Pocket cube について

数学１班 守澤秀亮　藤谷佑　戸部尚征　藤原亮空
指導教員　瀬々将吏 先生

1. はじめに

ポケットキューブとは、俗に言う2×2×2のルービックキューブのことであり、ラリー・ニコルス氏が考案、開発したものである。ルービックキューブが20手以内に必ず解けることを知り、興味を持った我々は、そのことが本当なのかを確かめたいと思ったが、ルービックキューブのすべての配置の仕方が4000京通りを超え、膨大にあることから断念し、すべての配置の仕方がルービックキューブよりも少ないであろうポケットキューブを題材とした。

2.目的

ポケットキューブの最短手数の最大値「God’s number」を求め、数ある最短手数から法則を見つけ出す。「God's number」とは、ルービックキューブ最短手数の最大値のことで、「God’s number」= t とすると、ルービックキューブはどのような配置でも t 手以内で必ず解ける。以下のサイトにはポケットキューブの「God's number」は11手であると記載されていた。
※ IT用語の解説+判じ絵 https://www.daido-it.ac.jp/~oishi/TH5/rubic.html

3.実験方法

I. 総配置数(ポケットキューブの全配置の総和)を求める。
II. googlecolaboratoryでpythonを用いてポケットキューブをすべての色が揃った状態からn手(nは1から1ずつ大きくする)動かし、できた配置がlist内にあるかを調べ、ある場合はlistに保存せず、ない場合はlistに保存するというプログラム を作成する。list内の配置の総和が総配置数と等しくなったときのnの値を求める。そのnの値が「God's number」である。※なぜlist内の配置の総和が総配置数と等しくなったときのnの値が「God' s number」と言えるのかというと、n=kのときに初めてでてきたある配置から色が揃った状態に戻すためには、最短でk手必要となるから。(nの値は1から1ずつ大きくしていき、list内に同じ配置があった場合は、保存しないため、n=k-1まで動作を行い出てこずn=kのときに初めて出てきた配置の最短手数はk手となる)
III. list内の配置とその配置に至るまでの手順をたよりに法則性を見つける。

4.実験の実行

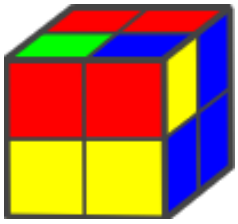
〈Ⅰ.総配置数を求める〉

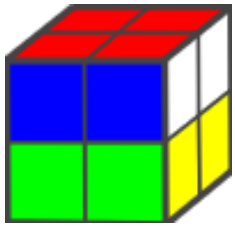
８個のブロックに分解し１つのブロックを固定し考える。
また、その色は考えない。
固定したブロック以外の７つのブロックの配置は７！通りある。
また、ひとつのブロックには色が３色あるためブロックの向きは３通り、
これを７個分考えるため3^7　通りとなる。よって計算式は　　$3^7 \times 7!$
となるがこのままでは普通にポケットキューブを動かしただけでは起こりえない、パリティと呼ばれる状態も含まれてしまう。
よってパリティを除くためにパリティが起こる確率を考えていく。

〈パリティの証明〉

先行研究ではポケットキューブでパリティが起こる確率は2/3とされていたが、その証明が記載されていなかったためその証明をしていく。

完成状態に対して向きに相違がある状態を「向きズレ」、
位置に相違がある状態を「位置ズレ」と呼ぶことにする。


向きズレ

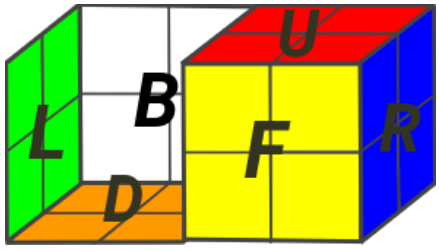

位置ズレ

下の図のように
時計回りに120°ズレているピースの個数＝ c （clock cyclic）
反時計回りに120°ズレているピースの個数＝uc（unticlock cyclic）
とする。


