コンピュータ教育におけるアルゴリズムの考え方

――ゲームの解法プログラムの利用――

兼　山　瓊　典

The education of a computer and an algorithm.

Tamafumi KANEYAMA

Abstract

When I am teaching the computer programming, I felt that it is difficult for students to understand the algorithms in the computer programming. In the beginning of the programming, students can easily understand the algorithms, but gradually they will be not to understand them. So I think that the good way to understand the algorithms is to use a game. When they solve the game, they will think a lot of algorithms. This paper is aimed to show the program which can solve the game. It is a good example to consider the algorithm.

１．は　じ　め　に

現在コンピュータの授業で，プログラムの考え方と作成を教えている。プログラムを理解し作成できることは，コンピュータを理解しいろいろなソフトを使いこなす上で有益である。ソフトは，単純に使うものと，プログラムの形で操作させるものがある。特に後者の場合にはその応用力が期待できる。しかし，授業の中で最初のうちは単純なプログラムであるので理解し易いが，段々と複雑になると当然のことであるが理解が大変に困難である。複雑になってきたときに，ゲームというものを考えると興味が持てる。興味を持ってそのゲームをいろいろ解いている過程で多くのアルゴリズムを考える。そして自分の考え方をプログラムにしようという意識を持たせたい。問題に対して，自分でアルゴリズムを考えることは相当大変である。考え方は分かってもそれがどのようなアルゴリズムであるかを具体的に述べることは難しい。いろいろな考え方があり，アルゴリズムは１通りとは限らない。どのようなアルゴリズムで考えるとよいプログラムが作成出来るかは実際に実行してみないと判断出来ない場合がある。ここでは，現在流行しているゲームを取り上げ，それを利用してコンピュータでプログラムを作り解くことを考える。プログラムを作る上で，どのようなアルゴリズムを考えるかが大変である。人間がゲームを解く場合の思考方法を考え，それを解析して単純化する。その本質的な部分を見つけてプログラムにする必要がある。本質的な部分は，人間の思考をアルゴリズムという形で考え，それをプログラムという形にしなければならない。ゲームを解く思考方法はいろいろあり，そのアルゴリズムは複雑である。どのアルゴリズムを選ぶかはプログラムを作るときの複雑さを軽減してくれる。ここでは，その中の１つで，理解し易くプログラムし易いものを取り上げて，解説する。

※ E-mail　kaneyama@ha.shotoku.ac.jp

２．ゲームについて

今回取り上げるゲームは，少し前から流行していて，世界的に好まれているものである。名前は，「数独」，「ナンプレ」，「SuDoKu」「Number Place」などいろいろある。ルールは単純でありながらその複雑性により多くの人々に愛好されている。このゲームは，単純でありながら考えなければ解けなく，その思考過程が楽しいという理由から多くの人々が熱中するゲームである。ここでは一般的な９×９のマス目のものを考える。ルールはどの縦１列にも１から９までの数字が入り，どの横１列にも１から９までの数字が入る。さらに３×３のどの９つのブロックにも１から９までの数字が入るというものである。最初に数字が多ければ簡単に解けるが，数字の位置や，数によってはなかなか解けない場合もある。易しい問題ではないが，後に問題１，問題２を載せてあるので，実際に解いてみると，どのようなものであるかが分かる。

３．アルゴリズムについて

１つの問題を考えそれを解くためのアルゴリズムは多くある。どのように解けばよいかは，問題を数多く解いてみることによって分かってくる。数多く解く過程でいろいろな解き方即ちアルゴリズムが分かる。この段階を経ることにより，人間の思考をアルゴリズムというものに変換し考え方をはっきりさせることが出来る。いろいろなアルゴリズムが見つかった後で，どのアルゴリズムが一番プログラムに適しているかを考えなければならない。複雑なアルゴリズムはプログラムを作成するときに苦労し，易しいアルゴリズムはプログラムを作成するとき楽である。易しいアルゴリズムであっても良いアルゴリズムかどうかは判断の難しいところである。数独問題解決のためのアルゴリズムはいろいろ考えられるが，ここでは理解しやすいことを１番に考える。人間の思考を，プログラムにするために次のように考える。人間の場合に縦横ブロックの数字をみて，ある空きの部分に入れる数字はこれしかないという思考を逆に考えて，空きの部分に入れてはいけない数字を消すことにより入れてもよい数字が１つになればその数字しかないという考えでプログラムを作る。このプログラムの例はアルゴリズムの考察のためであるので，完全に問題が解けるものではないことを断っておく。プログラムも理解を優先させて，理解しにくい記述は避けてある。この程度のアルゴリズムは，学生に理解されやすいと考えて，これからの学習の意欲がわきさらに発展させて学習することを期待する。また完全なプログラムは機会があれば発表することとする。

