ルービックキューブを解くプログラム

新谷敏朗†

概要：世界的に普及しているルービックキューブは，日本で公式発売される前にハンガリーからの留学生により持ち込まれていた．その歴史を振り返りつつ，筆者が当時自力で考えた解法を紹介する．そしてキューブを完成させるプログラムのデータ構造とアルゴリズムの概要を述べ，実際に計算した結果の例を示す．最後に，実際にロボットにルービックキューブを解かせるために必要なことについて簡単に考察する．

キーワード：ルービックキューブ，日本における歴史，解法，プログラム

Program to Solve Rubik's Cube

TOSHIO SHINTANI†

Abstract: World famous Rubik's cube had been brought by a student from Hungary before it was released officially in Japan. While looking back on its history, I introduce the algorithms to complete cubes which I figured out on my own. Then I outline the data structures and algorithms and show examples of calculation. I make a simple consideration about what is necessary for Robot to complete cubes.

Keywords: Rubik's Cube, History in Japan, Algorithms, Program

1. はじめに

ルービックキューブは改めて紹介する必要がないほど有名なパズルのひとつである．しかし，筆者はこのパズルが日本で発売されてブームを巻き起こした 1980 年より 2 年前の 1978 年春にすでにプレイしていた．ここでは，その経緯を紹介するとともに，その当時自力で考えた解法を説明する．また，その解法をプログラム化するためのデータ構造とアルゴリズムについて説明し，さらに探索アルゴリズムによる解法についても言及する．そして，最後にルービックキューブをロボットに解かせるために必要な技術について簡単な考察を行う．

2. ルービックキューブとその用語

以下のような用語を用いる．

(1) 全体的な用語

小キューブ：ルービックキューブを構成する立方体

1 面体：立方体を構成する 6 面のうち 1 面のみが外部から見えていて，その他の 5 面は隠れている小キューブ

2 面体：立方体を構成する 6 個の面のうち 2 面が見えていて，そのほかの 4 面は隠れている小キューブ

3 面体：立方体を構成する 6 個の面のうち 3 面が見えていて，そのほかの 3 面は隠れている小キューブ

辺：2 面体がある位置

隅：3 面体がある位置

面：ルービックキューブをひとつの立方体と考えたときにその立方体を構成する 6 個の正方形

元々のルービックキューブは 1 面体が 6 個，2 面体が 12 個と 3 面体が 8 個，それぞれ外部から見えているものである．これを 3×3×3，または 3^3 と表記する．理論的には，n×n×n (n=2, 3, 4, ….) に一般化可能である．

6 個の面を次のように呼ぶ．

上：U，下：D，東：E，西：W，南：S，北：N

(2) 回転に関する用語

回転の単位は 90 度であり，時計回りに回転するものとする．よって，ある面 f (f=U, D, E, W, S, N) を m (m=1,2, 3) 単位回転することを fm と書くことにする．

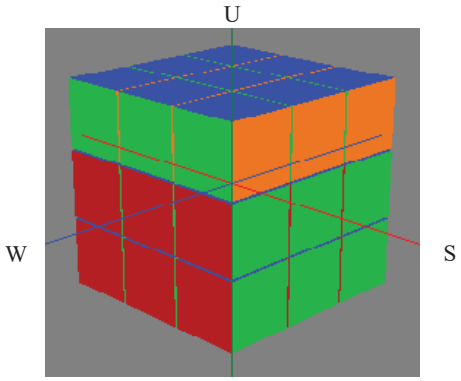


図 1 回転 U1 を実行した状態

Fig.1 State after rotating U1

†福山大学
Fukuyama University

例えば，U1 は図 1 のように，上面を時計回りに 90 度回転することを表す．2 回以上続けて回転する場合は，順に並べて表記する．例えば，上面の隅にある二つの 3 面体を，それぞれ 1/3 回転する手順 ”Rotate Two Corners Inward” (RTCI)は，

m_1 = E1U1E3U1E1U2E3,

m_2 = W3U3W1U3W3U2W1

を用いて，RTCI = m_1m_2 と表すことができる．(図 2)

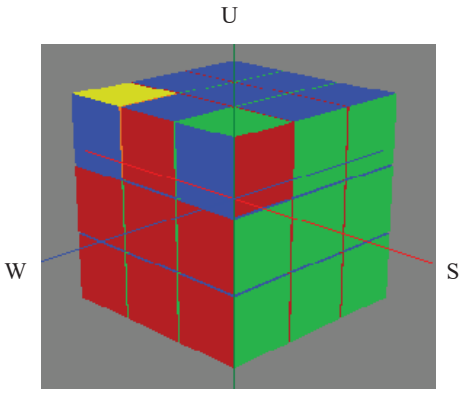


図 2 隅が 2 つ各々1/3 回転した状態

Fig.2 State after executing RTCI

また，上面 U に属する 9 個の小キューブを上段，下面 D に属する 9 個の小キューブを下段，それらの間に見えている 8 個の小キューブを中段と呼ぶ．さらに，2 面体の位置を UW，3 面体の位置を UWS などと呼ぶ．

