

B 3 立体ペントミノの解法について

磯部 俊夫 (航空宇宙技術研究所)

1. はじめに

正方形を5つ平面上に並べると、図1のような形の異なった12のパターンが得られる。これらをペントミノという。

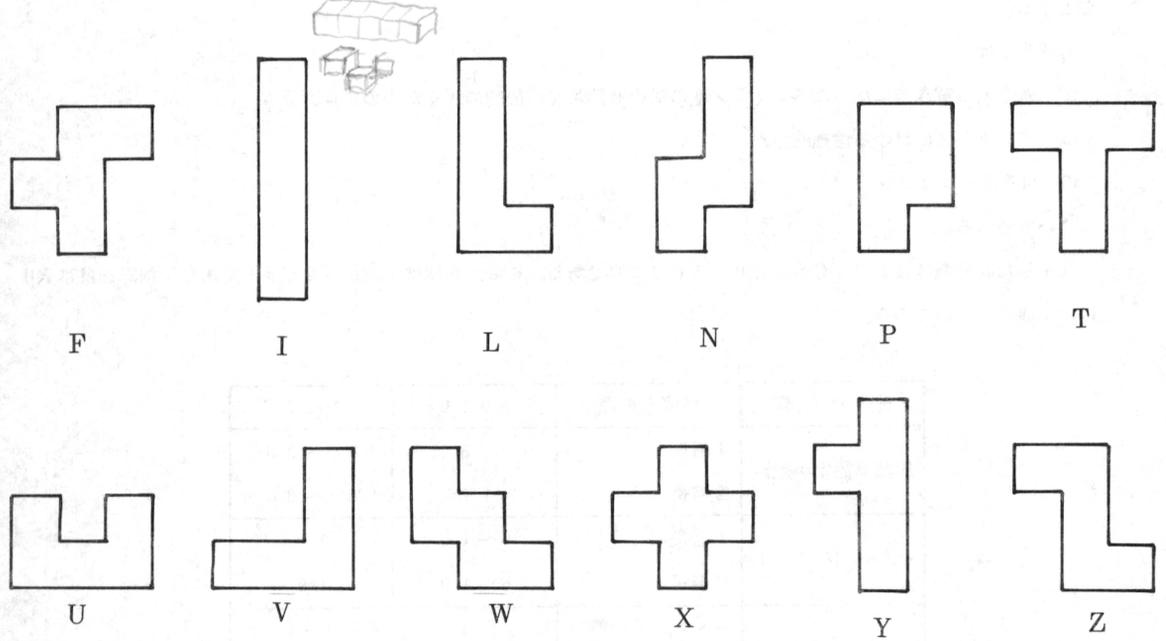


図1 ペントミノ12の素子

これらのパターンには、その形からF, I, L, N, P, T, U, V, W, X, Y, Zと名付けられている。正方形を立方体に置きかえると、これら12個の素子を使用して、直方体を組立てることができる。この12個の体積の合計は60であり、これらを使って次のような直方体を組立てられることが知られている。

$20 \times 3 \times 1$ 2種類

$15 \times 4 \times 1$ 368種類

$12 \times 5 \times 1$ 1010種類

$10 \times 6 \times 1$ 2339種類

$8 \times 8 \times 1$ 65種類

(ただし中心の 2×2 は使わない)

$10 \times 3 \times 2$ 12種類

$6 \times 5 \times 2$ 264種類

$5 \times 4 \times 3$ 3940種類¹⁾

several hundred times

ペントミノを計算機で解く試みは、日本では1960年代の中頃から富士通²⁾等で行なわれている。

ペントミノ問題の基本的な解き方は次の通りである。素子をまず1つ取り、定められた箱の中の適当な位置に適当な向きで入れる。次に別の素子を同様にして箱の中に入れる。このとき箱の中には、既に他の素子が入っているので、あ

程度の制約を受ける。このような操作を続けてゆく。後になるに従って、素子を箱の中に入れる自由度がだんだん少なくなってくる。素子全部がこのようにして箱の中に納まれば、解の1つが見つかったことになるが、大抵の場合、途中でどのようにしても入らないという状態に陥る。このようになったら、いま入れようとしている素子の前に箱に入れた素子を入れ直す。この素子をどのように入れ直しても駄目なときは、さらに前に入れた素子を入れ直す。このようにして解を見つける。この方法は、可能な組合せを全部調べるといふことになり、調べるべき数がすぐ膨大な数になってしまう。処理時間を早めるためには、能率よく処理するプログラムを作ることは勿論、処理の過程での論理的処理を必要とする。

