

1. 入力として盤面、チェックしたい行、チェックしたい列、マスに入れたい数字を与える。
2. 入力した行、列から調べたいマスを特定し、そのマスが所属するグループを調べる。
3. グループ内に入っている数字と盤面に入れたい数字を比較する。
4. 同じ数字がなければ制約を満たすためTrueを返す。
5. 同じ数字があれば、入れたい数字は制約を満たさないためFalseを返す。

テクニック1：エリア内確定法(各エリア内だけを見て数字を確定する方法) tech1\_solver()

行、列、グループそれぞれについてエリア確定法を行う関数を作成し、それらをまとめて全体のエリア確定法とする。エリア確定法は以下の手順で動く

　1. 入力としてパズルの盤面を与える。

2. 各エリアに対してエリア確定法を行い、1つでも盤面を解き進めることが出来ればもう一度エリア内確定法を行う。これを盤面を解き進めることが出来なくなるまで行う。

行でのエリア内確定法 tech1\_row()

1. 入力として盤面、調べたい行、マスに入れたい数字を与える。
2. 与えられた行の空きマスの数をカウントする。

(空きマスは盤面上では0で入っているため、０の数をカウントする。)

1. 空きマスが1つであれば、その行と各列についてチェック関数を使う。

(i)制約を満たす場合、入れたい数字をマスにいれ、Trueを返す。

(ii) 制約を満たさない場合、何も操作をせずFalseを返す。

　　　　　　（すでに盤面にマスが埋められている部分に対してこの操作が起こらないように

マスに数字を入れる前に空きマスであることを確認する。）

　　　(b)調べたい行の空きマスが1つでなければ何も操作をせずFalseを返す。

　列でのエリア確定法 tech1\_column()

1. 入力として盤面、調べたい列、マスに入れたい数字を与える。行の場合と同様に進めていく。
2. 与えられた列の空きマスの数をカウントする。
3. 空きマスが1つであれば、その列と各行についてチェック関数を使う。

(i) 制約を満たす場合、入れたい数字をマスにいれ、Trueを返す。

(ii) 制約を満たさない場合、何も操作をせずFalseを返す。

　　　 (b)調べたい列の空きマスが1つでなければ何も操作をせずFalseを返す。

　グループでのエリア確定法 tech1\_group()

1. 入力として盤面、調べたい行、調べたい列、マスに入れたい数字を与える。
2. グループでのチェック関数を作成した時と同様に与えられた調べたいマスが所属するグループを求める。
3. 空きマスが1つであれば、そのグループの全てのマスについてチェック関数を使う。

(i) 制約を満たす場合、入れたい数字をマスにいれ、Trueを返す。

(ii) 制約を満たさない場合、何も操作をせずFalseを返す。

(b)グループ内に空きマスが1つでなければ何も操作をせずFalseを返す。

テクニック２：可能性確定法(可能性が唯一の場合に確定する方法)

　全てのマスの可能性を調べる関数 tech2\_possibility()

1. 入力として盤面を与える。
2. 盤面すべてのマスを調べていく。

それぞれのマスの可能性を保持するリスト(マス可能性リスト)を左上から順に作成し、それを、可能性全体をまとめるリスト(盤面可能性リスト)に順に追加していくことでマスの順番に可能性を保存した盤面可能性リストを作成することができる。

すでに盤面に数字が入っているマスは可能性を調べる必要がないため、マス可能性リストは空リストとなる。空きマスには1~9までの数字を入れ、制約を満たすかをチェックする。制約を満たす数字はそのマスに入る可能性があるため、マス可能性リストに追加する。

可能性確定法　tech2\_solver()

1. 入力として盤面を与える。
2. 入力の盤面に対してtech2\_possibilityを使い、すべての可能性を調べる。
3. マス可能性リストの長さを調べ、リストの長さが1であれば可能性(そのマスに入れることが出来る数字)が1つだけであるため、その数字を盤面に入れる。
4. もう一度可能性確定法が使えないかを調べる。

使える場合は3で数字を入れた後の盤面を入力として1に戻る。

これを盤面の変化がなくなるまで行う。

（補足）

　　ここで盤面可能性リストにはすべてのマス可能性リストが順番に入っているが、盤面に入っている数字を指定するときと同じ方法は使えない。

　(盤面は2重リストでvalue[2][3]のように9×9の表現で指定出来るが、盤面可能性リストはマスの順番にマス可能性リストが入っているだけでなので、potential[マスの順番][マス可能性リストの中身]のように指定する必要がある。そのためマスの順番を[2][3]のように9×9の形で指定するのではなく、[9 \* y + 1]という式で指定する必要がある)