3. 日本におけるルービックキューブ

日本では, 1980 年 6 月 1 日付の朝日新聞「遊びの博物誌」というコラムで究極の立体パズルとして紹介されている．最初はマジック・キューブと呼ばれていたようである．その記事に日本でも石毛照敏氏がマジック・キューブそっくりに動く 3 列 3 段の形のゲームを特許申請して，公告されたという事実も記載されている．実際に発売されたのは 7 月 25 日で，やはり朝日新聞の 7 月 19 日付朝刊の「ぷりずむ」というコラムに「5 分で解けたら天才？」というタイトルで紹介されている．発売時には「ルービック・キューブ」という名称になっていた．その後，大流行したことは記憶に新しいが，最近になって，再びブームになっているようである．[1]

4. 私的な歴史

ルービックキューブはハンガリーのエルノー・ルービックにより 1974 年に考案され，同国では 1977 年に発売されている．[2] したがって，大流行するより前に，ルービックキューブが日本に持ち込まれていたことは十分に考えられる．以下で紹介するのはその一例に過ぎないとは思われるが，公表されるのは初めてという可能性はある．

4.1 ハンガリーから

1978 年春に，ハンガリーからの留学生マルコン・シャンドルさん†が，京都大学工学部電気系教室の O 研究室にルービックキューブの実物を 1 個持ってきて置いていった．マルコン氏は隣の U 研に所属していたが，O 研では当時，プラパズルという商品名で売られていたペントミノをはじめとする種々のパズルが日常的にプレイされていたことが背景にあったと考えられる．

4.2 最初に完成させたのは？

研究室のメンバーは皆が 6 面完成の状態にしようとしたが，すぐにはできなかった．持ち込まれたときはバラバラで揃っていない状態だったので，当然ながらだれも完成された状態は見たことがなかった．「これ，ほんまにできるんか？」という感じもあった．さらに実物が 1 個しかないので，取り合いになる．その部屋は当時助手であった N 先生と学部，大学院生あわせて数名の学生で構成されていた．当然のことのように N 先生がまずプレイして，先生がいない間に筆者をはじめとする学生が争ってプレイするという状況であった．筆者はで学生の中では最年長だったので，プレイする時間はある程度は確保できていたが，いつでも好きな時にプレイするということではなかった．そういう理由もあって，筆者はいわゆる「紙と鉛筆」で解法を考えた．しかし完成できていないうちに，ある朝来てみたら 6 面揃ったルービックキューブが机の上に置いてあった．結果として N 先生が最初に完成させるということになった．確かに完成できるということがわかってしまうと，後追いは楽なもので，筆者が次に完成させ，他の学生も続いて 6 面揃えることができるようになっていった．

4.3 どうやって考えたか？

図 1 や図 2 のような操作をすべて手書きで図 3 のように書いていき，

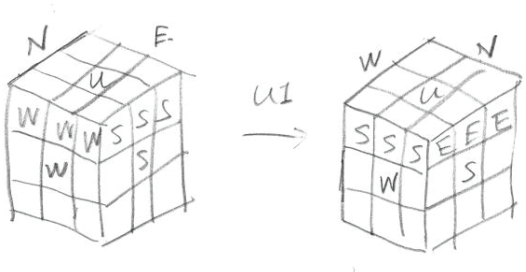


図 3 手書きで図 1 の操作を表した例

Fig.3 Handwritten example of a rotation in Fig.1

以下に説明する手順を見出したということである．筆者の

† 現在，神戸情報大学院大学におられます．

方法では，3 段のうち下段から揃えていく．下段は適当にやっても揃ったので，中段に進む．中段の 2 面体が揃っていない場合は，以下の 3 通りがある．

1) 位置は合っているが，向きが違っている
2) 中段の別の場所にある
3) 上段にある

まず, 3) の場合にそれを中段の本来あるべき場所に向きも合わせて移動する手順(**2 面体移動操作**と呼ぶことにする)を考えた．次に 1) と 2) の場合に一旦上段に移動したのちに，2 面体移動操作を行う．これにより中段までが完成する．最後に上段に移る．3 面体の位置合わせと向き合わせの手順はそれほど時間がかからずに考えだすことができた．しかし, 2 面体については向きが合わない．上段にある 2 個の 2 面体の向きを変更する手順を思いつくのに時間がかかった．このような調子で 6 面完成できるまでに 2 週間くらい経過していたと記憶している．

残念ながら以上のことは，研究としてではなく，研究の合間の息抜きのような形で行ったことだったので，記録も残していなければ，ましてや外部に公表するなどということは夢にも思わずという次第であった．研究室では「数学セミナー」も購読していたので，その後ブームになって書かれた解説記事も読みはしたが，自分にとっては過去のことという受け止め方であった．4×4×4 や 5×5×5 のものも発売され次第購入したが，特に新味はなかった．