本報告では

- (1) 基本的な解き方に従ったペントミノ型の組合せ問題を一般的に解くためのプログラム
- (2) ペントミノにおける論理的処理
- (3) 4次元ペントミノ

について述べる。

使用した計算機はHITAC 5020, 5020Fである。この計算機は1語が32ビットであり、演算速度は表1に示す通りのものである。

種類	精度その他	5020	5020F
加減算固定小数点	1倍長	8	1.0 ~ 3.0
	2倍長	12	1.5 ~ 3.0
シフト	1倍長	6~10	1.0
	2倍長	6~10	1.5
ジャンプ	ジャンプする時	4	0.56 ~ 1.75
	ジャンプしない時	8	1.0 ~ 2.0
ストア	1倍長	10	1.25 ~ 3.25
	2倍長	16	1.25 ~ 3.25

表1 HITAC 5020, 5020Fの演算速度 (単位 μ 秒)

2. 基本プログラム

箱が直方体であり、かつ素子が立方体より構成されているペントミノ型の一般問題を取り扱えるようなプログラムについて述べる。

このプログラムでは、まず各素子についてその素子が箱の中で取りうる全状態について調べた表を作る。ペントミノの場合では12の素子に対応して12の表ができる。しかる後、各表から各々1つの状態を取り出し、箱の中に入れる。全部が箱の中に入れば、そのときの組合せが解になる。この操作を全組合せについて調べる。と、いう方法で処理される。

プログラムは、箱の定義、素子の定義されたデータを読み込む入力ルーチン、定義された箱を計算機処理用のデータ構造に変換するルーチン (FRSTMS)、各素子の箱の中で取りうる全状態を調べ、表を作るルーチン (PSIFT)、各表から可能な組合せを作り箱に入るかどうかを調べるルーチン (PENTON)、出力印字ルーチンおよび全体を制

御するルーチンから構成される。

入力データ形式

ここでは、ペントミノ型に関する問題一般を扱かえるような入力データ形式について述べる。

(1) 箱の定義

箱は n 次元直方体であり

BOX / $X_1 * X_2 * \dots * X_n$

ただし X_k : k 軸方向の辺の長さ

$n = 1, 2, \dots, 4$

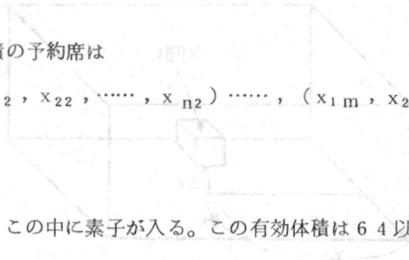
で表わす。

(2) 予約席の定義

箱の中で素子が入ることのできない領域を予約席と名づけ m 体積の予約席は

RESERVE / (記号) $(x_{11}, x_{21}, \dots, x_{n1}), (x_{12}, x_{22}, \dots, x_{n2}), \dots, (x_{1m}, x_{2m}, \dots, x_{nm})$

で表わす。



BOXの体積から予約席の体積を引いたものが有効体積になり、この中に素子が入る。この有効体積は64以下でなければならない。

(3) 素子の定義

i 体積, j 次元で定義される素子は

ELEMENT / (記号) $(x_{11}, x_{21}, \dots, x_{j1}), (x_{12}, x_{22}, \dots, x_{j2}), \dots, (x_{1i}, x_{2i}, \dots, x_{ji})$

で表わす。なお記号は、1文字で表わす各素子固有の名前である。

このような形式で書くことにより、扱かえる素子はペントミノならず直角座標系で表現できるものであれば良い。

この入力データ形式を使うと、 $5 \times 4 \times 3$ のペントミノでは、次のようにデータを書ける。

BOX / $5 * 4 * 3$

ELEMENT / (I) (1, 1), (2, 1), (3, 1), (4, 1), (5, 1)

ELEMENT / (X) (2, 1), (2, 2), (2, 3), (1, 2), (3, 2)

ELEMENT / (T) (2, 1), (2, 2), (2, 3), (1, 3), (3, 3)

ELEMENT / (U) (1, 2), (1, 1), (2, 1), (3, 1), (3, 2)

ELEMENT / (V) (1, 3), (1, 2), (1, 1), (2, 1), (3, 1)

ELEMENT / (W) (1, 1), (1, 2), (2, 2), (2, 3), (3, 3)

ELEMENT / (Z) (1, 1), (2, 1), (2, 2), (2, 3), (3, 3)

ELEMENT / (L) (1, 4), (1, 3), (1, 2), (1, 1), (2, 1)

