数理解析研究所講究録
第263巻 1976年 9-22

N-queens game, TANGRAM & TAIT's problem

電電公社武蔵野通研　竹内郁雄　奥乃博

1. はじめに

いろいろなpuzzleやgameを計算機でやらせることが大変楽しいことであるのは証明の要らない事実のようだ．たとえばパラメータnに依存してサイズの大きくなるようなpuzzleでsmall nについて有望風に成立するようなconjectureを，人間の手には届きかねる程度のa bit larger nで（軽いフットワークで）打ち破るのは，これを快感と呼ばずんば何だろう！　しかし単にこれだけで手と変え品を変え新しいpuzzleに挑戦する価値があると言ってはならないだろう．というのは少なくとも著者の場合，それは常にプログラミングにおける一つの発見をもたらすからである．特に表題の後二者については，著者が完成した（と信じている）新しいリスト処理言語LIPQ（リップク）を使ったので大変得るところが多かった．本報告では，これらの問題に関して得られた結果とともに，プログラミング上の諸問題についても触れてみたいと思う．

2. N-queens game

この問題はGPCCでauthorizeされたものではなく，著者が野崎昭弘先生からの私信で半強制的（御当人はそうは言わないだろうけれど）にやらされることになった問題である．問題は有名な8-queens problemの二人ゲーム版であって，図1のような盤面に交互にチェスの女王を置いていき，どちらかがすでに盤上にある女王のどれにも捕獲されないように女王を置けなくなったら負けである．（女王は将棋の飛車と角行を併せた能力を持つ駒である．）

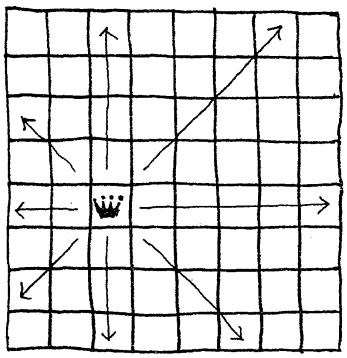

図1，
（図は8×8だが，一般にはn×n）

このgameが先手必勝か後手必勝かの解析はnが奇数の場合きわめて容易である——即ち先手必勝．nが偶数の場合，今日までのところシラミつぶし以外に手はなさそうである．著者がプログラムを組む以前に次の結果が得られている．

n = 2, 4, 6, 8	先手必勝	（野崎氏，手で数時間）
n = 10	後手必勝	（有澤誠氏* HITAC-8410 100分）

この問題を解析するプログラムは，backtrack programming

*）電技総研

では必ず引用される 8-queens problem のプログラムとほとんど同じであるからここでとりたてて述べるほどのものではない．著者は pdp-11/20 という 16 bit/word のミニコンでプログラムを組んだのであるが，hardware stack 機能があまので recursive call のあるプログラムをアセンブラで組み立てるのが大変快適である．そこで浮いた労力を専ら局所的に最も多く実行される部分の code optimization に費したのであるが，それが実行時間短縮に寄与したところはわずかであろうと考えられる．ただ面白いと思われるの，右上がりの対角線*と左上がりの対角線が他の駒の効き筋になっているかのチェック（この意味についてはたとえば [1], [2] などを御覧いただきたい）をただ一回の Boolean test で済ませたことである．すなわち n×n の盤には 2n-1 本の右上がり対角線と同じく 2n-1 本の左上がり対角線があるが，これらの対角線に対応する語（つまり array の element）に

$$\begin{cases} \phi & : \text{その対角線上に女王がない場合} \\ -2 \text{ or } -1 & : \text{その対角線上に女王がいる場合，右上がりには} -1, \text{左上がりには} -2. \end{cases}$$

を格納するのである．このようにすると現在話題になっているマス目に対応する二つの対角線のデータを比較するだけで

*) 正確な語法ではないが意味は了解していただけるだろう．

両方のチェックが行なえるわけである.

