ピラミンクスの群論的考察とその解法 (金子 晃)

GAP という有限群論の計算ソフトと，OpenGL という３次元グラフィックスのプログラミングライブラリを用いたピラミンクスの解法分析および動作表示を卒業研究で取り上げた．このゲームは群論の教材として最適なので，『応用代数講義』のサポートページのおまけとしてここに紹介します．

1 序

ピラミンクス (Pyraminx) は，ルービックキューブのテトラ (正４面体) 版で，ドイツの数学者 Uwe Meffert により発明された．下図 1 のように面に番号を振り，これに対応する配列 FACE[] を用意して，ピラミンクスの状態を，現在位置 i の面に有る実際の面の番号 j をこの配列に記録する FACE[i]$=j$ ということで表現すると，ピラミンクスの動きは S_{36} の部分群である置換群とみなせる．実際に動きを調べるため，ピラミンクスの構造を細かく見ると，その部分テトラは，

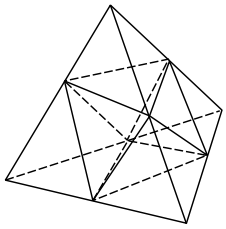

図 1

(1) 頂点の４個のテトラ
(2) 辺の中心にある６個のテトラ
(3) １面しか見えない１２個の正３角形

に分類されるが，最後の正３角形は，実は３個ずつペアになって正８面体を成して一緒に移動することが分かる．これは，正４面体の角を，図 1 のように辺の中点を通る４枚の平面により切り落すと，正８面体が残るという，初等数学の有名な結果に対応している．

この置換群は，以下のような，いずれも位数が３の基本置換より生成される. ここで，$A1$, $A2$, $A3$ は, それぞれ頂点 A から見て，１段目，２段目，３段目を図 2 の方向 (すなわち，右利きの人が右手で回しやすい右ネジの方向) に 120° 回転させる操作に対応する置換を表す. 他の頂点についても同様である．各々は, 位数３の独立な巡回置換のいくつかの積から成っている.

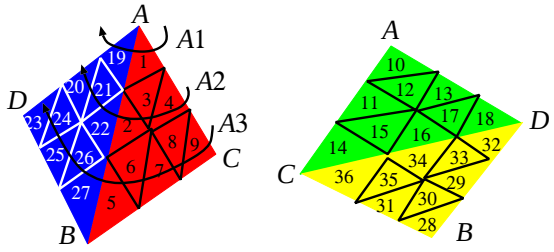

図 2

$$A1 = (10, 1, 19)$$
$$A2 = (11, 2, 20)(12, 3, 21)(13, 4, 22)$$
$$A3 = (14, 5, 23)(15, 6, 24)(16, 7, 25)(17, 8, 26)(18, 9, 27)(32, 36, 28)(33, 35, 30)(34, 31, 29)$$
$$B1 = (27, 5, 28)$$
$$B2 = (25, 2, 31)(26, 6, 30)(22, 7, 29)$$
$$B3 = (23, 1, 36)(24, 3, 35)(20, 4, 34)(21, 8, 33)(19, 9, 32)(10, 14, 18)(13, 11, 16)(12, 15, 17)$$
$$C1 = (9, 14, 36)$$
$$C2 = (7, 11, 34)(8, 15, 35)(4, 16, 31)$$
$$C3 = (5, 10, 32)(6, 12, 33)(2, 13, 29)(3, 17, 30)(1, 18, 28)(19, 23, 27)(21, 24, 26)(20, 25, 22)$$
$$D1 = (23, 32, 18)$$
$$D2 = (20, 29, 16)(24, 33, 17)(25, 34, 13)$$
$$D3 = (19, 28, 14)(21, 30, 15)(22, 31, 11)(26, 35, 12)(27, 36, 10)(1, 5, 9)(3, 6, 8)(2, 7, 4)$$

以下，$A1, B1, C1, D1$ を１型の回転，$A2, B2, C2, D2$ を２型の回転，$A3, B3, C3, D3$ を３型の回転と略称する．これらの置換群の元としての効果は，実は，上に記したような，ピラミンクスの初期状態における置換をそのまま数学的に表現している訳ではない．というのは，もし，例えば $B3$ が実行された後では，頂点 A の位置には，初期状態で頂点 D に有った部分テトラが来ており，その状態で回転 $A1$ を実行すると，得られる置換は $(10, 1, 19)$ ではなく，$D1$ の逆向き回転に相当する $(23, 32, 18)$ になる．一般に，テトラの現在の状態が置換 σ で表されているとき，回転 $A1$ は，巡回置換 $(10, 1, 19)$ そのものではなく，$(\sigma(10), \sigma(1), \sigma(19))$ となる．一般に，テトラの現在の状態が置換 σ の作用の結果として表されているということは，そのときの面の配列の内容が，

$$(\sigma^{-1}(1), \sigma^{-1}(2), \ldots, \sigma^{-1}(36))$$

であるということである．これに，初期状態のテトラに対しては置換 τ で表されるような回転を施した結果，面の配列の現在の第 i 要素の内容の行き先が第 $\tau(i)$ 要素の内容となるので，数の置換としては $j = \sigma^{-1}(i) \mapsto \tau(i) = \tau(\sigma(j))$ となる．すなわち，回転の結果, 配列の第 j 要素の内容は，$\tau(\sigma(j))$ となる．従って，ピラミンクスの現在の状態を順列として読み取って, それを結果とするような置換を考えた場合は, 逆置換をとらないと実情に合わなくなる. これは, あみだくじなど, 置換を作用させるゲームの場合に共通の現象である. そこで, 現在のピラミンクスの面の状態が順列, すなわち数学の意味での置換として, 基本操作の積

$$\sigma_1 * \sigma_2 * \cdots * \sigma_k \tag{1}$$

で表されていたとき，実際にこの結果を得るには，この逆置換

$$\sigma_k^{-1} * \cdots * \sigma_2^{-1} * \sigma_1^{-1}$$

を (左から順に) 初期状態のピラミンクスに施すことになる. 従って, これを初期状態に戻すには, 左から順に (1) を施せばよいことになる. あるいは, 言い替えると, 順列として現在状態が置換 σ で表されているとき, これに置換 τ で表される基本操作を施すと, 結果は置換 $\tau^{-1}\sigma$ が表す順列になる. なお, ピラミンクスの置換群としての構造を調べる際には，逆元を取るかどうかは無関係である．