反時計まわり


時計まわり

ポケットキューブを　正面をF面、背面をB面、左面をL面
右面をR面、上面をU面、下面をD面とすると



D面が揃っているとして、U面のピースの位置ズレと向きズレの
関係性を考える。　すると、下表のような結果になった。

位置ズレ（個）	向きズレ（個）				
0	c =0 uc=0	c=1 uc=1	c=2 uc=2	c=3 uc=0	c=0 uc=3
2	c =0 uc=0	c=1 uc=1	c=2 uc=2	c=3 uc=0	c=0 uc=3
3	c =0 uc=0	c=1 uc=1	c=2 uc=2	c=3 uc=0	c=0 uc=3

表は、向きズレの起きているピースの個数を表している。また、位置ズレが0,2,3個のいずれのときでも向きズレの組み合わせはすべて起こっていた。（位置ズレは1個だけで起こらないため、位置ズレの個数は、0,2,3のいずれかである。）表より、ピースの位置ズレに関係なく、cとucは３を法として合同であることに気づいた。このことから、パリティにならない条件はこのことと等しいのではと仮定して次のようなことを考えた。

パリティが起こるのはcとucがmod３で合同

かつ**D面が揃っていない場合も同様に成り立つのでは？**

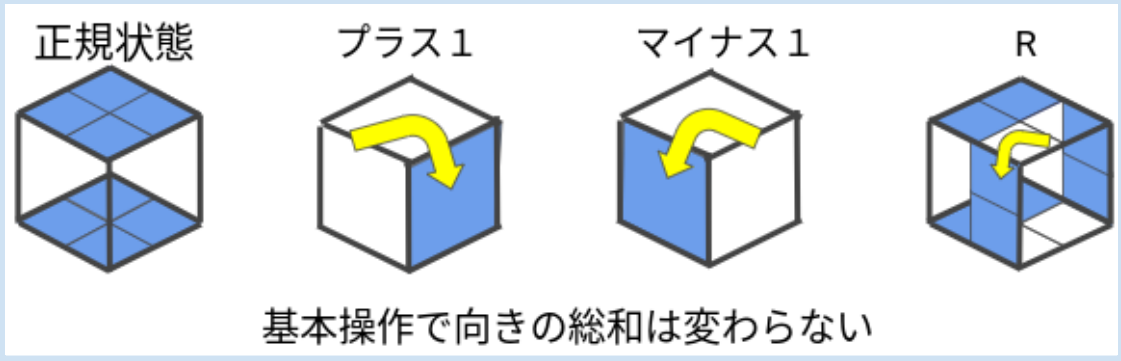
よって次のことを考える。「パリティが起こる要因は向きズレのみであるから、ピースの向きを変化させていきパリティが起こる確率を求める。ひとつのキューブを固定して考える。c＝ｘ個、uc＝ｙ個としてキューブが揃うには

$0 \leqq x + y \leqq 7$（$x + y \neq 1$）ー①
$x \equiv y \pmod 3$ ー②
　を満たすことが必要である。

①を満たすｘ、ｙの組は36組ある。これはパリティを含む。
また①かつ②を満たす組は12組ある。これはパリティを含まない。

このことから、
パリティの起こらない確率【Ｐ①(②)】＝Ｐ(①∧②)/Ｐ(①)
＝12/36
＝1/3(条件つき確率)
であると言える。
ただし、これは十分性の証明であり、必要性の証明が足りていないため、
その証明もしていく。

パーツに存在するU面またはD面の色が、U面かD面を向いている状態を「正規状態」と呼ぶことにし、
正規状態におけるパーツの向きのベクトルを０
時計回りに120°ズレているピースの向きのベクトルを＋１
反時計回りに120°ズレているピースの向きのベクトルをー１
とする。



一般に、基本操作では向きの総和は変わらないことが示されている。
例えば、右上の図は正規状態からRの操作を行ったものでも向きの総和は０になる。
「完成状態での向きの総和は０」
つまり、パリティでない限り、どんな配置でも向きの総和は０なので
向きにおいて完成状態にもっていくための条件はｃとucが３を法に合同となる。
また、これはｄ面u面に限らず成り立つので前のページで仮定した２つの条件と一致する。以上により条件の必要性を確認できたので、前のページで提起した条件が必要十分条件であることが証明できた。
よってパリティは2/3であるため、パリティを除いた計算式は
$3^7 \times 7! \times 1/3 = 3{,}674{,}160$
したがって、総配置数は3,674,160通りである。