4.4 後日談

その後しばらくは実物が 1 個しかない状態が続いていたが，1980 年の夏休みに研究室のメンバーで四国に旅行をした．何台かの車で徳島から高知に移動して，桂浜に行った際に, ルービックキューブを持っている小学生に遭遇した．びっくりして尋ねると,「西武で売っている」と言うではないか．筆者は高知市内の生まれであるが，高知に西武百貨店はなかったと思い，よく聞くと，昔の土電会館が西武の傘下になっているようだ．観光は中断して，急いで取って返し，はりまや橋交差点南東角の「とでん西武百貨店」に直行して全員が購入した．日本で発売されていることに筆者らはうかつにも気づいていなかったのであった．

福山大学に移ってからは，しばらく自宅に置いたままになっていた．2010 年頃にバラバラのキューブをそろえるための最大手数が 20 であるという研究結果が発表されたので, [3] 卒業研究でテーマとして取り上げて, 人間がプレイする機能を持つプログラムは作成した．[4]

本文中の図はそのプログラムを利用して，回転操作を実行したものである．今回このシンポジウムでルービックキューブが取り上げられたので，記憶をたどったが，思い出すことができない手順もあった．そういうこともあったので，探索アルゴリズムを利用して，実現したい手順を見出すためのプログラムを作成した．

5. 解法

ここでは, 筆者が 1978 年春に考え出したアルゴリズミックな解法，すなわち，全然揃っていない初期状態から 6 面揃った完成状態に戻す手順について述べる．

下段

この部分は, 4.3 で述べたように厳格には考えていなかったが，以下の順で揃えていく．

1) 隅の 3 面体どれかと隣接する 2 面体 2 個
2) まだ揃っていない 3 面体 3 個
3) 残った，揃っていない 2 面体 2 個

中段

上の 4.3 でのべた 2 面体移動操作を用いて，4 個の 2 面体を揃える．

上段

まず，4 個の 2 面体の向きを揃える．次に 4 個の 3 面体の位置を揃える．次に 4 個の 2 面体の位置を揃える．最後に 4 個の 3 面体の向きを揃える．これらの手順の詳細を以下に示す．

5.1 手順の関数化

既に説明した 2 面体移動操作の例として, SU にある 2 面体を SE に移動するためには,

U1 E1 U3 E3 U3 S3 U1 S1

を実行すればよい．ただし，着目していない上段のほかの 2 面体や 3 面体の状態は変化する．このような概ね 10 個以内の回転手順をひとつの関数と考えることにする．

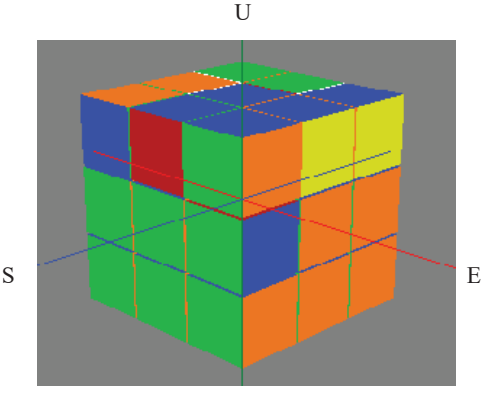


図 4　2 面体移動操作を行った状態
Fig.4 State after executing moving 2 edges

向きが違う場合は，　EU から ES に移動する同様の操作を行えばよい．その操作も含めて 2 面体移動操作と呼ぶ．

5.2 上段を揃えるために使用する関数

下段と中段がすべて揃った状態では，上段の 2 面体の向きには，次のような 3 通りの場合がある．

ア) 4 個とも合っている
イ) 2 個が合っていて，2 個が合っていない
ウ) 4 個とも合っていない

イ)　と ウ) の場合をア) の状態にするために 2 個の 2 面体の向きを変える操作を使用する．(**2 面体反転操作**と呼ぶことにする) 例えば，UN と UE の位置にある 2 個の 2 面体の向きを変えるためには，

N3 E1 N1 E3 U3 E3 U1 E1

を実行すればよい．すると，UE と UN の位置にあった 2 個の 2 面体の向きが変わる．しかし着目した 2 面体の向きとともに位置も移動するし，着目していない他の 2 面体や 3 面体の状態も変化する．したがって実際には，すぐあとで述べる 2 面体巡回置換操作を用いて 2 面体の位置を UE と UN に移動する必要が生じる場合もある．

