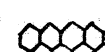


Tetrahex と Trihex に於ける 'Hexagon' の詰合せ

野 下 浩 平

(電気通信大学 電子計算機学科)

問題： 7 つの Tetrahex と 3 つの Trihex (図 1.) と 図 2. の
'Hexagon' (六角形) に詰合せる 解の総数はいくつか？



bar 'I'



worm 'J'



wave 'S'



arch 'U'



propeller 'Y'



pistol 'P'



bee 'Z'



straight '1'



bent 'V'



triangular 'D'

図 1. Tetrahex と Trihex

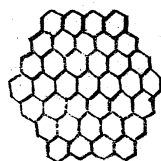


図 2. 'Hexagon'

答： 12,290 通り。

(但し、回転、裏返しによつて同一になる解は、繰返しとは数えない。)

この問題の由来については、[1] 参照。

解の総数は、バックトラック (backtrack, ‘行きつ戻りつ’) の探索方法 ([2]) に基くプログラムにより、計算機で求めた。

ここでは、プログラムの作成上の着眼点について略述する。

全体の方針としては、未解決の問題であったので、プログラムを簡単に作り、全体として早く結果を求めることである。

‘Hexagon’ (盤 (board) という) に、図3. に示すように一連番号をふり、その番号順に 10 個の駒 (piece) を次々に置いていくことにより、すべての置き方を数えあげる。

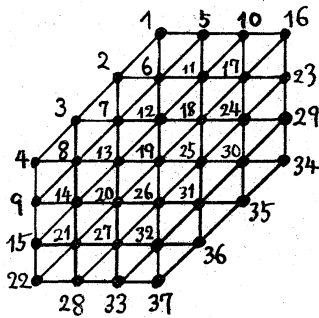
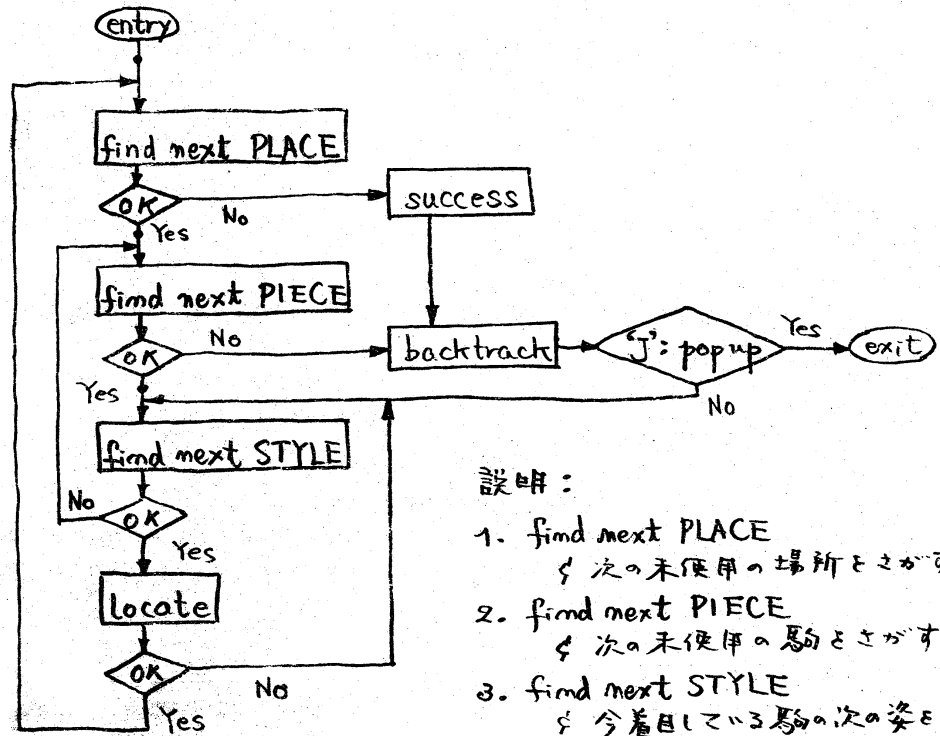