ELEMENT / (N) (1, 1), (1, 2), (2, 2), (2, 3), (2, 4)

ELEMENT / (Y) (2, 1), (2, 2), (2, 3), (2, 4), (1, 3)

ELEMENT / (F) (1, 2), (2, 1), (2, 2), (2, 3), (3, 3)

ELEMENT / (P) (1, 1), (1, 2), (1, 3), (2, 2), (2, 3)

また真中に4体積の穴のある 8×8 のペントミノでは

BOX / $8 * 8$

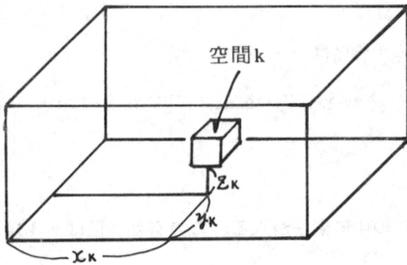
RESERVE / (*) (4, 4), (4, 5), (5, 4), (5, 5)

ELEMENT / (1)

とかける。

データ構造

入力されたデータは内部処理に適するデータ構造に変換する。有効体積は、2語（64ビット）で表わす。1ビットが体積1に対応し、先頭のビットから1, 2, ……64とアドレスを付ける。この処理上の箱をMBOXと呼ぶ。BOXとMBOXの対応表はルーチンFRSTMSで作られる。対応表は図2のような構造になっている。



(a) 実際の箱 (BOX)

空間番号	X 座 標	Y 座 標	Z 座 標
1	x_1	y_1	z_1
2	x_2	y_2	z_2
k	x_k	y_k	z_k
n	x_n	y_n	z_n

(b) BOXとMBOXとの変換テーブル



(c) 処理用の箱 (MBOX)

図 2

PSIFTルーチンでは、定義された素子が箱の中でとりうる状態すべてを求める。ここでは、各座標軸について回転操作を施し、しかる後、各軸に沿って平行移動を行なう。このようにして得られた素子の状態が、予約席を占めなければテーブルに登録する。テーブルのデータ構造はMBOXと同じである。5×4×3の箱において、例えばZ素子は、図3に示す12の異った向きをもつ。また、図3の1の向きに対して、図4のような18通りの平行移動が行なわれる。表2は各大きさの箱に対するペンタミノのとりうる状態の数と向きの数を示したものである。

PENTOMルーチン

各素子に対する状態表から、全部の素子が箱に入る組合せを見つけるルーチンである。各素子には通し番号をつけ、次のようなテーブルを用意する。

JOT (n, k) ; 番号nの素子のkの状態がかかっている。

JMT (n) ; 番号nの素子の状態の数。

JNT (n) ; MBOXに入っている番号nの素子の状態。まだ箱の中に入っていないとき0とする。

いまiの素子を箱の中に入れようとする。k=1としJOT (i, k)とMBOXの論理積をとる。0なら素子iはkの状態で箱の中に入る。そこでJMT(i)=kとし、次の素子へ移る。論理積が1なら箱に入らない。kを増加させ箱に入るものがあるかどうか探す。k=JMT(i)になっても入るものがなければ、i=i-1とし箱に入っているiの素