４．プログラムの実際

プログラム言語は，Ｃ言語を使う。Ｃ言語はプログラムの構造が理解しやすく，細かいことがいろいろ出来るからである。問題のための配列変数ｂ[10][10] を用意する。変数は次のように考える。

b[1][1]	b[2][1]	b[3][1]	b[4][1]	b[5][1]	b[6][1]	b[7][1]	b[8][1]	b[9][1]
b[1][2]	b[2][2]	b[3][2]	b[4][2]	b[5][2]	b[6][2]	b[7][2]	b[8][2]	b[9][2]
b[1][3]	b[2][3]	b[3][3]	b[4][3]	b[5][3]	b[6][3]	b[7][3]	b[8][3]	b[9][3]
b[1][4]	b[2][4]	b[3][4]	b[4][4]	b[5][4]	b[6][4]	b[7][4]	b[8][4]	b[9][4]
b[1][5]	b[2][5]	b[3][5]	b[4][5]	b[5][5]	b[6][5]	b[7][5]	b[8][5]	b[9][5]
b[1][6]	b[2][6]	b[3][6]	b[4][6]	b[5][6]	b[6][6]	b[7][6]	b[8][6]	b[9][6]
b[1][7]	b[2][7]	b[3][7]	b[4][7]	b[5][7]	b[6][7]	b[7][7]	b[8][7]	b[9][7]
b[1][8]	b[2][8]	b[3][8]	b[4][8]	b[5][8]	b[6][8]	b[7][8]	b[8][8]	b[9][8]
b[1][9]	b[2][9]	b[3][9]	b[4][9]	b[5][9]	b[6][9]	b[7][9]	b[8][9]	b[9][9]

b[i][j]の値は0から9までで，0の時は問題としてブランクのマス目とする。この問題としてのb[i][j]の値はプログラムを作り入力してもよいが，ここでは関数Init( )の中で決めるものとする。途中の判断としてb[i][j]に対して変数a[i][j][10]を考える。このa[i][j][k]の値が0の時はそのマス目に代入することが可能な数値であり，a[i][j][k]値が1の時はkが代入することが不可能であることを表す。代入することが可能な数字が1つであればその値をb[i][j]に代入する。最終的にすべてに代入することが可能でなければ，問題は解けないということである。

関数Solve( )で問題を解くのであるが，基本的には，関数Yoko( )，Tate( )，Block( )で縦横ブロックについて調べて，関数Set_a( )で変数aに0か1の値を代入していく。実際には関数Set_aSub( )でaに値を代入する。またaの値の結果により関数Set_b( )でbに値を代入してその結果を用いてさらに新しく調べる。この操作が終わったら，b[i][j]の値を関数Check2( )で調べ解けたかどうかを判断する。