図3. ‘Hexagon’ (盤) の番号づけ[†]

互に対称 (回転と裏返し) な解を除くため、駒 ‘J’ の置き方と図6. のように 18 種類に分けて、各場合 (初期配置) について調べる。一つの ‘J’ の初期配置に対してのプログラムの概略と図4. に示す (このプログラムの形は、特に今の問題に限らず、バックトラック法一般に通じるものである。こ

[†] 印刷形式に合わせて座標軸を定めている (cf. 図7.)。

の図を直接コーディングすることにより、制御の流れに関わるオーバーヘッドも (goto により) 極小にできるであろう。



説明:

1. find next PLACE
↳ 次の未使用の場所とさがす
2. find next PIECE
↳ 次の未使用の駒とさがす
3. find next STYLE
↳ 今着目している駒の次の姿とさがす
4. locate
↳ 盤に駒を置けるか否かを調べる。
もし置けるなら、実際に置く
5. success
↳ 1つの解が見つかった
6. backtrack
↳ 直前の状況に復帰する
7. 'J': popup
↳ 駒 'J' までほかあったか?

図4. バックトラック法による探索プログラム
(駒 'J' の1つの初期配置に対して)

次にプログラムで扱うデータはついで述べる。

静的な(探索中変化しない)データは、次の通りである。

- i) 盤の各場所(37ヶ所)に対して、駒が置かれる際に関係する(9種類の)まわりの場所の一覧表。
- ii) 各駒(10ヶ所)に対して、その姿を示すデータの集り(4.3)の先頭へのポインタの一覧表。
- iii) 駒がおかれる際の姿(style; 向きと裏表)も、その起点(盤上で番号が一番若くなる駒上の位置; ①で示す)から出発して、各位置を相対位置で表わしたデータの覧表。
例えば、駒‘Y’に対しては、姿は、2種類であり、各姿は、図5.の点線で示される相対位置を示す番号の3つ組で表わされる。

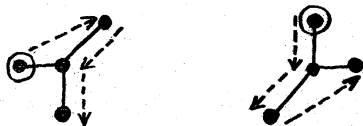


図5. 駒‘Y’の姿の表現

動的な(探索中変化する)データは、次の通りである。

- iv) 盤の使用状況(未使用を示す値が置かれてゐる駒の番号; 印刷にも用いる)をいれる大きさ37の1次元配列。
- v) 駒の使用状況(使用中か否かを示す値)をいれる大きさ10の1次元配列。

次の3つは、添数Iで指される深さ10のスタックである。

- 6) 第1番目におかれた駒(の番号),
- 7) その駒の姿(の番号),
- 8) その姿の起点の盤上の場所.

上記データは, 冗長なものもあるが, 探索中に無駄な手間と省くために導入している.(補助的なデータが他に少しある)

バックトラック法については,

"制限の強い場所からまず試せ"

という'哲学'がここでも有効である. 駒'J'の初期配置を図6. のように定めたのは, '複雑な形状'が早い段階で現われるように考慮したものである.

盤の場所の選択を(盤の状況によって)動的に計算するとか, 動的な対称性のチェック(詰合せパズルでは, 局所的な対称性も数多く現われるものが多い)等, 速度向上のため工夫が思いつくが, この問題については, その規模からみても, 静的なもので十分であった.

プログラムは, FORTRAN で書いている(使用計算機は, HITAC 8350). 大きさ(カード枚数)は, 次の通り.

プログラム全体 約 250 枚

† 使用した計算機システムの情報により, プログラムの実行を何度か中断した. '復旧'は, LPの印刷結果を見て Card で途中のデータを再入力する原始的なものであるが, 6), 7), 8)の内容(と'J'パタメ等)を用いると簡単に復旧できる.