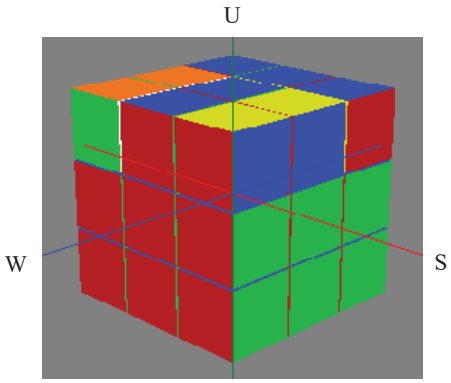


図 5　2 面体反転操作を行った状態
Fig.5 State after executing reversing 2 edges

下段と中段がすべて揃い，さらに上段の 2 面体の向きがすべて揃ったとすると，上段の 3 面体の位置を，次のような 2 通りのどちらかとすることができる．

エ) 4 個すべてが合っている
オ) 1 個が合っていて，3 個が巡回置換された状態にある

もしどちらでもなければ上段を回転することにより，エ) またはオ) のどちらかにすることができる．エ) の場合に，位置が合っていない 3 個の 3 面体を巡回置換することにより 4 個の 3 面体の位置をすべて合わすことができる．この手順を **3 面体巡回置換操作**と呼ぶことにする．この操作を図 6 に示す．この図では，黄色の面が W になっている．

その後，3 個の 2 面体の向きを変えずに位置を巡回置換する操作(**2 面体巡回置換操作**と呼ぶことにする)を実行することにより，4 個の 2 面体すべての位置と向きを合わせることができる．この操作を図 7 に示す．

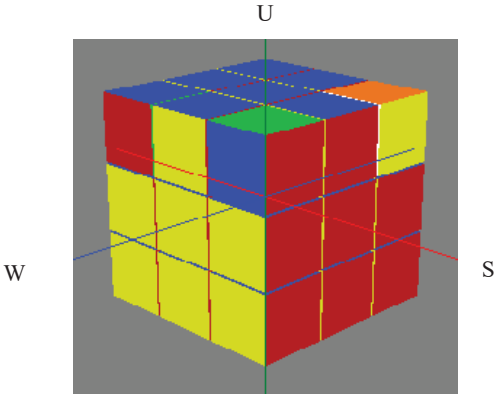


図 6　3 面体巡回置換操作を行った状態
Fig.6 State after permutating 3 corners cyclically

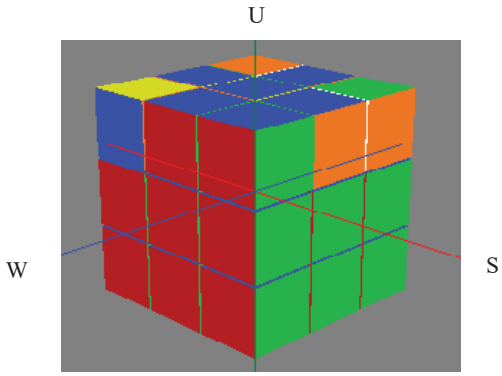


図 7　2 面体巡回置換操作を行った状態
Fig. 7 State after permutating 3 edges cycically

なお，3 面体巡回置換操作は，

W3N3E3N1W1N3E1N1

2 面体巡回置換操作は，

E1U1E3U1E1U2E3U2

である．最後に，向きが合っていない 3 面体があれば，隣り合う 3 面体の向きを変える操作(2 節の(2)で定義した RTCI)を行えば，6 面揃った完成状態にすることができる．結局，下段が揃った状態からは，2 面体移動操作，2 面体反転操作，3 面体巡回置換，2 面体巡回置換，RTCI の 5 つの基本操作を使ってルービックキューブを完成することができる．

5.3 ルービックキューブを解くプログラム

データ構造とアルゴリズムは以下のように設定した．

5.3.1 データ構造

キューブ全体は，9 個のセルからなる 6 個の面で構成する．面は，セルを要素とする 2 次元配列とする．セルは，属性として，

UP, DOWN, EAST, WEST, SOUTH, NORTH

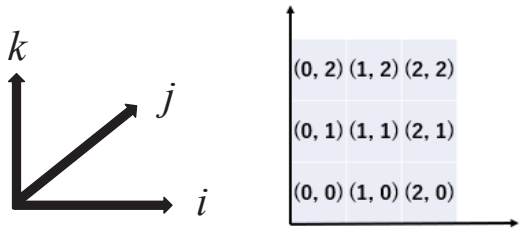
のうち，どれかのひとつの位置を保持する．具体的には C 言語のソースコードとして，

```
enum face {
    UP = 1, DOWN = 4, NORTH = 2, SOUTH = 3,
    EAST = 0, WEST = 5, OTHER = 7
};
```

とした．さらに，

```
#define SIZE 3      // 3x3x3
#define BITS 3
#define NUMOFFACES 6
struct cell {
    unsigned char position : BITS;
    // enum face のどの位置にあるかを保持する 3 ビットの
       ビットフィールド
};
struct cube {
    struct cell face[NUMOFFACES][SIZE][SIZE];
};
```

