群论和魔方（P2完）

Original:Accela推箱子 Accela推箱子 6/16

（续前文群论和魔方（P1）……）

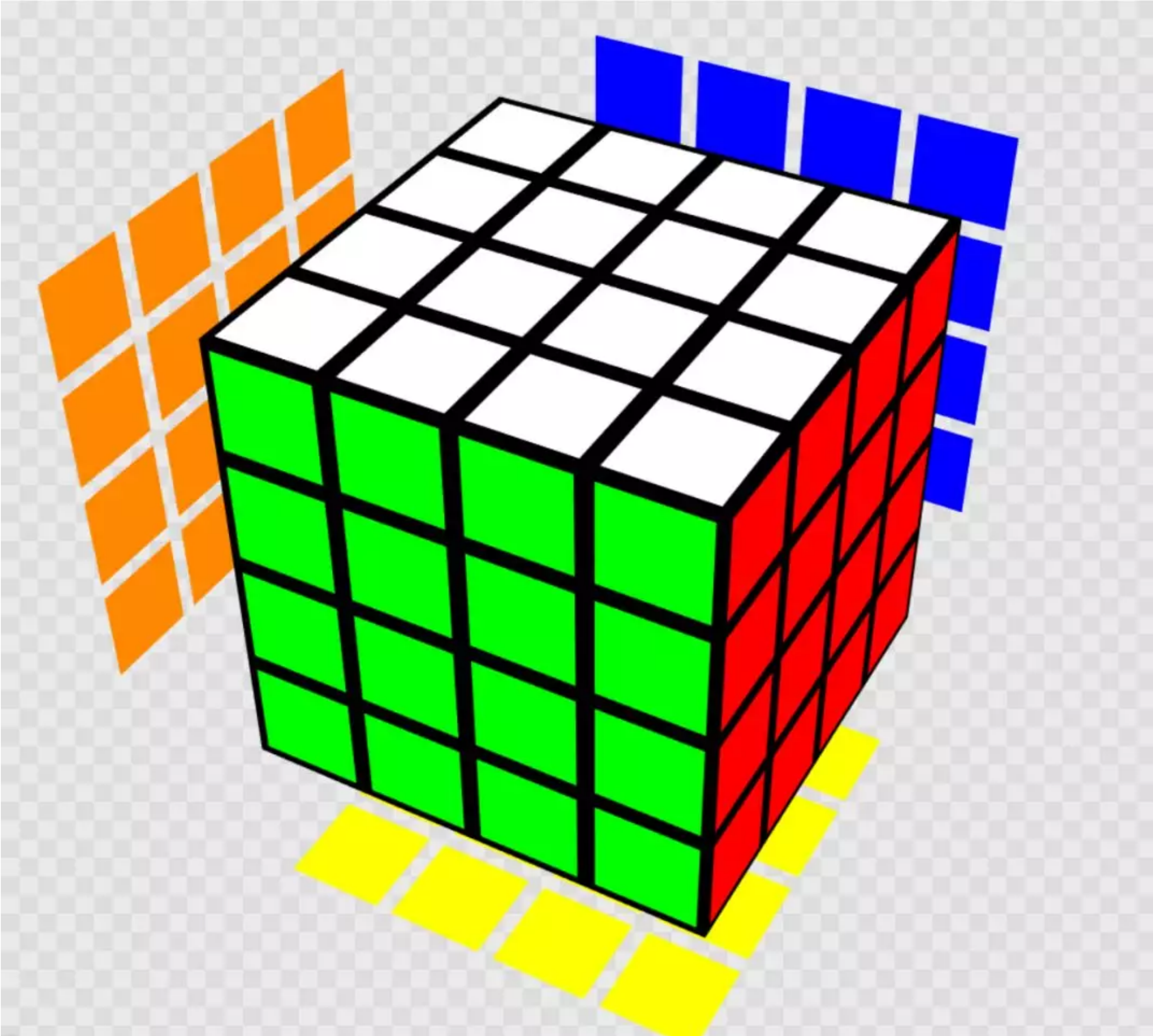
上一篇讲述群论原理，这一篇用程序求解4*4*4魔方。魔方技巧众多，解法多样，本文使用稳定链（Stabilizer Chain）方法。