Z 1

00000	00000	00000
0Z200	00000	00000
0Z000	00000	00000
ZZ000	00000	00000

Z 7

00000	00000	00000
ZZ000	00000	00000
0Z000	00000	00000
0Z200	00000	00000

Z 2

00000	00000	00000
Z0000	00000	00000
ZZ200	00000	00000
00Z00	00000	00000

Z 8

00000	00000	00000
00Z00	00000	00000
ZZ200	00000	00000
Z0000	00000	00000

Z 3

00000	00000	00000
00000	00000	00000
00000	00000	00000
ZZ000	0Z000	0ZZ00

Z 9

00000	00000	00000
Z0000	Z0000	00000
00000	Z0000	00000
00000	Z0000	Z0000

Z 4

00000	00000	00000
Z0000	00000	00000
Z0000	Z0000	Z0000
00000	00000	Z0000

Z 10

00000	00000	00000
00000	00000	00000
00000	00000	00000
00Z00	ZZZ00	Z0000

Z 5

00000	00000	00000
00000	00000	00000
00000	00000	00000
0ZZ00	0Z000	ZZ000

Z 11

00000	00000	00000
00000	Z0000	Z0000
00000	Z0000	00000
Z0000	Z0000	00000

Z 6

00000	00000	00000
00000	00000	Z0000
Z0000	Z0000	Z0000
Z0000	00000	00000

Z 12

00000	00000	00000
00000	00000	00000
00000	00000	00000
Z0000	ZZZ00	00Z00

図3 素子 Z の 5 × 4 × 3 の箱での向き

子を取り除き、 $k = JNT(i) + 1$ と同じ操作を行なう。図5にPENTOMのフロー・チャートを示す。

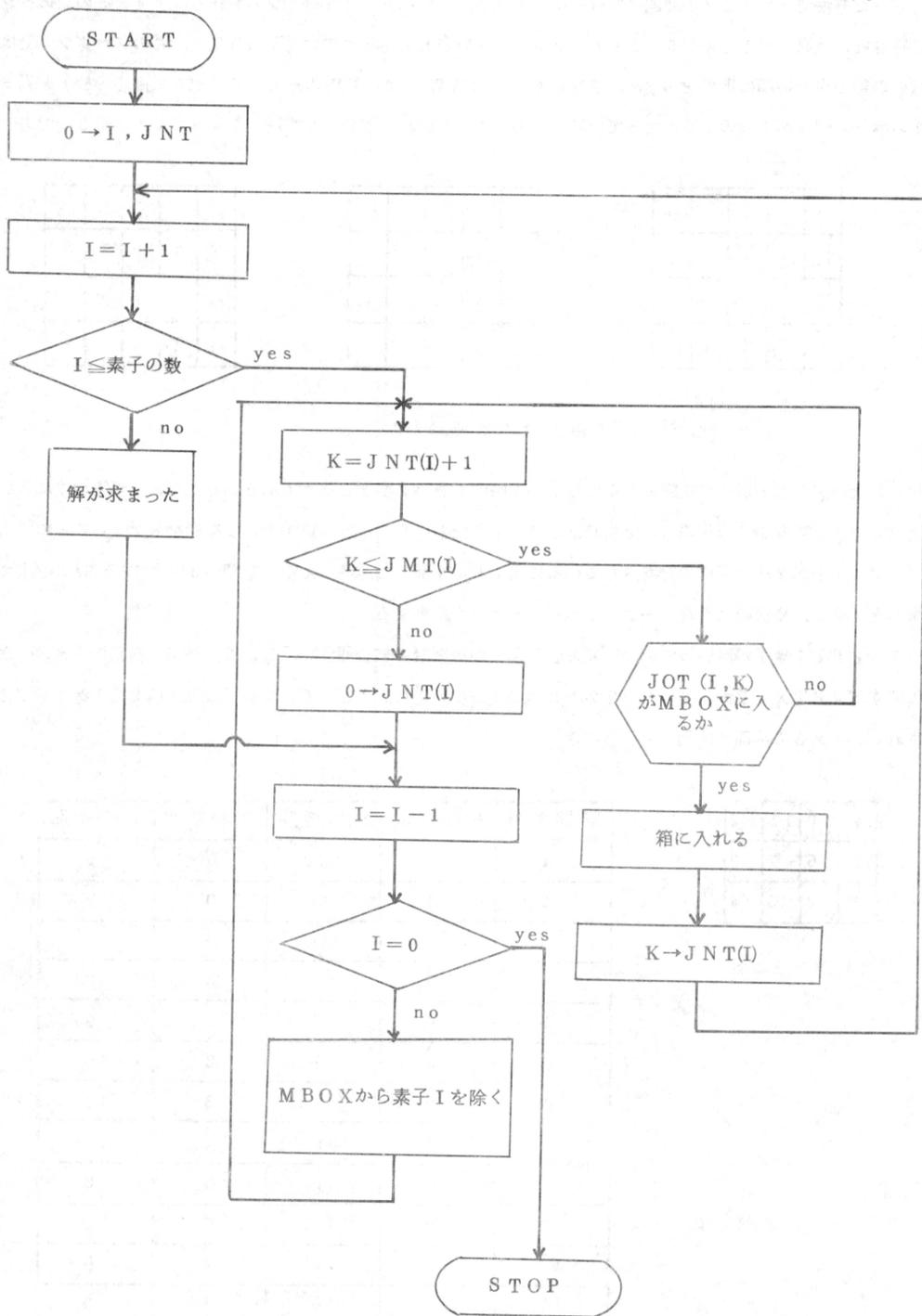


図5 PENTOMのフロー・チャート

3. ペントミノにおける論理的処理

2.で述べた方法でペントミノの問題は解けるはずである。これに従ってプログラムを作り $5 \times 4 \times 3$ の立体ペントミノを実行させた結果、HITAC5020Fで30分以上計算しても一つの解も得られない。このプログラムでは、素子は状態の数の少ない順に番号を付けた。すなわちIXTUVWZLNYPFの順である。組立て途中の様子を見ると、IXTの素子の関係が図6のようにになっている。このパターンでは、連続した空間21, 41, 46が3つであるから