と定義される構造体を用いた．そして，座標系は 3 面体 SWD の角を原点とする右手系とした．



(a) 座標系　　(b) 面を表す配列

図 8　座標系と面を表す 2 次元配列

Fig.8 Coordinate System and 2D Array expressing a face

5.3.2 アルゴリズム

作成するプログラムの目的は，まず前節で述べた手順を実行する関数の動作確認である．もう一つは過去には発見していた手順のうち思い出すことができなかったものを再発見するためである．そのために，6 面揃った完成状態を初期状態とする．そして 6 個の面に関する 90 度単位の回転操作を行う関数 rotate を作成し，それを用いて図 2 のような特定の特徴を持った状態を初期状態にできるようにする．探索アルゴリズムについては，高々10 回程度回転した状態を生成することができれば，今回の目的のためには充分なので，単純な「深さ優先」と「レベル順」の 2 通りとした．関数 rotate のプロトタイプ宣言は，

```
enum angle {
        RIGHTANGLE=1, RIGHTANGLE2,
        RIGHTANGLE3, RIGHTANGLE4
// 直角，2 直角，3 直角，4 直角
};
struct state * rotate(struct state * t, enum face f,
                      enum angle theta);
// ある面を 90 度単位で右回転した局面を作って返す
```

とした．一つの状態から 3(回転)×6(面)＝18 個の異なる状態が生成されることになる．

アルゴリズム上の工夫としては，同じ状態のチェックが必要である．単純な例としては，U1D1 と D1U1 は同じ状態を生成するし，U1U1U1U1 で元の状態に戻るということがある．つまり，いわゆるゲーム木の探索であるが，厳密には強連結成分と有向の閉路を持つグラフにおける探索なので，チェックしないと堂々巡りが生じて探索が終了しない結果となり得る．

新たに生成された状態がすでに生成済みかどうかをチェックするためには，2 分探索木を用いた．ふたつの状態の比較のために，enum face 型のセルが持つ値の大小で局面の大小関係≦を定義した．

5.3.3 計算結果

計算に用いた環境は以下の通りである．

- ソフトウェア：Windows10, Visual Studio 2013, 2017 上の C 言語
- ハードウェア：CPU：AMD Opteron6128x2，主記憶：DDR3 256GB，　仮想記憶：3.70TB(SSD)

結果の例を次頁以降の図 9，図 10 と図 11 に示す．どれも 3 面体巡回置換された状態を完成状態に戻す探索である．手順は完成状態からさかのぼるように表記されるので，図 9 では右端の W3 が初手であり，左端の N1 により操作が完成する．単純な深さ優先なので，2 手目と 3 手目の **W1<-W3** は冗長である．9 億個あまりが生成されている．経過時間の h が **-9** から始まっているのは日本時間と UTC の差のためである．なお，深さ優先の計算では，探索の深さを 10 までに制限した．図 10 は図 9 の続きであり，別解が 3 種類得られている．

レベル順の探索結果(図 11)では，初期状態から 5.3.2 で述べたように 18 個の状態が生成され(レベル 1), 合計 1+18=19 個になり，レベル 1 の 18 個からは初期状態に戻る 18 個(重複がチェックされる) を除いた 18×18－18＝306 個の新しい状態が生成されて，レベル 2 までの合計が 19+306=325 個となっている．レベル順は幅優先なので，ここで得られた 8 手の手順が最短手順である．今回は解を 1 個見出した段階で止めてあるが，探索を続ければ別解が得られる．