2 GAP

以下，ピラミンクスの置換群を GAP を用いて解析することを試みる．GAP は Groups, Algorithms and Programming の略語で，Scotland の University of St. Andrews の School of Math. and Comp. Sci. により開発・配布されており，`http://www.gap-system.org/` からソースまたはバイナリがダウンロードできる．GPL に従うフリーソフトであり．有限群論を中心とする主な代数計算が行える．現在バージョン 4, リビジョン 4, パッチレベル 7 まで進んでいる．ソースパッケージを入手し，解凍し configure, make を実行すれば，大抵の環境で動かすことができる．インストール後は，コマンドラインで `gap.sh` と入力するか，xgap のアイコンをマウスでクリックするかして立ち上げる．

以下に，簡単な GAP の指令の例を記す．プロンプト `gap>` の次が自分で打ち込んだコマンドで，次の行が GAP の応答である．

```
gap> (1,2)*(2,3);                    置換の積
(1,3,2)                              これから GAP は積を左から順に実行することが分かる．
gap> g:=Group((1,2,3,4),(1,2));      この二つで生成される置換群の定義
Group([ (1,2,3,4), (1,2) ])
gap> Order(g);
24                                   これから g = S_4 が分かる
```

```
gap> Elements(g);
[ (),(3,4),(2,3),(2,3,4),(2,4,3),(2,4),(1,2),(1,2)(3,4),(1,2,3),
  (1,2,3,4),(1,2,4,3),(1,2,4),(1,3,2),(1,3,4,2),(1,3),(1,3,4),
  (1,3)(2,4),(1,3,2,4),(1,4,3,2),(1,4,2),(1,4,3),(1,4),(1,4,2,3),
  (1,4)(2,3) ]
gap> h:=Group((1,2,3),(2,3,4));
12
gap> IsNormal(g,h);                    正規部分群か？
true
gap> k:=ConjugateGroup(h,(2,3));       共役 (2,3)h(2,3)^-1
Group([ (1,3,2), (2,4,3) ])
gap> h=k;                              h = k かどうか？
true
quit;
```

最後の quit **指令は，**GAP **をコマンドラインで起動したときの終了指令である．指令の最後には必ずセミコロンが必要なことに注意せよ．これは，**C **言語を始め，**Risa/Asir **など多くの数式処理ソフトと共通である．また，代入には** := **を使わなければならない．単に** = **だと** GAP **はこの条件判断の結果を返す．**

3 GAP によるピラミンクスの解析

最初に記したピラミンクスの基本置換データを GAP **に代入し, 得られた出力を下に示す**.

```
gap> A1:=(10,1,19);
(1,19,10)
gap> A2:=(11,2,20)*(12,3,21)*(13,4,22);
(2,20,11)(3,21,12)(4,22,13)
gap> A3:=(14,5,23)*(15,6,24)*(16,7,25)*(17,8,26)*(18,9,27)*(32,36,28)*\
(33,35,30)*(34,31,29);
(5,23,14)(6,24,15)(7,25,16)(8,26,17)(9,27,18)(28,32,36)(29,34,31)(30,33,35)
gap> B1:=(27,5,28);
(5,28,27)
gap> B2:=(25,2,31)*(26,6,30)*(22,7,29);
(2,31,25)(6,30,26)(7,29,22)
gap> B3:=(23,1,36)*(24,3,35)*(20,4,34)*(21,8,33)*(19,9,32)*(10,14,18)*\
(13,11,16)*(12,15,17);
(1,36,23)(3,35,24)(4,34,20)(8,33,21)(9,32,19)(10,14,18)(11,16,13)(12,15,17)
gap> C1:=(9,14,36);
(9,14,36)
gap> C2:=(7,11,34)*(8,15,35)*(4,16,31);
(4,16,31)(7,11,34)(8,15,35)
gap> C3:=(5,10,32)*(6,12,33)*(2,13,29)*(3,17,30)*(1,18,28)*(19,23,27)*\
(21,24,26)*(20,25,22);
(1,18,28)(2,13,29)(3,17,30)(5,10,32)(6,12,33)(19,23,27)(20,25,22)(21,24,26)
gap> D1:=(23,32,18);
(18,23,32)
gap> D2:=(20,29,16)*(24,33,17)*(25,34,13);
(13,25,34)(16,20,29)(17,24,33)
gap> D3:=(19,28,14)*(21,30,15)*(22,31,11)*(26,35,12)*(27,36,10)*(1,5,9)*\
(3,6,8)*(2,7,4);
(1,5,9)(2,7,4)(3,6,8)(10,27,36)(11,22,31)(12,26,35)(14,19,28)(15,21,30)
gap> g:=Group(A1,A2,A3,B1,B2,B3,C1,C2,C3,D1,D2,D3);
<permutation group with 12 generators>
gap> Order(g);
906992640
```

最後の結果から, **ピラミンクスの置換群が生成元１２個で位数** 9 **億** 699 **万** 2640 **を持つことが分かる**. **この計算を** 2GHz **の** AMD64 CPU **上では，**GAP **はほぼ一瞬でやってくれる**.