〈Ⅱ.プログラミング〉

①ポケットキューブを動かすプログラムの作成

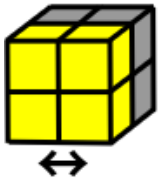
ポケットキューブの動かし方は以下の通りである。

Uの面の動かし方は
右か左のみ

Dの面の動かし方は
右か左のみ

Lの面の動かし方は
上か下のみ

Rの面の動かし方は
上か下のみ



Fの面の動かし方は
右か左のみ

Bの面の動かし方は
右か左のみ

Uの面の動かし方は右と左のみで、右に動かすときUr、左に動かすときUl、180°回転させるときU180°、Dもまた同様である。

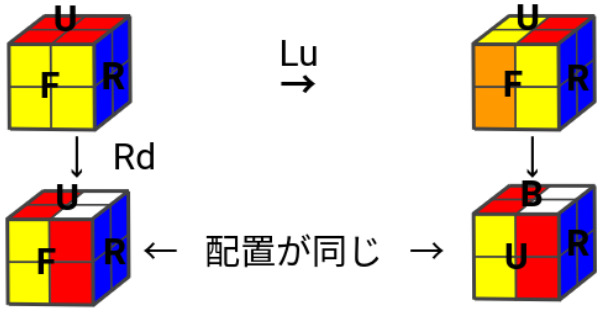
Lの面の動かし方は上か下のみで、上に動かすときLu、下に動かすときLd、180°回転させるときL180°、Rもまた同様である。

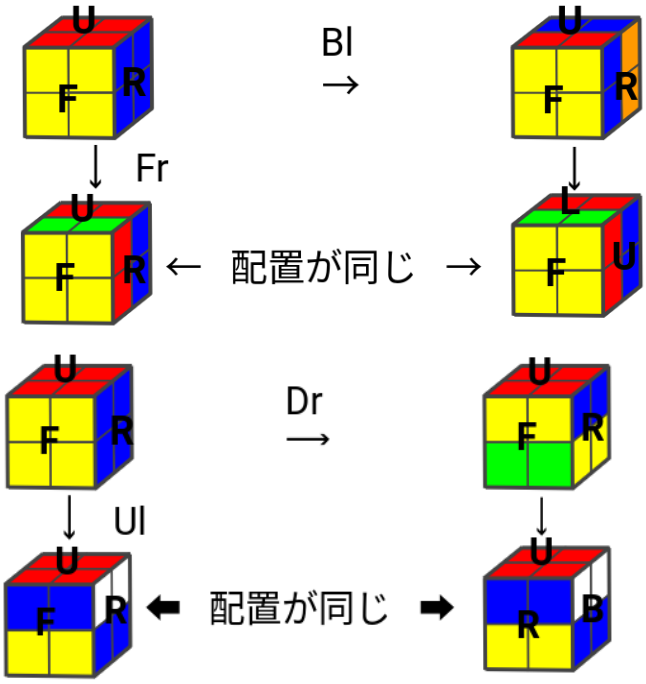
Fの面の動かし方は右か左のみで、右に動かすときFr、左に動かすときFl、180°回転させるときF180°、Bもまた同様である。

つまり、見る方向を変えることで、次のことが成り立つ。

Ur=Dl　Ul=Dr　　U180°=D180°
Ld=Ru　Lu=Rd　　F180°=B180°
Fr=Bl　Fl=Br　　F180°=B180°
※r=右に90° l=左に90° d=下に90° u=上に90°

よって、キューブの動かし方が、9通り(U,F,Rが3つずつ)のときすべての動かし方で動かすことができるため、すべての配置を作ることができる。
また、このとき1つのブロックが全く動かない。)
(B面,D面,L面に含まれるブロック