15				20
11				
6				
I	I	I	I	I

35				40
31	X			
X	X	X		
21	X			

55				60
T	T	T		
46	T			
41	T			

図6 この組合せでは解はない

解は得られない。T素子が次の状態に移るには、約1時間ぐらい必要としそうである。そこで素子が箱の中に置いたとき、連続した空き空間が4以下のものを生じたときは、ただちに次の状態に移すという機能をもたせることにする。ペントミノでは素子が2次元平面で定義されているので、ある平面上で連続した空き空間が5以上であることが必要である。この操作をさせるため次のようなテーブル、サブルーチンを用意する。

テーブルRPT;素子はいっていない箱の各要素と他の要素の結合関係を示した表。箱が3次元のときは、X, Y, Z軸に関するRPTX, RPTY, RPTZの3つのテーブルから成っている。2次元のときはRPTX, RPTYで構成される。4×3の平面では図7のようになる。

9	10	11	12
5	6	7	8
1	2	3	4

(a) 4×3の面

空間番号	左側の空間	右側の空間	下の空間	上の空間
1	0	1	0	5
2	1	2	0	6
3	2	3	0	7
4	3	0	0	8
5	0	6	1	9
6	5	7	2	10
7	6	8	3	11
8	7	0	4	12
9	0	10	5	0
10	9	11	6	0
11	11	12	7	0
12	12	0	8	0

RPTX

RPTY

図 7

(b) RPTテーブルのデータ構造

テーブルMBOXA；この処理を行なうための箱。箱の中に素子が入っていないときはテーブルRPTと同じデータが入っている。

サブルーチンELASM；素子を箱MBOXAに入れる。例えば図7のようなMBOXAに2, 5, 6, 7, 10の状態にあるX素子を入れるとMBOXAは図8ようになる。

9	X	11	12
X	X	X	8
1	X	3	4

空間番号	左側の空間	右側の空間	下の空間	上の空間
1	0	0	0	0
2	0	0	0	0
3	0	4	0	0
4	3	0	0	8
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	4	12
9	0	0	0	0
10	0	0	0	0
11	0	1 2	0	0
12	11	0	8	0

図8 4×3のMBOXAにX素子を入れたときのMBOXAのデータ構造

サブルーチンELERS；MBOXAに入っている素子をMBOXAから取り除く。

サブルーチンPLASC；点Pに関して、ある面での連続した空き空間の数と、それらの点を求める。図8で点3に関してこの操作を行なうと4および点列3, 4, 8, 12, 11が求められる。

基本プログラムで、素子をMBOXに入れるとき、同時に素子をMBOXAに入れる。そしてMBOXで0が入っているアドレスについて、PLASCにより連続した空き空間の数を数える。この操作は、2次元の場合は、XY平面だけでよいが、3次元のときは、XY, XZ, YZの3平面について調べる。もしこの中に連続した空き空間の数が4以下のものがあるなら、解は得られない。したがって、MBOX, MBOXAから今箱に入れた素子を取り除く。

以上のアルゴリズムにより作られたプログラムでは、約15分に1つの割解が得られるようになった。I, X素子が図6の状態にあるときの解を図9に示す。これらの解を求めるに要した計算時間はH-5020Fで約10時間であった。IとXの可能な組合せは、鏡像関係を除いて66通りあるから、もし一組I, Xの組合せにおいて、解を求めるのに10時間必要とするなら、全部の解を求めるのに660時間必要ということになる。