　よって、可能性が1つだけの場合にマスに数字を入れる際には、potential[9 \* y + 1][0]を入れることで、正確なマスと入れる数字を指定することが出来る。

　　もう一度可能性確定法が使えるかどうかは、方法を使う前と使った後で埋まったマスが増えたかどうかで決める。盤面に変化があるのであれば更に進めることが出来る可能性があるためもう一度使う。変化がなければ可能性確定法を終了する。

2つのテクニックを使う関数　technique()

1. 入力として盤面を与える。
2. その盤面に対してエリア確定法と可能性確定法の2つの基本テクニックを使う。
3. 入力として与えられた盤面とテクニック使用後の盤面を比較し、変化があれば変化後の盤面を

入力とし1に戻る。これを盤面の変化がなくなるまで行う。

盤面が正解であるかを判定する関数　answer()

1. 入力として盤面を与える。
2. 盤面すべてのマスを調べ空きマスの数を数える。

空きマスの数が0個であればすべてのマスが1~9の数字で埋まっている状態となっている。

空きマスを埋める際にはcheck関数を利用しているため、制約を満たさない数字が盤面に入っていることはない。つまりマスが全て埋まっていればそれが正解の盤面となる。

1. 空きマスの数が0個であればTrueを返す。
2. 空きマスの数が0個でなければFalseを返す。

プログラムを動かすうえで必要な関数

テキストを二次元配列に変換する関数　value\_from\_grid()

　入力するテキストを81文字に設定し(パズル内の数字の数)それを9\*9の配列に変換することで数独を

プログラムとして考えやすくするためのもの。

　空のリスト(row)に入力のテキストを入れていき、9文字入れたら盤面(value)の空リストにリストを追加する。これを最後まで行い、パズルの盤面を表す9\*9の二次元リストを作成する。

参考文献

<https://algoful.com/Archive/Algorithm/BFS>(幅優先探索の解説)

ここに「はじめに」を書く．

\section{論文の書式}

論文は，A4 版の PDF データで提出する．

LaTeX で作成する場合は，このサンプル PDF を作成するのに用いた TeX ファイルを編集し作成するとよい．

LaTeXの参考書には\cite{rakuraku,bibunsyo}がある．

この PDF ファイルの構成（表紙，概要，論文本体）と見た目に準じたものであれば

Word 等を用いて作成してもよい．

論文作成にあたっては指導教員の指示を仰ぐこと．

\subsection{使用言語}

論文を記述するのに使用する言語は，日本語または英語とする．

%

\subsection{ページのレイアウト}

製本その他読みやすさ等を考慮して，マージンは大きめにとること．

\vspace\*{3mm}

\hspace\*{10mm}

\begin{tabular}{ll}

上マージン & 25mm 程度 \\

下マージン & 30mm 程度 (ページ番号もマージン内に含む) \\

左マージン & 35mm 程度 (製本の都合上 30mm 以上は必要) \\

右マージン & 25mm 程度 \\

\end{tabular}

\subsection{文字の大きさ}

読みやすさ等を考慮して，極端に小さい文字や大きな文字はさけ，行間は十分にあけること．

文字サイズ 11-12pt, 1ページ 30 行で日本語の場合は 1 行あたり 40 文字程度が目安となる．

プログラム注意点

幅優先探索では数字を入れる際にchildの空きマスの部分を書き換えるが、1つの盤面に対して何度も書き換えを行う必要があるため、childのコピーを作成し、コピーを書き換えていくことで不具合なく次の子を作成することが出来る。

エリア確定法(a)で、すでに盤面にマスが埋められている部分に対してこの操作が起こらないように

マスに数字を入れる前に空きマスであることを確認する。

可能性確定法の補足

ここで盤面可能性リストにはすべてのマス可能性リストが順番に入っているが、盤面に入っている数字を指定するときと同じ方法は使えない。

　(盤面は2重リストでvalue[2][3]のように9×9の表現で指定出来るが、盤面可能性リストはマスの順番にマス可能性リストが入っているだけでなので、potential[マスの順番][マス可能性リストの中身]のように指定する必要がある。そのためマスの順番を[2][3]のように9×9の形で指定するのではなく、[9 \* y + 1]という式で指定する必要がある)

　よって、可能性が1つだけの場合にマスに数字を入れる際には、potential[9 \* y + 1][0]を入れることで、正確なマスと入れる数字を指定することが出来る。

　　もう一度可能性確定法が使えるかどうかは、方法を使う前と使った後で埋まったマスが増えたかどうかで決める。盤面に変化があるのであれば更に進めることが出来る可能性があるためもう一度使う。変化がなければ可能性確定法を終了する。

実験と評価

実験の目的は基本テクニックの繰り返しでどこまでパズルを解くことが出来るかを確かめること.