着着のプログラムの実行結果は次の通り.

n = 8　先手必勝　(2秒以下)

n = 10　後手必勝　(25分)

n = 12　後手必勝　(6時間半)

要するに野崎先生の予想はもろくも崩れ去ったのであった.

3. TANGRAM

正方形の板を図2のように7つの片に分割して切り離す. これらの片を組みあわせて平面図形を作る問題(遊び)をTANGRAMと言う. ここで扱うのはTANGRAMで出来る五角形は何種類あるかという問題*)である. (その五角形の内部構造が異なったものがあっても重複して数えない) 既に人手によって53種類まで数え上げられていて, 多分これが全数であろうという予想もあるので, 問題のサイズは思ったよりも小さいと考えられるが, この種の図形のかたちを扱う問題は計算機の苦手とするところである.

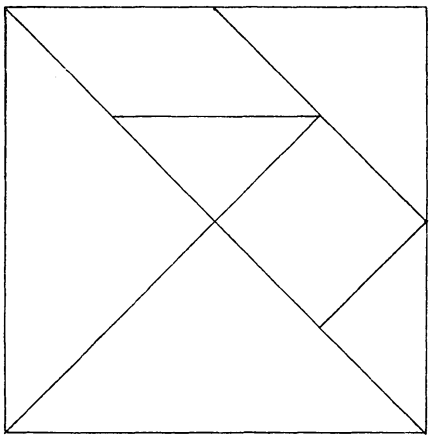

図2. TANGRAMの用具

*) 便宜上 TANGRAM-5 と呼ぼう.

TANGRAM-5の難かしさは，普通の箱詰めパズルが，いわば閉じた環境での探索であるのに対し，開いた環境での探索と伺えるところであろう．だからある片を固定し，その周りに順次残りの片をくっつけていって，最後に5角形が出来たかどうかを判定するようなプログラムは，その技術的細部の構成の難かしさもさることながら，相当能率の悪い，かつ全数探索の証明が困難なプログラムになりかねない．要するに手作業で解くときの困難度よりも大きい困難度がプログラム化にあたって要求される性質の悪い問題といえよう．

著者のとった方法は，名付けるならComputer Aided Enumeration (CAE) とでも言えよう．すなわち，まずプログラム化の容易なところをプログラムし，その結果を見てまた必要な再プログラミングまたは別のプログラミングを行なう．これは最終的な結果の信頼性をある程度落とすことになるが，このような性質の問題では手作業とプログラミングのよいtrade offをとってやるほうが全体的な能率は上がることになる．

CAE法のtoolとしては，会話的なセンスで使える高級言語が最も便利である．幸い著者が開発したリスト処理言語LIPQがコンパイラを含めて一段落したので，性能評価および虫くだしを兼ねてLIPQでTANGRAM-5のプログラムを組んだ．(この問題に関して言えばリスト・データを使う必然性はほとん

どないのだが)．ここで簡単にLIPQの紹介をしておく．LIPQはStanfordのLISP 1.6にほぼupper compatibleな言語で，言語仕様とで大きく違うところは，PROGの入れ子構造がALGOL 60のblock構造と同じ機能をもつこと，binary cellの他にcar, cbr, ccr, cdrの4つのfieldを持つquaternary cellを持つこと，数に浮動小数点がないことなどである．システムはpdp 11/20で稼動しており，二次記憶がフルに活用されている．(たとえばbinary program areaはoverlay mechanismにより事実上無限の大きさになっている．) なお，LIPQの使用について特筆すべきことは，LIPQの出力ルーチンがcycleやcommon sub-listを含むような任意のリスト構造を見易いformatで線型に打ち出せることであり，これがdebugやCAE methodに大きく寄与していることである．