ピラミンクスの解法を発見するには, 任意に与えられたこの置換群の元を, $A1 \sim D3$ の積に分解しなければならない. この分解は, これらの基本的な置換が, 置換群の生成元としてはあまり基本的でないということから, 人間が目の子で探すのはそう簡単ではない (だからゲームになっている！)

もう少し GAP で群の様子を調べてみると,

```
gap> h1:=Group(A1,B1,C1,D1);
<permutation group with 4 generators>
gap> Order(h1);
81
gap> h2:=Group(A2,B2,C2,D2);
<permutation group with 4 generators>
gap> Order(h2);
933120
gap> h3:=Group(A3,B3,C3,D3);
<permutation group with 4 generators>
gap> Order(h3);
3732480
gap> IsNormal(g,h1);
true
gap> IsNormal(g,h2);
true
gap> IsNormal(g,h3);
false
gap> s:=ClosureGroup(h1,h2);
<permutation group with 8 generators>
gap> Order(s);
75582720
gap> 81*933120;
75582720
gap> t:=FactorGroup(g,s);
Group([ f1, f2, f3 ])
gap> Order(t);
12
gap> StructureDescription(t);
``A4''
```

これより, 1 型と 2 型の回転はそれぞれ正規部分群 $h1$, $h2$ を作るが, 3 型の回転は正規部分群ではないことが分かる. また, 最初の二つはピラミンクスを動かしてみれば当然予想されるように, この置換群全体の中で直積 $s = h1 \times h2$ を成している. この部分群による剰余類群は位数がたった 12 で，しかもその構造は４次の交代群であることが分かった．これは，４頂点の (向きを変えない) 置換の群と一致する．しかも，明らかに $h3$ の元により各面の三角形は同じ面に留まる．

以上の考察で，一つの解法の方針として, まず３型の回転 $A3, B3, C3, D3$ を適当に用いて, 頂点のテトラを正しい位置に直し，次に１型の回転 $A1, B1, C1, D1$ を適当に用いて, 頂点のテトラの向きを合わせると，残るは $h2$ の元による置換だけとなる．ここまでは，特に群論的考察をしなくても，目の子で実行できる．よってこのゲームの核心は部分群 $h2$ の解析である．

ちなみに，群構造の問い合わせを直接部分群に行ってみると，

```
gap> StructureDescription(h1);
"C3 x C3 x C3 x C3"
gap> StructureDescription(h2);
"C3 x C3 x C3 x C3 x ((C2 x C2 x C2 x C2 x C2) : A6)"
```

ここで, x は直積を，: は半直積を表している．最後の結果は AMD Athlon64 2GHz，手記憶 1GB の機械で２分くらいの計算時間がかかったので，$h3$ や g は１日では終わらないだろうと想像される．実際にやってみると，

```
gap> StructureDescription(g);
Error, too many (531441) complements called from
SubgroupsSolvableGroup( r, rec(
    actions := auts,
    funcnorm := r,
    consider := ExactSizeConsiderFunction( Index( j, M ) ),
    normal := true ) ) called from
NormalSubgroups( G ) called from
DirectFactorsOfGroup( G ) called from
<function>( <arguments> ) called from read-eval-loop
Entering break read-eval-print loop ...
you can 'quit;' to quit to outer loop, or
you can 'return;' to continue
brk>
```

と３０分ほどで表示され，スタックオーバーフローのような症状となった．GAP のマニュアルには，大きな群の計算を守備良く行うためのテクニックがいろいろ書かれているので，工夫すればこの計算も遂行できるかもしれない．しかし，以上の考察から，GAP に頼らなくても，ピラミンクスの置換群は

$$(h_1 \times h_2) \ltimes A_4 = [C_3 \times C_3 \times C_3 \times C_3 \times C_3 \times C_3 \times C_3 \times C_3 \times \{(C_2 \times C_2 \times C_2 \times C_2 \times C_2) \ltimes A_6\}] \ltimes A_4$$

という構造をしていることが推察できる．(ここでは，半直積等に普通の群論での表記法を用いた．) 位数はもちろん $3^8 \times 2^5 \times 360 \times 12 = 906992640$ で，最初の計算結果と合っている．