WPPFN WYFFF YYYYY PPPLN WXTTT WZVVV LLLLN XXXTN WZZZV IIIII UXUTN UJUZY	WPPFN WYFFF YYYYY LPPPN WXTTT WZVVV LLLLN XXXTN WZZZV IIIII UXUTN UUUZY	PYZZ PYYYY NNVVV PFZLW PXTIT PNNVV FFZLW XXXTW LLLLW I1111 UXU1W UUULW
WFYVN WYYYY ZPPL FFFVN WXTTT WZPPL FVVVN XXXTN WZZPL IIIII UXUTN UJULL	WFZVN WYZZZ YYYYY FFFVN WXTTT WLLLL FVVVN XXXTN WLPPL IIIII UXUTN UOUPP	ZZPPN WYYYY WFYVL WZPPN WXTIT FFFVL WZZPN XXXTN FVVVL I1111 UXUTN UUULL
WFNNV WFFFV ZZFYV NNNLV WXTTT WZYYP LLLLL WXTTP WZZYP IIIII UXUTP UJUYP	LLLLV YYYYY ZYVVV ZPPLW ZXTTT ZNNNF ZPPPW XXXTW NFFFF IIIII UXUTW UOUFW	PPVVV YYYYY TTFV PPPLW UXYZZ UFFFF LLLLL XXXZW UTENN I1111 UXZZW UNNNW
NNVVV YYYYY TTFV LNNNV UXYZZ UFFFF LLLLL WXTTP WZZYP IIIII UXZZW UPPPW	ZZNNN WNNYL WFVVV WZPPP WXYLL FFFTV WZZPP XXXYL FTTTV IIIII UXUYL UUUFL	ZNNNP ZZPP FFZPP UWNNN UXWVV UFFWT UYYYY XXXYV UFTTT I1111 LXVVV LLLLT
ZWFFI WFFF WPNL ZYTTT ZXVVV ZPPL YYYYT XXXNV ZPPL IIIII UXUNV UJULL	PPFFW NNNW ZWWT PPFFF UXNNV UZTTT LLLLL XXXLV UZZYT IIIII UXVVV UYYYY	WUUPP WFFN ZWFF ZUOPP ZXVN ZYVVV ZUOPP XXXIN YYYVN I1111 LX1TT LLLLN
ZFPPP FFFPP FJUNW ZYVVV ZXVTW ZJUNW YYYYW XXXTW ZJUNW IIIII LX1TT LLLLN	VPZZF VLFFF VLYFN VPPZW UXYTW ULYNN VPPZZ XXXTW ULYNW IIIII UX1TT ULYNW	ZWPPP ZZPP LLZYP WVNNN UXVTF ULVYY WNNNF XXXTF ULVYF I1111 UX1TT ULVYF
PTWNN PNNNF YYYYY PTWVF PXZZF PYVVV TTTW XXXZF LLLLW IIIII UXUZZ UJULV	ZTQLV PPPLV PVVVV ZTWVF ZXVLF ZYYYY TTTW XXXLF ZNNNF IIIII UXULF UOUNN	P1FNV PFFV PVVVV PTZNV PXZZZ LLLLZ TT1NW XXXNW LYYYV I1111 UXUNW UOUYV
UTPPW NNNW UFWY UTPPP UXNNV UFFYY TTTZ XXXZV FZZY IIIII LXVVV LLLLY	UTNNW NNNW UFWY UTPPP UXPPV UFFYY TTTZ XXXZV FZZY IIIII LXVVV LLLLY	NN1V FNNV YYYYY FW1Z FXUOZ FYPPV W11T XXXUZ FPPPV I1111 LXUOZ LLLLZ
ZFWTN FFFYL FVVVL ZWVTN ZXYVN ZVPL WTTT XXXYN ZVPL IIIII UXUNV UJUPL	ZFWTV PLLV PVVVV ZWVTF ZXVLF ZYYYY WTTT XXXLF ZNNNF IIIII UXULF UOUNN	WZZIV WZLLV ZZVVV PP1TF WXYLF WYYYY P111 XXXLF WNNNF I1111 UXULF UOUNN
TFVVV FFFLV ZZFLV TTTW UXWVP UZWL YYYYY XXXYP UZL IIIII UXNNN UNNLP	PVWFN PVWFF WFFY PVTZN PXTZY LTTTY PVVVN XXXZN LLLLY IIIII UXUZV UOUZY	PP1V FPP1V ZZTTT FNNV FXUUY FZWWY NNVVV XXXYT FZZWV I1111 LXUUY LLLLW
NNZTV ZZZF ZYTT LNNNV UXFFF UYVW LLLLL XXXFV UYVW IIIII UXPPP UYWPP	WNNP WPPP TWZV TYNNN TXUUF TZZZV YYYYY XXXUF TZVVV I1111 LXUUF LLLLF	WNNP WNNN TWZV TYPPP TXUUF TZZZV YYYYY XXXUF TZVVV I1111 LXUUF LLLLF
FFPPP NNNP ZNNY UFFW UXWV UZVY UFTT XXXT UZZY IIIII LXVVV LLLLY	VZTTT UUTN LPPTW VZZF UXUNN LPPW VVZF XXXNF LPWF IIIII LXVNF LYYYV	NNWT FNN1 YYYYY FWWV FXZZT FVVVV FWPPP XXXT LLLLV I1111 UXUZZ UOULV
NNWT FNN1 YYYYY FPPW FXZZT FVVVV FPPW XXXT LLLLV IIIII UXUZZ UOULV	WVVV WTTT ZWNT ZVUF ZXFFF ZYFNN ZVUU XXXPP YYYVN IIIII LXPPP LLLLN	TWPP TTFL TLLL WPPP UXFF UZVVV WYYY XXXY UZZZV I1111 UXNNN UNNZV
TWPP TTFY YYYYY WPPP UXFF UZVVV WLLL XXXFL UZZZV IIIII UXNNN UNNZV		