CAE法によったため，TANGRAM-5のプログラムは，かなり手抜きインチキ的なものである．基本的に元の正方形に等しい面積をもつ5角形を求めるだけのプログラムである．TANGRAM-5の図形を考える限り，その上での代数は整数環に$\sqrt{2}$を付加した環$I(\sqrt{2})$で考えれば十分であるから，$I(\sqrt{2})$上での和，差，積，零テストをまずチョコッとつくる．(LIPQでは皆一行ないし二行のプログラム) あとは辺を一本一本生やしていき，4本目で五角形になり得るか否かをテストし，

OKなら最後の辺を生やしその面積と周囲の長さ（このときだけは1を$\sqrt{2}$ともに掛算して計算）が所定の条件をみたしたとき当該五角形を出力する——この関数を（特にその必要もないが）recursionを使って作る．このとき辺を生やすのに凹角は全体で一個しかないとか，一本の辺の長さは高々これだけというような条件を入れて不要な計算（のみならず出力）をしないようにする．ただし対称解のcheckは無理に完全なものにはしない．（CAEの第一回試行によればこうしても高々50個位しか解が出ず，うち対称解として余分なものは20個以下だからあとは絵を書いて人手でふるいわけることにする．プログラムで完全にやろうとすると結構面倒）これの出力結果（中間結果）が得られればあとは各々についての箱詰めパズルである．ただし解があるかないかだけの答が得られればよい．ところが，絵を書いてにらんでみると，ほとんどが5秒以内の観察で解けてしまうのである．要するにこれから先も人手でやった方がはるかに能率がよい．というわけでsnugな解*は総て求まった——計22個（予想通り）

non-snugな解についてもほぼ同様のapproachを行なう．す

---

*)各片のピッタリ合っている辺の頂点が互いにピッタリ合っているか，相手の辺の中点にあっているようなTANGRAM図形をsnugといい，そうでないものをnon-snugという．

なぬち，TANGRAM-5における non-snug解は，三角形（直角二等辺しかない）と三角形，あるいは三角形と四角形を組み合わせることによって出来る（証明は容易）　だからsnug解を求めたプログラムの中の5という定数を3あるいは4に変え面積の条件を緩めてやればよい．しかし三角形についてはプログラムをなおして走らせる時間（計3分位か）よりも，手でリストアップした時間の方が短かい．だから四角形だけを計算機で求めた．面積の条件は，元の四角形から各々の三角形をとり除いた残りの面積のどれかに等しくあればよし，というふうに変える．あとはsnug解のときと同様，注意深い観察のみである．これも予想通りの解（31個）が得られ，結局TANGRAM-5の解の総数は53個であることが確かめられた．

尚，プログラムの大きさは種々の初期化関数を含めてLP用紙2枚，実行時間はコンパイルされたもので，五角形が42分，四角形が3分（含LP印刷時間）位である．（インタプリタでは走らせてないがこの30～60倍かかると思われる．）残念乍ら，問題が簡単すぎたかあるいは人手が出しゃばり過ぎたかのため，本格的CAEには程遠い感であったが，CAEというのが一つの有効な方法であることを確信した次第である．少なくともこれは人間側に楽しみを残している点で楽て難いのではなかろうか．

4. TAIT's problem

図3のように白と黒が交互に並んだ状態から、「必ず隣り同志の二個の石を一緒に空いている場所に移す」という規則に従って石を動かし、白と黒とが各々一団となるようにする最少手順を $n=3$ から $n=10$ まで網羅的に求めよというのがこの問題である。

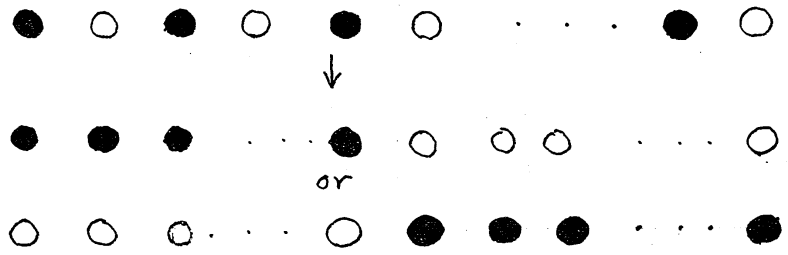

図3　TAIT's problem の initial state と final state

一般に $2n$ コの石がある時には、n 手で出来ること、及びその手順の個数は数学的に求まっているのであるが［3］、計算機で、その手順を求めるためには、$(2n)^n$ の木を探索しなければならないので、良い枝刈り algorithm を用いないと有限時間で解を得ることができない。