剰余類群 t の生成元 $f1, f2, f3$ 等は GAP に問い合わせても直接は知ることができないので，剰余類を尋ねてみた．GAP には，離散数学で普通に学ぶ左剰余類を与える函数が存在しないので，右剰余類を尋ねると

```
gap> RightCosets(g,s);
[ RightCoset(Group( [ (18,23,32), ( 9,14,36), ( 5,28,27), ( 1,19,10),
      ( 2,20,11)( 3,21,12)( 4,22,13), ( 2,31,25)( 6,30,26)( 7,29,22),
      ( 4,16,31)( 7,11,34)( 8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
),()),
  RightCoset(Group( [ (18,23,32), ( 9,14,36), ( 5,28,27), ( 1,19,10),
      ( 2,20,11)( 3,21,12)( 4,22,13), ( 2,31,25)( 6,30,26)( 7,29,22),
      ( 4,16,31)( 7,11,34)( 8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
),
    ( 5, 9,18)( 6, 8,17)(14,23,28)(15,24,30)(26,35,33)(27,36,32)),
  RightCoset(Group( [ (18,23,32), ( 9,14,36), ( 5,28,27), ( 1,19,10),
      ( 2,20,11)( 3,21,12)( 4,22,13), ( 2,31,25)( 6,30,26)( 7,29,22),
      ( 4,16,31)( 7,11,34)( 8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
),
    ( 5,18, 9)( 6,17, 8)(14,28,23)(15,30,24)(26,33,35)(27,32,36)),
  RightCoset(Group( [ (18,23,32), ( 9,14,36), ( 5,28,27), ( 1,19,10),
      ( 2,20,11)( 3,21,12)( 4,22,13), ( 2,31,25)( 6,30,26)( 7,29,22),
      ( 4,16,31)( 7,11,34)( 8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
),
    ( 1,27,23)( 3, 6,33)( 4,20,29)
( 5,32,19)(10,28,18)(11,13,25)(12,26,24)
    (17,21,30)),
  RightCoset(Group( [ (18,23,32), ( 9,14,36), ( 5,28,27), ( 1,19,10),
      ( 2,20,11)( 3,21,12)( 4,22,13), ( 2,31,25)( 6,30,26)( 7,29,22),
      ( 4,16,31)( 7,11,34)( 8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
),
    ( 1,27,36,19, 5, 9,10,28,14)( 3, 6, 8,21,30,15,12,26,35)( 4,20,29)
    (11,13,25)(17,33,24)(18,32,23)),
  RightCoset(Group( [ (18,23,32), ( 9,14,36), ( 5,28,27), ( 1,19,10),
      ( 2,20,11)( 3,21,12)( 4,22,13), ( 2,31,25)( 6,30,26)( 7,29,22),
```

```
      ( 4,16,31)( 7,11,34)( 8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
),
    ( 1,27,19, 5,10,28)( 3, 6,21,30,12,26)( 4,20,29)( 8,33,35,24,15,17)
    ( 9,32,36,23,14,18)(11,13,25)),
  RightCoset(Group( [ (18,23,32), ( 9,14,36), ( 5,28,27), ( 1,19,10),
      ( 2,20,11)( 3,21,12)( 4,22,13), ( 2,31,25)( 6,30,26)( 7,29,22),
      ( 4,16,31)( 7,11,34)( 8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
),
    ( 1,36,28)( 2,31,13)( 3,35,26)( 5,10,14)( 6,21, 8)( 7,20,22)
( 9,27,19)
    (12,15,30)),
  RightCoset(Group( [ (18,23,32), ( 9,14,36), ( 5,28,27), ( 1,19,10),
      ( 2,20,11)( 3,21,12)( 4,22,13), ( 2,31,25)( 6,30,26)( 7,29,22),
      ( 4,16,31)( 7,11,34)( 8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
),
    ( 1,36,32,19, 9,18,10,14,23)( 2,31,13)( 3,35,33)( 5,27,28)( 7,20,22)
    ( 8,17,21)(12,15,24)),
  RightCoset(Group( [ (18,23,32), ( 9,14,36), ( 5,28,27), ( 1,19,10),
      ( 2,20,11)( 3,21,12)( 4,22,13), ( 2,31,25)( 6,30,26)( 7,29,22),
      ( 4,16,31)( 7,11,34)( 8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
),
    ( 1,36,19, 9,10,14)( 2,31,13)( 3,35)( 5,18,27,32,28,23)( 6,17)
( 7,20,22)
    ( 8,21)(12,15)(24,30)(26,33)),
  RightCoset(Group( [ (18,23,32), ( 9,14,36), ( 5,28,27), ( 1,19,10),
      ( 2,20,11)( 3,21,12)( 4,22,13), ( 2,31,25)( 6,30,26)( 7,29,22),
      ( 4,16,31)( 7,11,34)( 8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
),
    ( 1,32,14)( 2, 4,16)( 3,17,15)( 8,12,33)
( 9,10,23)(11,34,22)(18,36,19)
    (21,24,35)),
  RightCoset(Group( [ (18,23,32), ( 9,14,36), ( 5,28,27), ( 1,19,10),
      ( 2,20,11)( 3,21,12)( 4,22,13), ( 2,31,25)( 6,30,26)( 7,29,22),
      ( 4,16,31)( 7,11,34)( 8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
),
    ( 1,32,27,19,18, 5,10,23,28)( 2, 4,16)( 3,17, 6,12,33,26,21,24,30)
    ( 8,15,35)( 9,36,14)(11,34,22)),
  RightCoset(Group( [ (18,23,32), ( 9,14,36), ( 5,28,27), ( 1,19,10),
      ( 2,20,11)( 3,21,12)( 4,22,13), ( 2,31,25)( 6,30,26)( 7,29,22),
      ( 4,16,31)( 7,11,34)( 8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
),
    ( 1,32,19,18,10,23)( 2, 4,16)( 3,17,12,33,21,24)( 5,36,27,14,28, 9)
    ( 6,15,30,35,26, 8)(11,34,22)) ]
```

となり，１２個の右剰余類の代表元

```
();
x1:=(5,9,18)(6,8,17)(14,23,28)(15,24,30)(26,35,33)(27,36,32);
x2:=(5,18,9)(6,17,8)(14,28,23)(15,30,24)(26,33,35)(27,32,36);
x3:=(1,27,23)(3,6,33)(4,20,29)(5,32,19)(10,28,18)(11,13,25)(12,26,24)(17,21,30);
x4:=(1,27,36,19,5,9,10,28,14)(3,6,8,21,30,15,12,26,35)(4,20,29)(11,13,25)(17,33,24)(18,32,23);
x5:=(1,27,19,5,10,28)(3,6,21,30,12,26)(4,20,29)(8,33,35,24,15,17)(9,32,36,23,14,18)(11,13,25);
x6:=(1,36,28)(2,31,13)(3,35,26)(5,10,14)(6,21,8)(7,20,22)(9,27,19)(12,15,30);
x7:=(1,36,32,19,9,18,10,14,23)(2,31,13)(3,35,33)(5,27,28)(7,20,22)(8,17,21)(12,15,24);
x8:=(1,36,19,9,10,14)(2,31,13)(3,35)(5,18,27,32,28,23)(6,17)(7,20,22)(8,21)(12,15)(24,30)(26,33);
x9:=(1,32,14)(2,4,16)(3,17,15)(8,12,33)(9,10,23)(11,34,22)(18,36,19)(21,24,35);
xa:=(1,32,27,19,18,5,10,23,28)(2,4,16)(3,17,6,12,33,26,21,24,30)(8,15,35)(9,36,14)(11,34,22);
xb:=(1,32,19,18,10,23)(2,4,16)(3,17,12,33,21,24)(5,36,27,14,28,9)(6,15,30,35,26,8)(11,34,22);
```