図9 $5 \times 4 \times 3$ での解の一例

基本プログラム（プログラムA）とこのアルゴリズムによるプログラム（プログラムB）との処理速度を比較するため、図10-(a)の初期値でそれぞれのプログラムを走らせた。プログラムAでは(b)に4270秒、(c)に4693秒で、

	V	U	U	U
	V	U		U
	V	V	V	
I	I	I	I	I

		W		
		W	W	
			W	W

	X	T	T	T
X	X	X	T	
	X		T	

(a) 初期値

L	L	L	L	V
Y	Y	Y	Y	V
W	Z	V	V	V
I	I	I	I	I

L	Z	P	P	N
W	Z	Y	F	N
W	Z	F	F	N
U	U	U	F	F

W	Z	P	P	P
W	X	T	T	T
X	X	X	T	N
U	X	U	T	N

(b) プログラムA 4270秒
" B 223秒

N	N	Z	Z	V
F	N	N	N	V
Y	Y	Y	Y	V
I	I	I	I	I

W	W	Z	L	V
F	W	W	L	P
F	Y	W	L	P
U	U	U	L	P

F	Z	Z	L	V
F	X	T	T	T
X	X	X	T	F
U	X	U	T	F

(c) プログラムA 4693秒
" B 264秒

図 10

プログラムBでは(b)に223秒、(c)に264秒で達した。プログラムBは、その処理速度が、Aに比べ(a)-(b)で1.9倍、(b)-(c)で10倍となっている。

プログラムBで、連続した空間の数を数えるという処理には相当のステップ数が必要であり、素子番号の小さなときは、素子が次の状態に移るまでの時間が非常に長く、この処理に要する時間は問題に成らないが、大きな素子番号では、よけいな処理を加えない方が早くなるであろうと考えられる。そこで、A、B両プログラムの処理過程と処理時間を調べて見た。図11-(a)の状態から始める。次に処理されるのは、第9番目のN素子である。この状態から始めて最初に得られた解は図11-(b)であり、この解を得るためには、第6番目のW素子から上の素子の入れ換えが行なわれる。図11-(b)の解を得るまでの時間は、プログラムAでは279秒、Bでは44秒であった。これらで処理と処理時間がど

U	Z	V	V	V
U	T	T	T	V
X				V
I	I	I	I	I

U	Z	Z	Z	
X		T	W	
X		W	W	
X	W	W		

U			Z	L
U		T		L
				L
			L	L

(a)

U	F	V	V	V
U	T	T	T	V
X	W	Z	Z	V
I	I	I	I	I

U	F	F	F	Y
X	W	T	Y	Y
X	W	Z	N	Y
X	P	P	N	Y

U	W	F	N	L
U	W	T	N	L
X	Z	Z	N	L
P	P	P	L	L

(b)

図 11

のように成っているかを示したのが図 12 である。プログラム B では、太線に沿って 9-9-9-8-8-8-8 と処理される。上の括弧中の数値は、連続した空間の数を調べるための時間である。太線の上の数値は、A、B どちらのプログラムでも必要な処理時間である。プログラム A は、太線から下にかかっているような道筋で処理される。下の括弧中の数値は、太線から外れた処理に必要な時間である。図 12 から判断すると第 9 番目以上の素子の処理では、直接処理の方が早いようであるが、第 8 番目の素子の処理では、プログラム B の方が 2 倍以上早くなっている。B のアルゴリズムのための処理時間の平均値を測定してみると約 2.1 m 秒、処理を行なっている平均の素子番号は 8.5、空き空間が 5 以上になるものを見つける確率が 0.14 であった。プログラム B では、処理している平均の素子番号は 9.5 であった。これらは図 11-(a) から始まる 2000 データについての平均値であり、非常に解に近い所でプログラム B にとっては不利な条件での測定である。このような事を考えるとプログラム B は、非常に有効であることが分かる。なおこの測定は、HITAC 5020 で行なった。