また,それで答えにたどり着くことが出来なければ, どのような工夫をすれば最後まで解くことが出来るか,

より早く解くことが出来るかを考察することである.

使用する問題

難易度は本, サイトで定められた通りである。

問題の順番は難易度ごとに本での問題, サイトでの問題の順である。

ただし, 超難問に関してはサイトに問題がなかったため, 本から2題選んだ.

以下の問題を順に問題1, 問題2, …, 問題12とする。

入門

問題1.

506080703

190020085

007509400

070000030

400937006

063000170

700010009

040608050

001000300

空きマス47個

問題２

840030092002000600030406080500080009104090207060304010007000300050060020308509106

840030092

002000600

030406080

500080009

104090207

060304010

007000300

050060020

308509106

問題3

026050307

839017050

074030800

000023709

213098006

907405023

091500038

300000600

768002910

空きマス 37個

初級

問題4.

700050004

050207010

010090070

601000705

007605400

000804000

530020048

028000650

006000200

空きマス 50個

問題5

940060051001070300000105000090000020310259064020000010000603000004010600530090070

940060051

001070300

000105000

090000020

310259064

020000010

000603000

004010600

530090070

問題6

000007020

057609004

000401098

001000060

520916800

000000901

079040000

310890405

000000010

空きマス 50個

中級

問題7.

200703009

400000006

080609010

008050600

700080005

050102040

090000050

107000802

000804000

空きマス 53個　幅優先を使う場面2度

問題8

800095300050300040009020005020100004004000200900003080200060700070001020001780009

800095300

050300040

009020005

020100004

004000200

900003080

200060700

070001020

001780009

問題9

000026030

907000050

600080000

000002005

030000600

000090470

050007018

000430090

800010000

空きマス 57個　幅優先を使う場面2度

上級

問題10.

020030070

006090500

000102000

735000126

000010000

000607000

003060400

070408050

400000008

空きマス 55個　幅優先1度

問題11

031050000200300100400700080600800000083000960000005003060003005004001006000060730

031050000

200300100

400700080

600800000

083000960

000005003

060003005

004001006

000060730

問題12.

080000000

004000730

000657100

000940005

008300009

060100000

001500020

007006000

043000000

空きマス 58個　幅優先1度

難問

問題13.

020000010

004000800

060010040

700209005

003000400

050000020

006801200

800050004

500030006

空きマス 56個　幅優先2度

問題14

000310008006080000090600100509000000740090052000000409007004020000020600400069000

000310008

006080000

090600100

509000000

740090052

000000409

007004020

000020600

400069000

問題15.

000630000

000000200

428700006

102000079

080000300

009003008

000007020

000090450

056000000

空きマス57個 　幅優先5度

超難問

問題16.

001503900

040000080

002000500

010060050

400000003

000201000

900080006

500406009

006000300

空きマス 57個　31度

[0, 0, 1, 5, 0, 3, 9, 0, 0]

[0, 4, 0, 0, 7, 0, 0, 8, 0]

[0, 0, 2, 0, 0, 0, 5, 1, 0]

[0, 1, 0, 0, 6, 0, 0, 5, 0]

[4, 0, 0, 0, 0, 0, 0, 0, 3]

[0, 0, 0, 2, 0, 1, 0, 0, 0]

[9, 0, 0, 0, 8, 0, 0, 0, 6]

[5, 0, 0, 4, 0, 6, 0, 0, 9]

[0, 0, 6, 0, 0, 0, 3, 0, 0]]

問題17.