が得られる．これらは見にくいし，当然あるはずの， $\mathtt{x3}^{-1}$ **が属する剰余類がすぐには見付からないなどの不便さを持つ．そこで，代表元を標準的なもの** (**なるべく小さな数で記述される代表元**) **に取り直すと**

```
gap> y1:=CanonicalRightCosetElement(s,x1);
(5,9,18)(6,8,17)(14,23,28)(15,24,30)(26,35,33)(27,36,32)
gap> y2:=CanonicalRightCosetElement(s,x2);
(5,18,9)(6,17,8)(14,28,23)(15,30,24)(26,33,35)(27,32,36)
gap> y3:=CanonicalRightCosetElement(s,x3);
(1,5,18)(3,6,17)(10,27,32)(12,26,33)(19,28,23)(21,30,24)
gap> y4:=CanonicalRightCosetElement(s,x4);
(1,5,9)(3,6,8)(10,27,36)(12,26,35)(14,19,28)(15,21,30)
gap> y5:=CanonicalRightCosetElement(s,x5);
(1,5)(3,6)(8,17)(9,18)(10,27)(12,26)(14,23)(15,24)(19,28)(21,30)(32,36)(33,35)
gap> y6:=CanonicalRightCosetElement(s,x6);
(1,9,5)(3,8,6)(10,36,27)(12,35,26)(14,28,19)(15,30,21)
gap> y7:=CanonicalRightCosetElement(s,x7);
(1,9,18)(3,8,17)(10,36,32)(12,35,33)(14,23,19)(15,24,21)
gap> y8:=CanonicalRightCosetElement(s,x8);
(1,9)(3,8)(5,18)(6,17)(10,36)(12,35)(14,19)(15,21)(23,28)(24,30)(26,33)(27,32)
gap> y9:=CanonicalRightCosetElement(s,x9);
(1,18,9)(3,17,8)(10,32,36)(12,33,35)(14,19,23)(15,21,24)
gap> ya:=CanonicalRightCosetElement(s,xa);
(1,18,5)(3,17,6)(10,32,27)(12,33,26)(19,23,28)(21,24,30)
gap> yb:=CanonicalRightCosetElement(s,xb);
(1,18)(3,17)(5,9)(6,8)(10,32)(12,33)(14,28)(15,30)(19,23)(21,24)(26,35)(27,36)
```

となって，かなり見通しがよい．特に，取り替えられたものの中から，

```
y1:=(5,9,18)(6,8,17)(14,23,28)(15,24,30)(26,35,33)(27,36,32);
y5:=(1,5)(3,6)(8,17)(9,18)(10,27)(12,26)(14,23)(15,24)(19,28)(21,30)(32,36)(33,35);
y8:=(1,9)(3,8)(5,18)(6,17)(10,36)(12,35)(14,19)(15,21)(23,28)(24,30)(26,33)(27,32);
```

を取ると，これらが g/s **を生成することが容易に確かめられる**:

$$
\begin{aligned}
&[y1]=[x1],\ [y1][y1]=[x2],\ [y1][y5]=[x3],\ [y1]^2[y5]=[x4],\ [y5]=[x5],\ [y1][y8]=[x6],\\
&[y1]^2[y8]=[x7],\ [y8]=[x8],\ [y1][y5][y8]=[x9],\ [y1]^2[y5][y8]=[xa],\ [y5][y8]=[xb],\\
&[y1]^3=[1],\ [y5]^2=1,\ [y8]^2=1
\end{aligned}
$$

ちなみに, **もともとの** $h3$ **の生成元は**

```
gap> CanonicalRightCosetElement(s,A3);
(5,18,9)(6,17,8)(14,28,23)(15,30,24)(26,33,35)(27,32,36)
gap> CanonicalRightCosetElement(s,B3);
(1,9,18)(3,8,17)(10,36,32)(12,35,33)(14,23,19)(15,24,21)
gap> CanonicalRightCosetElement(s,C3);
(1,18,5)(3,17,6)(10,32,27)(12,33,26)(19,23,28)(21,24,30)
gap> CanonicalRightCosetElement(s,D3);
(1,5,9)(3,6,8)(10,27,36)(12,26,35)(14,19,28)(15,21,30)
```

より, **上の記号でそれぞれ** $[A3]=[y2]$, $[B3]=[y7]$, $[C3]=[ya]$, $[D3]=[y4]$ **となっている**. **従って，逆に，**

$$
\begin{aligned}
&[y1]=[y2]^{-1}=[A3]^{-1}, && [y2]=[A3],\\
&[y3]=[y1]*[y5]=[A3]*[D3], && [y4]=[D3],\\
&[y5]=[y1]*[y4]=[A3]^{-1}*[D3], && [y6]=[y1]*[y8]=[A3]*[B3],\\
&[y7]=[B3], && [y8]=[y1]*[y7]=[A3]^{-1}*[B3],\\
&[y9]=[y1]*[y5]*[y8]=[A3]*[D3]*[A3]^{-1}*[B3], && [ya]=[C3],\\
&[yb]=[y5]*[y8]=[A3]^{-1}*[D3]*[A3]^{-1}*[B3]
\end{aligned}
\tag{2}
$$

なお，以上の考察から $h1$ は全置換群の中で全く独立なので，群のセンターに含まれるように錯覚されるが，ここでは $A1$ 等の置換を，元の頂点 A での回転と定義しているので，これらはそのままでは $B3$ 等と可換でない．実際のセンターは

```
gap> c:=Center(g);
<permutation group of size 18 with 3 generators>
gap> StructureDescription(c);
"C6 x C3"
```

という構造をしている．この元は，

```
gap> Elements(c);
[(),
 (3,12,21)(6,26,30)(8,35,15)(17,33,24),
 (3,21,12)(6,30,26)(8,15,35)(17,24,33),
 (2,22)(4,11)(7,31)(13,20)(16,34)(25,29),
 (2,22)(3,12,21)(4,11)(6,26,30)(7,31)(8,35,15)(13,20)(16,34)(17,33,24)(25,29),
 (2,22)(3,21,12)(4,11)(6,30,26)(7,31)(8,15,35)(13,20)(16,34)(17,24,33)(25,29),
 (1,10,19)(5,27,28)(9,36,14)(18,32,23),
 (1,10,19)(3,12,21)(5,27,28)(6,26,30)(8,35,15)(9,36,14)(17,33,24)(18,32,23),
 (1,10,19)(3,21,12)(5,27,28)(6,30,26)(8,15,35)(9,36,14)(17,24,33)(18,32,23),
 (1,10,19)(2,22)(4,11)(5,27,28)(7,31)(9,36,14)(13,20)(16,34)(18,32,23)(25,29),
 (1,10,19)(2,22)(3,12,21)(4,11)(5,27,28)(6,26,30)(7,31)(8,35,15)(9,36,14)
     (13,20)(16,34)(17,33,24)(18,32,23)(25,29),
 (1,10,19)(2,22)(3,21,12)(4,11)(5,27,28)(6,30,26)(7,31)(8,15,35)(9,36,14)
     (13,20)(16,34)(17,24,33)(18,32,23)(25,29),
 (1,19,10)(5,28,27)(9,14,36)(18,23,32),
 (1,19,10)(3,12,21)(5,28,27)(6,26,30)(8,35,15)(9,14,36)(17,33,24)(18,23,32),
 (1,19,10)(3,21,12)(5,28,27)(6,30,26)(8,15,35)(9,14,36)(17,24,33)(18,23,32),
 (1,19,10)(2,22)(4,11)(5,28,27)(7,31)(9,14,36)(13,20)(16,34)(18,23,32)(25,29),
 (1,19,10)(2,22)(3,12,21)(4,11)(5,28,27)(6,26,30)(7,31)(8,35,15)(9,14,36)
     (13,20)(16,34)(17,33,24)(18,23,32)(25,29),
 (1,19,10)(2,22)(3,21,12)(4,11)(5,28,27)(6,30,26)(7,31)(8,15,35)(9,14,36)
     (13,20)(16,34)(17,24,33)(18,23,32)(25,29) ]
```