6. ロボットに解かせる

現状では実現できていないが，ロボットにルービックキューブを解いてもらうには，まずキューブの状態を把握

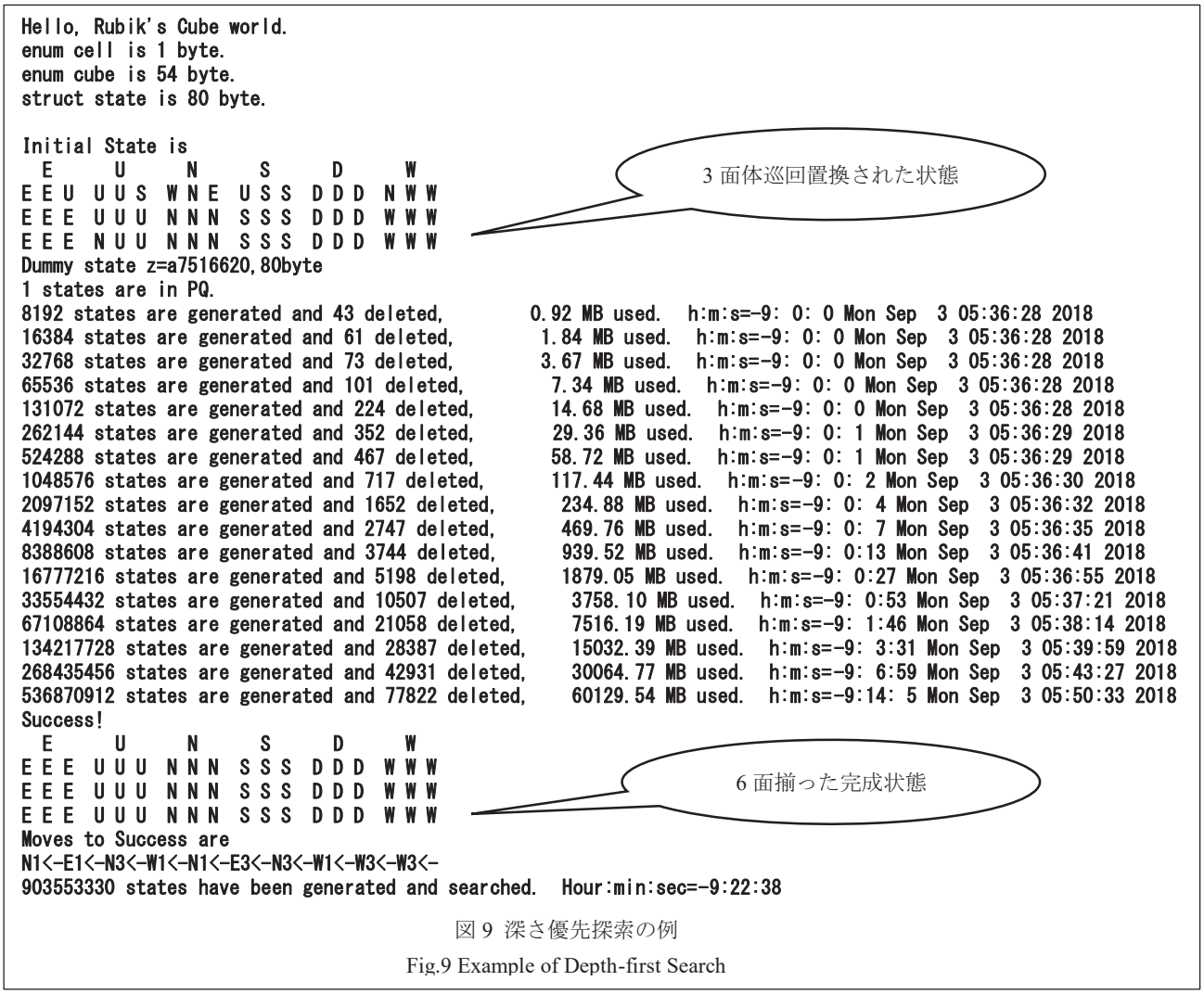


図 9 深さ優先探索の例

Fig.9 Example of Depth-first Search

させることが必要である．解くためのアルゴリズムはこれまでに述べたことから,すでに得られていると考えられる．最終的には 2 本の手でキューブを持ち，回転操作を行うことが必要である．最近のいわゆるスピードキューブは非常に滑らかに回転することが可能であるが，少しでも引っかかった時に，いったん戻してやり直すような修正動作をすることが必要になる．以上のことが実現すれば，人間より速く正確にルービックキューブを解くことはできると考えられる．技術の進歩は速いので遠からず実現することを期待したい．

7. 質疑応答

Q. (広島市立大学 小林) この発表はルービックキューブを解く方法のうちアルゴリズムで解く方法ががメインですか.

A. はい，そうです．個人的には，ロボットに解かすのであれば手順がはっきりしているアルゴリズムを使う方法で実現すべきだと思っています.

Q. (IIJ 和田) グーグルがすごい計算をしている，それに負けないようにパターン化して，手数を減らす工夫はできないか.

A. ビッグデータですか？

Q. (IIJ 和田) そう.

A. 考えてみます.

Q. (広島市立大学 川端) アルゴリズムがあるのに探索プログラムをつくる意味は？

A. アルゴリズムの動作を確認するツール､改良して手順を減らす(最短手数)手段として意味があると考えています.

Q. (大座畑) ヒューリスティック､いろいろな人から解法のノウハウを集めるのがよいのでは？

A. その通りだと思います.

Q. (大座畑) 色々な解があるということは最短経路が複数ある可能性は？

A. 当然その可能性はあります.

8. おわりに

本報告で述べたことは，以下の通りである.