石を動かした時に、その状態がどれ程最終状態に近いかという量 difference を次の様に定義する。

まず、石の並んだ状態を次の様に表わす。

$\alpha_0\ \alpha_1\ \alpha_2 \cdots \alpha_{2n+1}\ \alpha_{2n+2}\ \alpha_{2n+3}$　（空地○も含めて考える）

α_0 = end　　α_{2n+3} = end

difference D は：

$$D = \sum_{i=0}^{2n+2} diff(\alpha_i, \alpha_{i+1})$$

where

$$diff(\alpha,\beta) = \begin{cases} 0 & \text{if } \alpha = \beta \\ 0 & \text{if either } \alpha \text{ or } \beta \text{ is end} \\ 1 & \text{if } \alpha \neq \beta \ \&\ \alpha \neq \circ \ \&\ \beta \neq \circ \\ diff(\alpha,\gamma) & \text{where } \gamma \text{ s.t. } \alpha \circ\circ \gamma \text{ if } \beta = \circ \\ diff(\delta,\beta) & \text{where } \delta \text{ s.t. } \delta \circ\circ \beta \text{ if } \alpha = \circ \end{cases} \quad (1)$$

と定める。

<u>lemma</u>　一手で減る difference は高々２である。

<u>Theorem</u>　TAIT's problem で石の数が $2n$ の時，$(n-1)$ 手では解は得られない。

［証明］初期状態での一手では difference は高々１減らないことと lemma から直ちに証明できる。 ∎

(1)の様に difference を定義すると、(a), (b) の２つの状態は

● ● ・・・ ● ○ ○ ・・・ ○ ◌ ◌ (a)

● ● ・・・ ● ◌ ◌ ○ ○ ・・・ ○ (b)

区別がつかないので，両端以外の◌◌の組も１と数える様に difference を定義しなおす。　この difference に於ては，一手で difference が３減少することがあるが，次の一手では高々１しか減少しないこと，初期状態での一手では，difference は

減少しないことがわかるので、i手動かした後のdifference Dは

$$D \leqq 2(n-i)+1 \qquad (i>1) \qquad (2)$$

を満足していないと最小手順の解とはならないことが言える。

TAIT's problemを解くプログラムは次の様な構成になっている。

(I) (2)で枝刈りを行なうdepth-first-search.

(II) 対称解を除くため、空地は右端に一組置く。

(III) 最終状態かのチェックはdifferenceが1かどうかでチェックする。

(IV) 各状態は次の様に表現する。

(state parent-stateへのpointer 次にexpandすべき箇所を示すpointer level)

計算は次の様に進行し、

```
(((($1:1 1 2 2 1 2 0 0 1 2 2 1)
  ($2:(1 0 0 2 1 2 1 $3:2 1 2 2 1)
   ($4:(1 2 $5:1 2 1 2 1 2 1 2 0 0) NIL $5 0)
   $3 1 )
  $1 2 )
 $2 $4 )
```

$n: は一種のラベルである.

```
(((($1:1 1 2 2 2 2 2 1 1 0 0 1)
  ($2:(1 1 2 0 0 2 2 1 1 2 $3:2 1)
   ($4:(1 1 2 2 $5:1 2 0 0 1 2 2 1)
    ($6:(1 0 0 2 1 2 1 $7:2 1 2 2 1)
     ($8:(1 2 $9:1 2 1 2 1 2 1 2 0 0) NIL $9 0)
     $7 1 )
    $5 2 )
   $3 3 )
  $1 4 )
 $2 $4 $6 $8 )
```