- 内訳：・MAINと初期設定----- 80 枚
- ・探索部分 (cf. 図4)----- 120 枚
 (復旧手続き ----- 20 枚)
- ・印刷 ----- 30 枚
- (うち 宣言, 証明等は, 60 枚)

図6. は, 駒 'J' の初期配置と各々に対する解の数である。

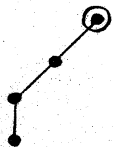
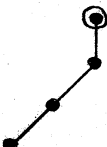
起点○ がみかへる 盤上の場所 姿				
1	2,400	499	169	138
2	6,024	1,351	173	0
5	238	243	196	141
6	172	113	130	139
11	48	116	—	—

図6.. 駒 'J' の初期配置と解の数

解の総数は, 川合慧氏 (東大・理), 竹内郁雄氏 (電々公社通研) によってその後各々求められたが, 異なる人が異なる計算機で異なるプログラムによって同じ結果を得たので, "総数" が正しいことは, 確実であろう。(数えあける以外に検証の方法がない時の "検証の実例" である。) なお要し

た計算時間は、いずれも数時間である。

図7. に、駒 'J' の 1 つの初期配置に対するいくつかの解の印刷例を載せる。

JYVV	JYVV	JYUU	JYUU	JYUU	JYUU	JYUU	
JUYYV	JUYYV	JSYYU	JSYYU	JSYYU	JSYYU	JSYYU	
JUYSDD	JUYPPP	JSYPUV	JSYPUV	JSYPUV	JSYPUV	JSYPUZ	JS
IJUUSD	IJUUPSS	IJSPPP	IJSPPP	IJSPPP	IJSPPP	IJSPPZ	IJS
IZZSPP	IDSS1	ISZZDV	IS111V	IS111V	ISDZZV	ISPVDZ	ISF
IZZSP	IDZZ1	IZZDD	IZZDD	IDDZZ	IDDZZ	IVVDD	IVV
I111	IZZ1	I111	IZZD	IDZZ	I111	I111	I11

JYUU	JYUU	JYUU	JYUU	JYUU	JYUU	JYUU	
JSYYU	JSYYU	JSYYU	JSYYU	JSYYU	JSYYU	JSYYU	
JSYPUD	JSYPUD	JSYPUZ	JSYZUD	JSYZUP	JSYZUP	JSYZUV	JS
IJSPPDD	IJSPPDD	IJSPPZ	IJSZZDD	IJSZZPP	IJSZZPP	IJSZZPV	IJS
ISVVP1	ISVVP1	ISDVPZ	ISVZP1	ISVZP1	ISDZPV	ISDZPV	ISD
IVZZ1	IZZV1	IDDVV	IVPP1	IVDD1	IDDVV	IDDPP	IDD
IZZ1	IZZ1	I111	IVP1	IVD1	I111	I111	IVV

JYUU	JYUU	JYUU	JYUU	JYUU	JYUU	JYUU	
JSYYU	JSYYU	JSYYU	JSYYU	JSYYU	JSYYU	JSYYU	
JSY1UD	JSY1UD	JSY1UV	JSY1UZ	JSY1UZ	JSYVUP	JSYVUZ	JS
IJS1VDD	IJS1VDD	IJS1DDV	IJS1PZZ	IJS1DZZ	IJSVPP1	IJSVPZZ	IJS
IS1PVV	IS1VPP	IS1PDV	IS1PPZ	IS1DDZ	ISZVP1	ISDVPZ	ISD
IZZPP	IZZVP	IZZPP	IVPDD	IVPPP	IZZD1	IDDPP	IDD
IZZP	IZZP	IZZP	IVVD	IVVP	IZZD	I111	I11

図7. 解の印刷例

参考文献

- [1] 一松信, “計算機によるプロパズル — とくにテトラハックスとポイントキューブ,” 京都大学・数解研・講究録 98 (1970), 3-11.
- [2] S. W. Golomb, et. al., “Backtrack Programming,” JACM, 12, 4 (1965), 516-524.