1) これまで公表されていなかった「大流行する前にルービックキューブが日本に持ち込まれた例」を報告した.
2) 当時筆者が私的に考案していた基本的な回転操作をまとめた.

```
1000000000 states are generated and 149103 are deleted,    112000.00 MB used.  h:m:s=-9:27: 6 h:m:s= 6: 3:34
1073741824 states are generated and 156814 deleted,  120259.08 MB used.  h:m:s=-9:29:14 Mon Sep  3 06:05:42 2018
2000000000 states are generated and 228167 are deleted,    224000.00 MB used.  h:m:s=-9:57:12 h:m:s= 6:33:40
3000000000 states are generated and 281544 are deleted,    336000.00 MB used.  h:m:s=-8:29: 6 h:m:s= 7: 5:34
4000000000 states are generated and 322637 are deleted,    448000.00 MB used.  h:m:s=-7: 3:55 h:m:s= 7:40:23
5000000000 states are generated and 381346 are deleted,    560000.00 MB used.  h:m:s=-7:37:16 h:m:s= 8:13:44
6000000000 states are generated and 423859 are deleted,    672000.00 MB used.  h:m:s=-6:10: 4 h:m:s= 8:46:32
7000000000 states are generated and 480551 are deleted,    784000.00 MB used.  h:m:s=-6:38:19 h:m:s= 9:14:47
8000000000 states are generated and 524154 are deleted,    896000.00 MB used.  h:m:s=-5: 8: 4 h:m:s= 9:44:32
9000000000 states are generated and 592810 are deleted,   1008000.00 MB used.  h:m:s=-5:42:37 h:m:s=10:19: 5
10000000000 states are generated and 704219 are deleted,   1120000.00 MB used.  h:m:s=-4:17: 1 h:m:s=10:53:29
11000000000 states are generated and 848857 are deleted,   1232000.00 MB used.  h:m:s=-4:48: 5 h:m:s=11:24:33
Success!

  E     U     N     S     D     W
E E E  U U U  N N N  S S S  D D D  W W W
E E E  U U U  N N N  S S S  D D D  W W W
E E E  U U U  N N N  S S S  D D D  W W W
Moves to Success are
N1<-E1<-N3<-W1<-N1<-E3<-N3<-W2<-W2<-W3<-
12000000000 states are generated and 986436 are deleted,   1344000.00 MB used.  h:m:s=-3:24:53 h:m:s=12: 1:21
13000000000 states are generated and 1063428 are deleted,   1456000.00 MB used.  h:m:s=-3:58:14 h:m:s=12:34:42
Success!

  E     U     N     S     D     W
E E E  U U U  N N N  S S S  D D D  W W W
E E E  U U U  N N N  S S S  D D D  W W W
E E E  U U U  N N N  S S S  D D D  W W W
Moves to Success are
W2<-D3<-E2<-D1<-W1<-D3<-E2<-D1<-W2<-W3<-
14000000000 states are generated and 1106233 are deleted,   1568000.00 MB used.  h:m:s=-2:37:32 h:m:s=13:14: 0
15000000000 states are generated and 1162068 are deleted,   1680000.00 MB used.  h:m:s=-1:11:23 h:m:s=13:47:51
16000000000 states are generated and 1205856 are deleted,   1792000.00 MB used.  h:m:s=-1:47:31 h:m:s=14:23:59
17000000000 states are generated and 1261641 are deleted,   1904000.00 MB used.  h:m:s= 0:23:12 h:m:s=14:59:40
18000000000 states are generated and 1304418 are deleted,   2016000.00 MB used.  h:m:s= 0:58:18 h:m:s=15:34:46
19000000000 states are generated and 1350518 are deleted,   2128000.00 MB used.  h:m:s= 1:34:11 h:m:s=16:10:39
20000000000 states are generated and 1432616 are deleted,   2240000.00 MB used.  h:m:s= 2: 9:49 h:m:s=16:46:17
21000000000 states are generated and 1531720 are deleted,   2352000.00 MB used.  h:m:s= 2:46:17 h:m:s=17:22:45
22000000000 states are generated and 1647333 are deleted,   2464000.00 MB used.  h:m:s= 3:21:55 h:m:s=17:58:23
23000000000 states are generated and 1656265 are deleted,   2576000.00 MB used.  h:m:s= 3:56:59 h:m:s=18:33:27
24000000000 states are generated and 1686537 are deleted,   2688000.00 MB used.  h:m:s= 4:32:11 h:m:s=19: 8:39
25000000000 states are generated and 1733584 are deleted,   2800000.00 MB used.  h:m:s= 5: 8: 5 h:m:s=19:44:33
Success!

  E     U     N     S     D     W
E E E  U U U  N N N  S S S  D D D  W W W
E E E  U U U  N N N  S S S  D D D  W W W
E E E  U U U  N N N  S S S  D D D  W W W
Moves to Success are
N1<-E1<-N3<-W1<-N1<-E3<-N3<-D1<-D3<-W3<-
26000000000 states are generated and 1829834 are deleted,   2912000.00 MB used.  h:m:s= 5:43:27 h:m:s=20:19:55
27000000000 states are generated and 1838728 are deleted,   3024000.00 MB used.  h:m:s= 6:18:50 h:m:s=20:55:18
Out of memory in copyState.
```

別解その 1

別解その 2

別解その 3

図 10 深さ優先探索の例(図 9 の続き)

Fig.10 Example of Depth-first Search (following Fig.9)