となる．(見やすくするように，GAP の出力を少し整形してある．)

4 GAP が示唆するピラミンクスの解法

これからの課題は，$h2$ の元の中でなるべく単純な置換を発見することである．今まで試みたことを記す．

```
gap> m1:=A2*B2*A2^-1*B2^-1;
(2,25,11)(4,22,29)
gap> m2:=A2*C2*A2^-1*C2^-1;
(4,31,13)(7,20,11)
gap> k:=m1*m2;
(2,25,7,20,11)(4,22,29,31,13)          ← 位数５の元
gap> Order(k);
5
gap> m3:=B2*C2^-1*B2^-1*C2;
(2,31,4)(7,11,22)
gap> m4:=A2^-1*C2*A2*C2^-1;
(2,11,7)(4,31,22)
gap> m3*m4;
(2,22)(4,11)                           ← 位数２の元
```

```
gap> m4*m3;
(2,22)(7,31)
```

位数が５の元が得られたのは興味深い．また，位数２の元の比較的 (偶置換に限られているので，実は最も) 単純な元も見出された (図 3 は $(2,22)(4,11)$ の結果)．(実はこのような例を得るまでにかなりの実験をしている．(^^; 群論の専門家なら，もっと理詰めで発見できるかもしれない.) これらを組み合わせると，実際の解法が得られる．これは最後の節で検討する.

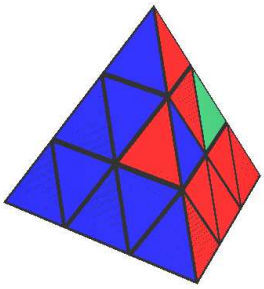

図 3

GAP で部分群 $h2$ を解析すると，

```
gap> GeneratorsOfGroup(h2);
[ (2,20,11)(3,21,12)(4,22,13), (2,31,25)(6,30,26)(7,29,22),
  (4,16,31)(7,11,34)(8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
```

となり，生成元として $A2, B2, C2, D2$ がそのまま使えることが分かるので，試しに GAP が $h2$ の勝手な元をこれらの生成元で表示できるか調べてみた．勝手な元と生成元を指定して，語の問題を解くという函数は存在しないが，マニュアルには次のような計算法が説明されている：すなわち，まず，$h2$ の生成元と同じ個数の自由元より成る自由群を想定し，そこから $h2$ への全射準同型を定義する．$h2$ の勝手な元の，この準同型による逆像を与える函数は存在するので，それを使うと，自由群の生成元の語として，求める表現と同等なものが得られる，という方針である．

```
gap> x:=A2*B2*C2*A2*D2*B2*C2;
(2,34,29)(3,12,21)(6,26,30)(7,31)(8,35,15)(13,20)(16,25,22)(17,24,33)
gap> hom:=EpimorphismFromFreeGroup(h2:names:=["a","b","c","d"]);
[ a, b, c, d ] -> [ (2,20,11)(3,21,12)(4,22,13), (2,31,25)(6,30,26)(7,29,22),
  (4,16,31)(7,11,34)(8,15,35), (13,25,34)(16,20,29)(17,24,33) ]
gap> PreImagesRepresentative(hom,x);
b*c*d*c^-1*d^-1*b^-2*a^-1*c*a^-1*b^-1*a^-1*b^-1*a^-1*d^-1*a^2*b^-1*a^-2*c^-1*a*d^-1
*c^-1*a^-1
gap> y:=B2*C2*D2*D2*D2*D2*D2*B2*A2*A2*C2*A2*A2*B2*B2*A2*A2*B2*B2*A2*A2*D21*a^2*b^-1
*a^-2*C2*C2*A2*D2*D2*C2*C2*A2*A2;
(2,34,29)(3,12,21)(6,26,30)(7,31)(8,35,15)(13,20)(16,25,22)(17,24,33)
gap> x*y^-1;
()
```

上述のように，実際やってみると，最初に与えた勝手な元の語よりは，ずっと複雑な語が返された．(最後の行で，両者が確かに同値であることは確認できている．) 群 $h2$ においては，明らかに `a^-2=a` なのに，自由群に持ち上げたところでは，そのように置き換えられていない．従って，GAP にこのような計算をやらせれば，必ず解法を示してくれることにはなるが，一般にはかなり無駄な手数を要求されることが想像できる．

実際の解法例として，シードをデフォールトにして, `random()` を 30 回呼び，$A1 \sim D3$ をランダムに適用して得られた S_{36} の元 (我々の OpenGL ピラミンクス描画ソフトがターミナルウインドウに出力したもの)

```
[5,16,6,2,23,33,13,35,14,27,22,26,11,36,8,7,21,19,28,4,30,34,10,12,29,
24,18,32,25,17,20,1,3,31,15,9]
```