プログラム

```
int     a[10][10][10], b[10][10];
void    Init ( );
void    Display ( );
int     Solve ( );
int     Check1 (int x, int y);
int     Check2 ( );
int     Yoko (int x, int y);
int     Tate (int x, int y);
int     Block (int x, int y);
void    Set_a ( );
void    Set_aSub (int x, int y, int l);
int     Set_b ( );
void main (int argc, char *argv[ ])
{
        Init ( );
```

```
        if (Solve ( )==0) Display ( );
}
void Init ( )
{   int i, j, k ;
        for (i=1 ; i<10 ; i++) for (j=1 ; j<10 ; j++) for (k=1 ; k<10 ; k++) a[i] [j] [k] =0 ;
        ここで問題 b[i] [j]に値を代入する。ブランクの時は=0とする。
}
void Display ( )
{
        結果（b[i] [j]）を表示する。
}
int Solve ( )
{       int i, j, m ;

        for (i=1 ; i<10 ; i++) for (j=1 ; j<10 ; j++) if (Check1 (i, j)==1) return1 ;
        Set_a ( );
        do {    m=0 ;
                for (i=1 ; i<10 ; i++) for (j=1 ; j<10 ; j++){
                        m=m+Yoko (i, j);
                        m=m+Tate (i, j);
                        m=m+Block (i, j);
                }
        } while (m ! =0);

        return Check2 ( );      /* 0：解けた      1 ：解けない*/
}
int Check1 (int x, int y)
{       int i, j, h, k, l, m ;

        l=b[x] [y];
        if ((l< 1 ) ‖ (l> 9 )) return 0 ;

        m= 0 ;
        for (i=1 ; i<10 ; i++) if (b[x] [i] ==l) m++;
        if (m> 1 ) return1 ;  /*だめであるとき*/

        m=0 ;
        for (i=1 ; i<10 ; i++) if (b[i] [y] == l) m++;
        if (m> 1 ) return1 ;  /*だめであるとき*/
```

```
        h=((x−1)/3)*3; k=((y−1)/3)*3;
        m=0;
        for(i=1; i<4; i++) for (j=1; j<4; j++) if (b[h+i][k+j]==1)m++;
        if (m>1) return1;

        return 0;
}
int Check2( )  /*解けたか k=0の時解けた,  k=1の時解けない*/
{       int i, j;

        for (i=1; i<10; i++) for (j=1; j<10; j++) if (b[i][j]==0) return1;
        return 0;
}
int Yoko (int y, int k)
{       int i, m=0,  x;

        for (i=1; i<10; i++){
                if (a[i][y][k]==0){
                        if (m==0)x=i;
                        m++;
                }
        }
        if (m==1){b[x][y]=k; Set_a ();  return1;}
        return 0;
}
int Tate (int x, int k)
{       int i, m=0, y;

        for (i=1; i<10; i++){
                if (a[x][i][k]==0){
                        if (m==0) y=i;
                        m++;
                }
        }
        if (m==1){b[x][y]=k; Set_a ( );  return1;}
        return 0;
}
int Block (int w, int k)
{       int i, j, m=0, x, y, z1, z2;
```

```
        y=(w-1)/3:
        x=w-1-3*y;
        x=x*3; y=y*3;
        for (i=1; i<4; i++) for (j=1; j<4; j++){
                if (a[x+i][y+j][k]==0){
                        z1=i;
                        z2=j;
                        m++;
                }
        }
        if (m==1){b[x+z1][y+z2]=k; Set_a( );  return1;}
        return 0;
}
void Set_a ( )
/* b[ ][ ]の値により縦横等を a[ ][ ][ ]の値を決める。*/
/* a[ ][ ][ ]により b[ ][ ]が決まるときは，b[ ][ ]に代入する  */
{       int i, j;

        do{
        for (i=1; i<10; i++) for (j=1; j<10; j++)
                if (b[i][j]>0)Set_aSub(i, j, b[i][j]);
        }while (Set_b ( )==1);
}
void Set_aSub (int x, int y, int l)
/* b[x][y]=l の値により a[ ][ ][l]=1とする*/
{       int i, j, h, k;

        if ((l<1) || (l>9)) return;
        for (i=1; i<10; i++){a[x][i][l]=1; a[i][y][l]=1; a[x][y][i]=1;}

        h=((x-1)/3)*3; k=((y-1)/3)*3;
        for (i=1; i<4; i++) for (j=1; j<4; j++) a[h+i][k+j][l]=1;
}
int Set_b( )  /*a[ ][ ][ ]の値により b[ ][ ]に代入*/
{       int i, j, k, m, n, index=0;

        for (i=1; i<10; i++) for(j=1; j<10; j++)
        {if (b[i][j]==0){
                m=0;
                for (k=1; k<10; k++) if (a[i][j][k]==0){m++; n=k;}
```

```
                if(m==1){b[i][j]= n ;
                        Set_aSub (i, j, n);
                        index=1 ;}
                }
        }
        return index ;
}
```

５．プログラムの考察

このプログラムでかなりの問題が解ける。実際に解いてみよう。左側に問題右側にその解答を表示する。

		6					1	
	7			6		5		
8					3	2		
		5		4		1		
	4		7				9	
		8				7		
		1	2		5			3
	6			7			8	
2						4		

問題１

3	2	6	5	9	7	8	1	4
1	7	4	8	6	2	5	3	9
8	5	9	4	1	3	2	6	7
7	3	5	9	4	8	1	2	6
6	4	2	7	5	1	3	9	8
9	1	8	3	2	6	7	4	5
4	9	1	2	8	5	6	7	3
5	6	3	1	7	4	9	8	2
2	8	7	6	3	9	4	5	1

解答１

これを利用して，少し付け加えることにより，問題を作成することが出来る。マス目の数字を変えて解けるかどうかを調べていけばよいからである。しかしこのように解ける場合はよいが，次の問題のようにこのプログラムでは解けない問題もある。このような場合は別のアルゴリズムをさらに考えなければならない。このプログラムで解けない問題は人間が解いても簡単には解けない難しい問題である。次に示した問題は，かなり難しい問題である。しかし，ゲームを解く思考を考察しその本質を見抜いた後，それらのアルゴリズムでコンピュータのプログラムにあうようなものを抜き出しプログラムにすることは，コンピュータ教育において有用であろう。このプログラムに先読みといわれるアルゴリズムを考えたものを付け加えることにより，ここで述べたプログラムでは解けない問題も解けるようになる。どのように先読みを付け加えるかは結構大変である。筆者は先読みを付け加えたプログラムを作成することにより，問題２も解けるようにしている。問題２はそのプログラムを利用して解いたものである。

	1			4		8	2	
	7		6	5	1		3	
		5		8				
	4	1	5		2	3	8	
		7		3		1		
	8		7		3			
	6	3		1		2	7	

問題 2

4	3	9	8	2	7	5	6	1
5	1	6	3	4	9	8	2	7
2	7	8	6	5	1	9	3	4
3	9	5	1	8	6	7	4	2
6	4	1	5	7	2	3	8	9
8	2	7	9	3	4	1	5	6
1	8	2	7	6	3	4	9	5
9	6	3	4	1	5	2	7	8
7	5	4	2	9	8	6	1	3

解答 2