080100000000070016610800000004000702000906000905000400000001028450090000000003040

080100000

000070016

610800000

004000702

000906000

905000400

000001028

450090000

000003040

[0, 8, 0, 1, 3, 0, 0, 0, 0]

[0, 0, 0, 0, 7, 0, 0, 1, 6]

[6, 1, 0, 8, 0, 0, 0, 0, 0]

[0, 0, 4, 0, 0, 0, 7, 3, 2]

[0, 0, 0, 9, 0, 6, 0, 0, 0]

[9, 0, 5, 0, 0, 0, 4, 0, 0]

[0, 0, 0, 0, 0, 1, 0, 2, 8]

[4, 5, 0, 0, 9, 0, 0, 0, 0]

[0, 0, 0, 0, 0, 3, 0, 4, 0]

空きマス57個　11度

問題18

001040600000906000300000002040060050900302004030070060700000008000701000004020500

001040600

000906000

300000002

040060050

900302004

030070060

700000008

000701000

004020500

　使用するソルバー

ソルバー1. エリア確定法のみ(1番簡単)

ソルバー2. エリア確定法と可能性確定法(2番目に簡単)

ソルバー3. 4.1.1のプログラム(エリア確定法と可能性確定法と幅優先探索

(必ず答えにたどり着く方法で簡単))

1,2に関しては最後まで解くことが出来るかを確かめる。

3に関しては時間があれば必ず解が出るためどれくらいの時間で解が出るかを確かめる。

結果

ソルバー１の結果

問題1～12. 1マスも埋められなかった

ソルバー2の結果

問題1. 最後まで解けた

2.最後

3.最後

4.最後

５．1マスも

６．最後

問題7 .6マス

８．1マスも

問題9 .8マス

問題10.２８マス

11. 1マスも

問題12. 10マス

問題１３. 1マスも埋められなかった

14.1マス埋めることが出来た

15.1マスも

16.1マスも

17.1マスも

18.1マスも

ソルバー3の結果

問題1. 0.095s

問題2. 0.082s

問題3. 0.093s

問題4. 0.090s

問題5. 12.784s

問題6. 13.311s

問題7. 0.165s

問題8. 2.092s

問題9. 17.927s

問題10. 1m 6.435s

問題11. 7m 33,498s

問題12. 2m 27.038s

全体の結果のまとめ

基本テクニックの繰り返しのみで解くことが出来るのは入門,初級まで. ただ中級,上級までは途中まで進めることが出来る. 難問, 超難問に関しては基本テクニックだけで解くことは難しい.

幅優先探索を部分的に使うと上級までの問題は早い段階で解き終わるが, 難問以上になるとかなり時間がかかる.

幅優先探索で数字を入れるときは多くても2マスで, 数字を入れたときにその数字が正解であるかや入れたことによって基本テクニックがどれだけ使えるかで正解にたどり着く時間が変わる. 例えば, 同じ超難問の問題でも問題11では, 2行, 5列に7と3行8列に1を入れたときに基本テクニックが正解にたどり着くまで使えるが,

問題12では, 1行, 5列に3と4行8列に3を入れたときに基本テクニックが正解にたどり着くまで使えるので,

解き終わる時間がかなり早くなっている.

結果の考察

　ソルバー１とソルバー2の結果より基本テクニック1より2の方がより強力であることが分かった.

1つの問題で基本テクニックが使えない場面が訪れるのは0～2回で, 回数が少なければ問題を解く時間が早くなる. 幅優先探索で数字を入れたときにその数字が正解であるかや入れたことによって基本テクニックがどれだけ使えるかで正解にたどり着く時間が変わる.例えば, 同じ超難問の問題でも

問題11では, 2行, 5列に7と3行8列に1を入れたときに基本テクニックが正解にたどり着くまで使えるが,

問題12では, 1行, 5列に3と4行8列に3を入れたときに基本テクニックが正解にたどり着くまで使えるので,

解き終わる時間がかなり早くなっている.

もっと早く, 最後まで問題を解くためには基本テクニックだけでなく, 消去法など他のマスの可能性から数字を確定させていく方法を用いる必要があるだろう.

考察

一般的な数独の問題の難易度について

　まずは問題の空きマスの数が1つの指標となる. 空きマスが少なければ少ないほど, 空きマスに対する数独の制約が厳しくなるため, 基本テクニックが使える可能性が高くなる. よって簡単な問題となる. 逆に空きマスが多ければ多いほど, 問題の時点では基本テクニックが使えず, 空きマスの可能性がより多くなるため, より難しい方法で解かなければならない. 今回取り上げた問題も難易度が上がるにつれて空きマスの数も増えていく傾向にある.ただしパズルは基本的に答えが1つに限られるため, 空きマスを極端に増やしすぎることは出来ない.

　また今回実践したような, 基本テクニックだけで解けるかどうかも難易度に関わる.

実際に実験結果から入門と初級は基本テクニックだけで解けることが分かっている. 今回は基本テクニックしか扱っていないが, 中級や上級はこれより少し難易度が高い解き方だけで解くことが出来るのかもしれない.

基本テクニックのみを使う場合の問題の難易度について