3) それらを利用してロボットにルービックキューブを解かせるためのデータ構造を定めた.
4) キューブを完成させるための基本手順 5 個をアルゴリズム化し，それらの動作を確認できるプログラムを作成した.
5) 解を求めるための単純な探索プログラムも作成した.
6) 10 回程度の回転操作について探索プログラムが正しく動作することを確認した.

```
Hello, Rubik's Cube world.
enum cell is 1 byte.
enum cube is 54 byte.
struct state is 80 byte.
Type filename of datafile.
Datafile is Permute3Corners.cub
  E     U     N     S     D     W
E E U  U U S  W N E  U S S  D D D  N W W
E E E  U U U  N N N  S S S  D D D  W W W
E E E  N U U  N N N  S S S  D D D  W W W
Dummy node z=7f0873d0,80byte
1 nodes at level   0, total 1, 0 deleted. time -9: 0: 0,Fri Sep 21 08:28:14 2018
18 nodes at level   1, total 19, 0 deleted. time -9: 0: 0,Fri Sep 21 08:28:14 2018
306 nodes at level   2, total 325, 18 deleted. time -9: 0: 0,Fri Sep 21 08:28:14 2018
5472 nodes at level   3, total 5797, 54 deleted. time -9: 0: 0,Fri Sep 21 08:28:14 2018
      8192 states are generated and 64 deleted,        0.92 MB used.  h:m:s=-9: 0: 0 Fri Sep 21 08:28:14 2018
     16384 states are generated and 96 deleted,        1.84 MB used.  h:m:s=-9: 0: 0 Fri Sep 21 08:28:14 2018
     32768 states are generated and 164 deleted,       3.67 MB used.  h:m:s=-9: 0: 0 Fri Sep 21 08:28:14 2018
     65536 states are generated and 296 deleted,       7.34 MB used.  h:m:s=-9: 0: 1 Fri Sep 21 08:28:15 2018
98100 nodes at level   4, total 103897, 450 deleted. time -9: 0: 1,Fri Sep 21 08:28:15 2018
    131072 states are generated and 491 deleted,       14.68 MB used.  h:m:s=-9: 0: 1 Fri Sep 21 08:28:15 2018
    262144 states are generated and 643 deleted,       29.36 MB used.  h:m:s=-9: 0: 1 Fri Sep 21 08:28:15 2018
    524288 states are generated and 998 deleted,       58.72 MB used.  h:m:s=-9: 0: 1 Fri Sep 21 08:28:15 2018
   1048576 states are generated and 1706 deleted,      117.44 MB used.  h:m:s=-9: 0: 2 Fri Sep 21 08:28:16 2018
1763496 nodes at level   5, total 1867393, 2754 deleted. time -9: 0: 3,Fri Sep 21 08:28:17 2018
   2097152 states are generated and 3034 deleted,      234.88 MB used.  h:m:s=-9: 0: 4 Fri Sep 21 08:28:18 2018
   4194304 states are generated and 4388 deleted,      469.76 MB used.  h:m:s=-9: 0: 6 Fri Sep 21 08:28:20 2018
   8388608 states are generated and 7071 deleted,      939.52 MB used.  h:m:s=-9: 0:12 Fri Sep 21 08:28:26 2018
  16777216 states are generated and 12706 deleted,    1879.05 MB used.  h:m:s=-9: 0:24 Fri Sep 21 08:28:38 2018
  33554432 states are generated and 23942 deleted,    3758.10 MB used.  h:m:s=-9: 0:47 Fri Sep 21 08:29:01 2018
31721688 nodes at level   6, total 33589081, 23994 deleted. time -9: 0:50,Fri Sep 21 08:29:04 2018
  67108864 states are generated and 34068 deleted,    7516.19 MB used.  h:m:s=-9: 1:37 Fri Sep 21 08:29:51 2018
 134217728 states are generated and 52558 deleted,   15032.39 MB used.  h:m:s=-9: 3: 9 Fri Sep 21 08:31:23 2018
 268435456 states are generated and 91504 deleted,   30064.77 MB used.  h:m:s=-9: 6:19 Fri Sep 21 08:34:33 2018
 536870912 states are generated and 169394 deleted,  60129.54 MB used.  h:m:s=-9:12:42 Fri Sep 21 08:40:56 2018
570824352 nodes at level   7, total 604413433, 190026 deleted. time -9:15:18,Fri Sep 21 08:43:32 2018
Success!
  E     U     N     S     D     W
E E E  U U U  N N N  S S S  D D D  W W W
E E E  U U U  N N N  S S S  D D D  W W W
E E E  U U U  N N N  S S S  D D D  W W W
Moves to Success are
N1<-E1<-N3<-W1<-N1<-E3<-N3<-W3<-
903553330 states have been generated and searched.  Hour:min:sec=-9:22:38
```

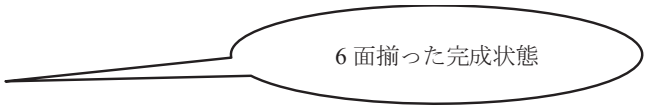


図 11 レベル順の探索例

Fig.11 Example of level-first search

今後の課題としては,

a) 下段を揃える手順をアルゴリズム化すること

b) 完成までに必要な操作回数を減らすこと

c) 探索プログラムに関する計算量の評価と効率化

d) 解いていく過程の可視化

などがあげられる.

謝辞 公式発売より前に日本に持ち込まれていた時期について，貴重な情報を快く提供していただいたマルコン・シャンドル氏に謹んで感謝の意を表する.

参考文献

[1] http://news.nicovideo.jp/watch/nw3552013

[2] https://rubikcube.jp/history/

[3] http://www.cube20.org/

[4] 石田交良. XNA によるルービックキューブシステムの開発 - ルービックキューブにおけるキューブの回転の実装 - . 福山大学卒業論文, 2012