解が得られたら次の様に出力する。

```
((1 2 1 2 1 2 1 2 1 2 0 0)
 (1 0 0 2 1 2 1 2 1 2 2 1)
 (1 1 2 2 1 2 0 0 1 2 2 1)
 (1 1 2 0 0 2 2 1 1 2 2 1)
 (1 1 2 2 2 2 2 1 1 0 0 1)
 (0 0 2 2 2 2 2 1 1 1 1 1) )
```

得られた結果を表1に掲げる。

n	最小手順の数	計算時間 (LIPQ プログラム コンパイルして走らす disk に出力)
4	1	2秒
5	1	7秒
6	1	27秒
7	2	2分13秒
8	16	11分1秒
9	32	56分57秒
10	96	5時間23分30秒

表1. 一組の空地を用いた時の解

n	最小手順の数
3	1
4	1
5	1
6	1
7	4
8	16
9	32
10	96

補表. 二組の空地を用いた時の解

このプログラムは、(II)で述べた様に空地を一組しか用いていないので、二組以上の空地を用いる解を求めることはできない（$n=3,7,11$ではその様な解が存在する。$n\equiv 3 \bmod 4$！）

これらの結果が得られた後、文献3が発見された。これに依ると、TAIT's problemは「おしどりの遊び」と呼ばれて江戸時代（1713年）の文献に収められている。おしどりの遊びの解の表が $n=36$ まで挙げられており、$n=23$ の時、最少手順は 7838208000000 通りある。この文献では、一般解を求めるだけでなく、「数学的というか理論的というか

Nおしどりを片付ける方法を求めるのでなく、一種の遊びとして片付ける一つの方法がある」として「実際の遊びとして解く場合の手順」を暗記する暗記法についても書いている。それは、「品(シナ)は嶮(ケワ)く虫食(ムシク)い。光陰矢(ヤ)の如(ゴト)し、8の暗記法は無し・・・」となっている。

この文献も空地を2組以上用いる手順については触れていない。　n=10までに関しては表1と同じである。

5. プログラミング上の所感

ここでは本題から少し離れるがLIPQでのプログラミングについての所感を述べてみたい。まず言えるのは、一般にpuzzleを解くプログラムをリスト処理言語のような遅いプログラミング言語で作っても意味がないだろうというのは迷信だろうということである。3時間でよいものが6時間かかる(五十歩百歩!)というのでまずいのでなければ高級言語のmeritは絶大である。また、LIPQの場合、source programを常にきれいな書式で見ることが出来るのでLisp系言語で一般に要求される括弧の数え合わせがまったく要らないので虫取りの手間が著しく軽減される。さらに虫取りのため、あるいは上に述べたCAEのためいろいろな途中結果を打たせることが極めて容易である。これはLIPQの扱うデータが単一であって、常

に印刷可能であるという事実に基く．それゆえ会話型プログラミングとバッチ型プログラミングの中間のようなプログラミングが出来，多忙な時など精神衛生が大変良いのである．またプログラムの修正が容易なので，アルゴリズムそのものに力を注ぐことが出来る．実際，ちょっとしたアルゴリズムの変更が実行時間を変える（場合によっては数十倍）ことが身にしみて感じられるという意味で，プログラムを作るというよりアルゴリズムを作るという印象をもつことが出来るわけである．（もっともこれは高級言語たるものの資格なのであるけれど）

謝辞　御指導いただいた池野信一室長に感謝します．なお，池野室長は $n = 7, 11$ のTAITの問題にて空地を二組使う解の存在することを著者に教えて下さった．

文献

[1]. Dahl, Dijkstra & Hoare: Structured Programming
(Academic Press, London & New York 1972)

[2]. Dijkstra: A Short Introduction to the Art of Programming.
(Mimeographed Note 1971)

[3]. 森山善雄：おしどりの遊び　（がり版刷り，1958）