に相当するパターン (図 4) を上記の方針で解いてみる．

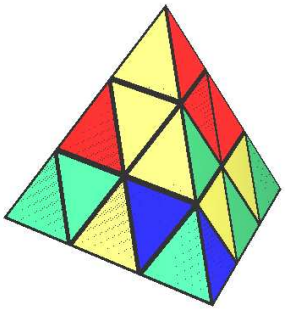

図 4

GAP は，S_{36} の元をこのような形で入力する関数が無いようなので，まず，この置換を，離散数学のレポートで作った自作の簡単なＣプログラムにより，互いに独立な巡回置換の積

```
(1,5,23,10,27,18,19,28,32)(2,16,7,13,11,22,34,31,20,4)(3,6,33)(8,35,15)
(9,14,36)(12,26,24)(17,21,30)(25,29)
```

に分解する．これを GAP に食べさせて，まずその剰余類群 $g/(h1 \times h2)$ における表現を見ると，

```
gap> x:=(1,5,23,10,27,18,19,28,32)(2,16,7,13,11,22,34,31,20,4)(3,6,33)(8,35,15)
(9,14,36)(12,26,24)(17,21,30)(25,29);
(1,5,23,10,27,18,19,28,32)(2,16,7,13,11,22,34,31,20,4)(3,6,33)(8,35,15)(9,14,
36)(12,26,24)(17,21,30)(25,29)
gap> CanonicalRightCosetElement(s,x);
(1,5,18)(3,6,17)(10,27,32)(12,26,33)(19,28,23)(21,30,24)
```

となる．これは先の表 (2) で

$$[y3] = [A3] * [D3]$$

に相当することが分かる．この置換の種類は高々位数 12 の範囲での選択なので，計算機を使わなくても目の子で同じものを見出せるが，ここでは計算機で自動的に解くための手順を記している．さて，$h1 \times h2$ のある元 y が有り，$x = y * [A3] * [D3]$ となっている訳だが，この $[A3] * [D3]$ を剥ぎ取るために，τ という置換を適用すると，結果は最初の節で注意したように，$\tau^{-1} * x = \tau^{-1} * y * [A3] * [D3]$ が与える順列となる．よって，τ をどう選んでも, 因子 $[A3] * [D3]$ を直接剥ぎ取ることはできないが，この結果を正規部分群 $h1 \times h2$ の中に落とすだけなら，$\tau = [A3] * [D3]$ でよいことが分かる．よってこの置換 $[A3] * [D3]$ の作用の結果は合成置換 $[D3]^{-1} * [A3]^{-1} * x$ が表す順列となり，

```
[19,2,21,7,5,6,16,8,36,1,31,3,11,9,15,29,24,18,10,4,12,22,23,33,13,26,27,
28,20,30,34,32,17,25,35,14]
=(1,19,10)(3,21,12)(4,7,16,29,20)(9,36,14)(11,31,34,25,13)(17,24,33)
```

となる．

次に頂点部分に対応する配列成分, 例えば FACE[1], FACE[5], FACE[9], FACE[18] に，それぞれ $1, 5, 9, 18$ が入っているかどうかを見て，だめなら, それぞれ $A1, B1, C1, D1$ を使ってそろえる．これは目の子でやる方が間単なくらいである．この例では，$A1 * C1^{-1}$ で角が揃う．以上の結果は，上で因子 $(1, 10, 19)$ と $(9, 14, 36)$ を取り去った

```
[1 2 21 7 5 6 16 8 9 10 31 3 11 14 15 29 24 18 19 4 12 22 23 33 13 26 27 28 20 30 34 32 17 25 35
=(3,21,12)(4,7,16,29,20)(11,31,34,25,13)(17,24,33)
```

となることは明らかである．以上で実際のピラミンクスは

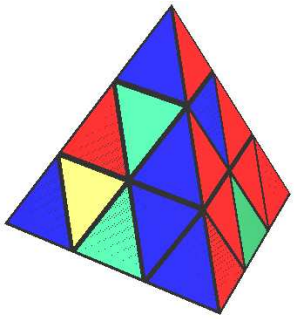

図 5

となっている．この置換に対して，いよいよ GAP **にこのゲームの核心である** $h2$ **における語の問題を解かせると，**

```
gap> z:=(3,21,12)(4,7,16,29,20)(11,31,34,25,13)(17,24,33)
(3,21,12)(4,7,16,29,20)(11,31,34,25,13)(17,24,33)
gap> PreImagesRepresentative(hom,z);
a*d^-1*c*b^-1*a*b*a^-2*d^-1*a*b^-1*c^-1*b*a*b*c*b^-1*c^-1*a^-1*b*c^-1*b^-1*c
*b^-1*a*b*a^-1
```

と，数秒で答が返る．よって，今までの説明により，これを読み替えた置換

$$A2 * D2^{-1} * C2 * B2^{-1} * A2 * B2 * A2 * D2^{-1} * A2 * B2^{-1} * C2^{-1} * B2 * A2 * B2 * C2 * B2^{-1} * C2^{-1}$$
$$* A2^{-1} * B2 * C2^{-1} * B2^{-1} * C2 * B2^{-1} * A2 * B2 * A2^{-1}$$

を頭から順に実行してゆけば，元に戻るはずである．ここで，明らかな関係式 $A2^{-2} = A2$ **等は適用して簡略化した．**

実際にやってみると，途中で間違えなければ，確かに元に戻る．

以下，群 $h2$ **の特殊な元について，**GAP **を用いて調べを続ける．見やすくするため，** $A2, B2, C2, D2$ **の代わりに本文中でも** a, b, c, d **を用いることにする．偶然で見付けた位数２の元について，**GAP **が与える解は**

```
gap> z:=(4,11)(25,29);
(4,11)(25,29)
gap> PreImagesRepresentative(hom,z);
a*b^-1*a^-1*b*c^-1*b*c*b^-1*a*c*b*c^-1*b^-1*a^-1*c^-1*b*c*d*c^-1*d^-1*b^-2*a^-1
*c*a^-1*b^-1*a^-1*b^-1
gap> z:=(4,11)(16,34);
(4,11)(16,34)
gap> PreImagesRepresentative(hom,z);
a*c*b*c^-1*b^-1*a^-1*b^-1*c^-1*a^-1*c*a*b
gap> z:=(16,34)(25,29);
(16,34)(25,29)
gap> PreImagesRepresentative(hom,z);
c*a*b*c*b^-1*c^-1*a^-1*c*b^-1*a^-1*c*a^-1*b^-1*a^-1*b^-1
gap> z:=(4,11)(7,31);
(4,11)(7,31)
gap>  PreImagesRepresentative(hom,z);
a*b^-1*a^-1*b*c^-1*b*c*b^-1
```