② n 手を求めるプログラムの作成

すべての色が揃った状態のポケットキューブの8つのブロックの配置
(位置と向き)をキューブを回す動作をn手分変更し、
できた配置が既にあればlistの中に保存せず、なければ保存する。
ｎの値を１から順にこの動作を繰り返し行い、1手目からn手目までの配置数の値が求めた総配置数の値と等しくなったときのnの値を求める。
このプログラムを、ブルートフォース法と呼ばれる方法で行った。
※ブルートフォース法とは
考えられるものすべてを試行する方法。
例:)n=3のとき
上記より動かし方は、9通りあるから、考えられる動かし方の総数は
9^3=729通り
11手目まで試行すると34926271571通り
キューブの定義のコード⇓

```
an=1
bn=2
cn=3
dn=4
en=5
fn=6
gn=7
hn=8
a=["w","b","r",an]
b=["w","g","r",bn]
c=["w","g","o",cn]
d=["w","b","r",dn]
e=["y","b","r",en]
f=["y","g","r",fn]
g=["y","g","o",gn]
cube=[a,b,c,d,e,f,g]
```

```
def front(a):
  for i in range(a):
    for p in range(7):
      if 4 > cube[p][3]:
        cube[p][1],cube[p][2]=cube[p][2],cube[p][1]
        cube[p][3]+=1
      elif 4 == cube[p][3]:
        cube[p][1],cube[p][2]=cube[p][2],cube[p][1]
        cube[p][3]=1
```

```
def right(a):
  for i in range(a):
    for p in range(7):
      if 2==cube[p][3]:
        cube[p][0],cube[p][2]=cube[p][2],cube[p][0]
        cube[p][3]=6
      elif 6==cube[p][3]:
        cube[p][0],cube[p][2]=cube[p][2],cube[p][0]
        cube[p][3]=7
      elif 7==cube[p][3]:
        cube[p][0],cube[p][2]=cube[p][2],cube[p][0]
        cube[p][3]=3
      elif 3==cube[p][3]:
        cube[p][0],cube[p][2]=cube[p][2],cube[p][0]
        cube[p][3]=2
```

```
def up(a):
  for i in range(a):
    for p in range(7):
      if 1==cube[p][3]:
        cube[p][0],cube[p][1]=cube[p][1],cube[p][0]
        cube[p][3]=5
      elif 2 ==cube[p][3]:
        cube[p][0],cube[p][1]=cube[p][1],cube[p][0]
        cube[p][3]=1
      elif 5==cube[p][3]:
        cube[p][0],cube[p][1]=cube[p][1],cube[p][0]
        cube[p][3]=6
      elif 6==cube[p][3]:
        cube[p][0],cube[p][1]=cube[p][1],cube[p][0]
        cube[p][3]=2
```

上の３つはFr Ru Ulの３つの回し方の定義である
このプログラムはキューブの１つのブロックのベクトルに向きと位置の２つの要素を持たせ、キューブを回す動きに合わせて要素を変更してキューブを動かす動作を再現している。

５.結果

Ⅰ.総配置数は　　7!×3^7÷３　　より3674160通り。

Ⅱ.プログラムの作成には成功したが、nの値が大きくなるほど配置がlist内に存在するかどうかを確認する動作にかかる時間が膨大になってしまい、list内の配置の総和が総配置数となるまではいかなかった。具体的には、8手目で1時間、9手目は3時間以上待っても計算が終了せず、さらに時間がかかりそうだったため中断した。したがって、現在到達できているのは8手目までで1159968通りとなった。ここまでで試行し、できた配置は同じものを含めて、48427561通りであった。また、それまでの結果は下表のとおりである。

手数	新規配置数	総配置数
0	1	1
1	9	１０
2	５４	６４
3	３２１	３８５
4	１８４７	２２３２
5	９９９２	１２２２４
6	５０１３６	６２３６０
7	２２７５３６	２８９８９６
8	８７００７２	１１５９９６８

この結果は先行研究で求められていたものと一致し、我々のコードが正しい結果を与えることが検証できた。

Ⅲ.Ⅱが終了していないため行うことができなかった。

６.今後の展望

プログラムの中のポケットキューブの定義自体をコンピュータが処理しやすいものに変更して１つ１つの動作を効率化することで最適化をする。
できたプログラムを用い「God’s number」を求める。
プログラムからキューブを解く最短手順を取り出して、最短手順の法則を見つける。