4. 4次元ペントミノ

60は, $5 \times 3 \times 2 \times 2$

とかくことができる。いま4次元空間で各座標軸をそれぞれX, Y, Z, A軸とすると,

$$0 \leq x \leq 5$$

$$0 \leq y \leq 3$$

$$0 \leq z \leq 2$$

$$0 \leq a \leq 2$$

で囲まれる4次元直方体を12個のペントミノ素子で組立てられるかという問題が考えつく。4次元直方体を感覚的に理解することは困難であるが, 計算機中での処理は, 2次元, 3次元の問題を抜かうのも4次元を抜かうのも全く同じでよい。

12の素子の内X, T, V, W, Z, Fの6つの素子は, 3×3 以上の面を必要とする。したがって, これらの素子はXY面上にしか置くことができない。また素子IもXY面上に置く必要がある。12の素子のうち7つまでがXY面上に置かねばならず, ZA面上には置ける素子が一つもないという厳しい条件が付けられているので, 解がないであろうと予想される。

まず各素子が $5 \times 3 \times 2 \times 2$ の直方体内でいくつの状態を取りうるかを調べると, 表3のようになる。I素子は12

	F	I	L	N	P	T	U	V	W	X	Y	Z
$5 \times 3 \times 2 \times 2$	96/8	12/1	160/12	160/12	384/24	48/4	192/12	48/4	48/4	12/1	160/12	48/4
1	72/8		136/12	136/12	302/24	36/4	146/12	36/4	36/4	9/1	136/12	36/4
2	72/8		128/12	128/12	280/24	36/4	150/12	36/4	36/4	9/1	128/12	36/4

表3 $5 \times 3 \times 2 \times 2$ の箱におけるペントミノの状態の数と向きの数

の状態をもつが, 鏡像関係を考慮すると図13の2つの状態について調べれば良い。

$a = 1, z = 1$

I	I	I	I	I

(a) 状態 1

$a = 1, z = 2$

$a = 2, z = 1$

$a = 2, z = 2$

I	I	I	I	I

(b) 状態 2

図13 $5 \times 3 \times 2 \times 2$ ではIの状態が(a), (b)の場合だけ調べれば良い。

I素子を固定素子として問題を解くことにすると、入力データは次のようになる。

状態 1

BOX / 5 * 3 * 2 * 2

RESERVE / (I) (1, 1, 1, 1), (2, 1, 1, 1), (3, 1, 1, 1), (4, 1, 1, 1),
(5, 1, 1, 1)

ELEMENT / (X) (2, 1), (2, 2), (2, 3), (1, 2), (3, 2)

.....

状態 2

BOX / 5 * 3 * 2 * 2

RESERVE / (I) (1, 2, 1, 1), (2, 2, 1, 1), (3, 2, 1, 1), (4, 2, 1, 1),
(5, 2, 1, 1)

ELEMENT / (X) (2, 1), (2, 2), (2, 3), (1, 2), (3, 2)

.....

これらの各状態における11個の素子の状態数は、表3のようになる。

これらの計算には、HITAC 5020Fで約1時間を要したが、予想通り5 * 3 * 2 * 2の直方体を作る組合せは得られなかった。

5. おわりに

ペントミノの一般化した解法と、その論理的処理の一つについて考察した。本報告で述べた手法のポイントは、箱の中で取り得る素子の状態を調べることである。この種の組合せパズルには、ペントミノの他に正三角形等を要素として作られたものなどがあるが、これらのものも同じ手法で解が求められるであろう。これらをも含んだこの種の問題の一般化を行なうことが今後の課題である。また、これはどのようにして、各々のパズルの性質を記述するかという問題でもある。

連続した空き空間が4以下であるかどうかという処理により、立体ペントミノの処理速度を大幅に早めた。このアルゴリズムをそのまま使って、残りの空間がちょうど5に成るとか、特定の形に成るとかした場合、その空間を埋める素子があるかどうかを調べ、もしあれば先にその空間を埋めるという処理が行なえる。しかし、このような論理的な処理は、多くの時間を要するであろうから、状況に応じて使い分けなければかえって処理時間を長くするという結果にも成りかねない。

参考文献

- (1) 一松信：ゲーム・パズルと電子計算機，信学誌Vol.55, No.1 (1972) pp.32-34
- (2) 池田敏雄編：電子計算機概論—計算機システムと情報処理—(昭和43) pp.260-263, オーム社

本 PDF ファイルは 1965 年発行の「第 6 回プログラミング—シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の検索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場(=情報処理学会電子図書館)で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者(論文を執筆された故人の相続人)を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者搜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思えます。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長(tsuji@math.s.chiba-u.ac.jp)までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日～2021 年 3 月 19 日

掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程

<https://www.ipsj.or.jp/copyright/ronbun/copyright.html>