人間が見たら明らかに同等の置換でも，GAP **が与える解の長さが異なるのが面白い．これらを組み合わせると，生成元の間の関係式がいろいろ求まる．本当は，** $h2$ **の基本関係を何とか求めたいのだが，短い語で表されたものは容易には見付からない．自明な関係式** $a^3 = b^3 = c^3 = d^3 = 1$, $[a, b]^3 := (aba^{-1}b^{-1})^3 = 1$ **等以外で見**

付かった比較的短いものは

$$bacdc = acdaba^{-1}b^{-1}cb,$$

$$cbcad = dcd^{-1}a^{-1}dabca$$

程度である. これらが上述の基本関係式から導けるかどうか, まだ調べていないが, 交換子のすべての組合せを用いれば関係式が生成できそうな気がする.

5 ピラミンクスの実用的解法

最初にこれを書けという声もあるでしょうが，本文書はあくまで，群論の教育に資することを目的として書かれたものなので，遊びは後まわしで，この順にしました．以下の解法手順のうち，一部は既に書いたことと重複します．

(1) まず，$A3, B3, C3, D3$ を用いて，頂点のサブテトラを正しい位置に直す．これは置換群 $A4$ の範囲なので，目の子でも容易にできる．

(2) 次に，頂点のサブテトラを適当に回転して，正しい向きに直す．これは，更に容易．

(3) 次に，$A2, B2, C2, D2$ を適当に用いて，頂点のサブテトラに接する内部三角形の色を合わせる．これも (2) と同じくらい容易にできる．

(4) 最後に，次のような位数３の基本置換 (交換子) を適当に組み合わせて，辺のテトラを正しい位置にもってゆく．
$[B2, C2^{-1}] := B2 * C2^{-1} * B2^{-1} * C2$
この置換は, 同一面内の三つの辺を巡回置換するが, そのうち二つは面の裏返しを起こすことに注意せよ.

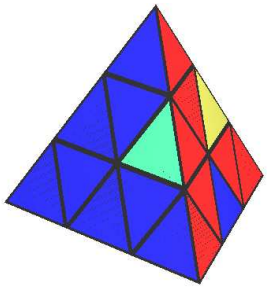

図 6

最後に述べた３次基本置換から，本質的には部分群 $h2$ を解くのに必要なすべての操作が合成できる．例えば，この３次基本置換を二つ合成すると，図 3 のような，二つの隣接辺テトラを裏返す操作が作れる：
$(2, 22)(4, 11) = [B2, C2^{-1}] * [A2^{-1}, C2] = B2 * C2^{-1} * B2^{-1} * C2 * A2^{-1} * C2 * A2 * C2^{-1}$
更に，それと３次基本操作を合成すると，下図 7 のような，一つの面内の三つの辺テトラを裏返さずに巡回置換することが可能となる：
$(7, 16, 25)(29, 31, 34) = [C2^{-1}, D2] * [B2, D2^{-1}] * [B2^{-1}, D2]$
$= C2^{-1} * D2 * C2 * D2^{-1} * B2 * D2^{-1} * B2^{-1} * D2 * B2^{-1} * C2 * B2 * C2^{-1}$

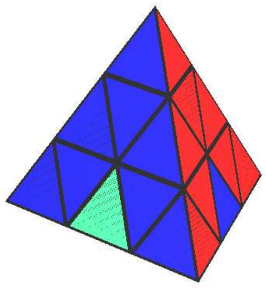

図 7

更に，３次基本置換を，予めずらしておいて適用する (共役元をとる) と，例えば，一つの中段の三つの辺テトラを向きを変えずに巡回置換することができる：

$(2,20,11)(4,22,13) = C2 * [A2, D2]] * C2^{-1}$

$= C2 * A2 * D2 * A2^{-1} * D2^{-1} * C2^{-1}$

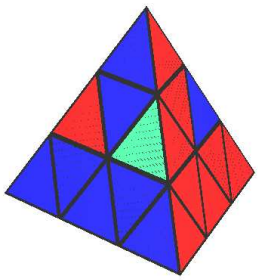

図 6

ここで挟まれた３次の置換は, 式表現上からは最初に述べたものより簡単でより基本的に見えるが, 離れた三つの辺テトラを巡回させるので, その作用を記憶するのは面倒であろう. そう思って記憶すべき基本置換には入れなかった. おそらく, これは, ピラミンクスの解法に絶対必要なものではないであろう.

同等の置換は他にも沢山有るが，最後に紹介した二つの置換は，頂点Ａの位置だけを決めれば，残りの頂点はどの位置に有っても操作が同等になるという意味で, 最も分かりやすいものであろう.

実用的解法としては，長い手順は記憶しづらいし, 途中で間違える可能性も大きくなるので, ルービックキューブの場合もそうであったように, 最初に述べた３次の基本置換を身体で覚え，あとはピラミンクス全体を回して, 適当に頂点を読み替えながら, この一つの置換だけを適用して変形してゆくのが最も分かりやすいであろう. ただし，この逆置換 $[C2, B2^{-1}] := C2 * B2^{-1} * C2^{-1} * B2$ の操作も簡単なので，それも一緒に覚えると，同じ置換を２度繰り返す $[B2, C2^{-1}]^2$ よりは効率的であろう．

なお，これら $h2$ の置換の実行に当たっては，苦労して第２段だけを回す必要はなく，その上の頂点テトラも一緒に回した方が操作しやすい．例えば，$A2$ の代わりに $A1A2$ を実行する等．上の合成諸置換では，いずれも，頂点テトラは操作の途中では動いても，最後は必然的に元の位置に戻る．