本文已实现可运行程序上传Github，见后文。

魔方表示

表示法是解决问题的根基。

魔方已经演化出一套标准表示法，可查维基百科。在线魔方ALG.CUBING.NET，可交互，可动画，简单好用。



一般用F、B、L、R、U、D表示前、后、左、右、上、下面。

F表示顺时针转动前第一面90度，2F表示顺时针转动前第二面90度，2F3表示顺时针转动前第二面90度3次，2F3′表示逆时针转动前第二面90度3次。

其它各面都以此类推……

解法框架

本文的魔方解法由下面几部分组装。稳定链是迭代的框架，Schreier子群引理是迭代的桥梁。"稳定子群的商与轨道同构"，则绘制出地图区块。

很多资料也把这套方法称作"Schreier-Sims Algorithm"。

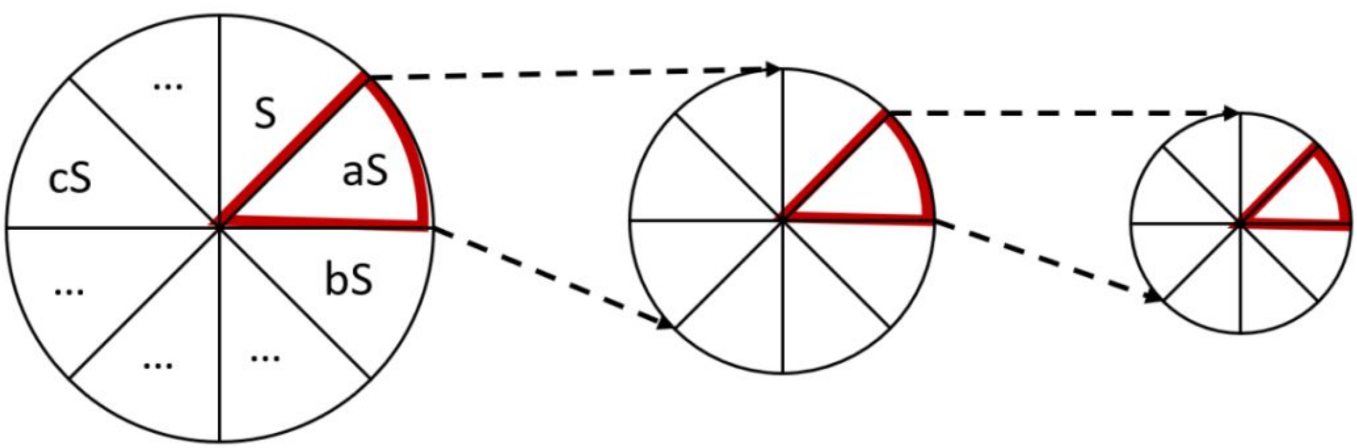
稳定链（Stabilizer Chain）

首先，我们将魔方的第一个方块移到正确位置。接下来的魔方操作（本文中的"魔方操作"，可以是多个魔方旋转的组合；对应魔方群中的一个元素），需保持第一个方块的位置不变——它们是第一个方块的**稳定子群**G1。

然后，我们将第二个方块移到正确位置。接下来的魔方操作，需保持第一、二个方块的位置不变——它们是第一、二个方块的稳定子群G2。注意G2也是G1的稳定子群。

以此类推，我们逐个把魔方方块移到正确位置。移好第i个方块后，接下来的魔方操作，需保持第1..i个方块位置不变——它们是第1..i个方块的稳定子群Gi。

稳定子群G> G1 > G2 > .. > Gi > .. > Gn = I 构成**稳定链**。依照稳定链，我们将魔方方块逐个移正至解完，同时也将魔方群分解。移正方块的顺序可以任意选取。



可以注意到，这个分解方法就是上一篇中的陪集划分，同时也与正规子群的分解相通。

这是我们求解魔方的框架主体。

Schreier子群引理（Schreier Subgroup Lemma）

有了稳定链，接下来的问题是，我们如何从Gi迭代到下一层G(i+1)。

在程序中，我们并不容易穷尽Gi的所有元素，而是用Gi的Generator集合来表示Gi。（用Generator集合中的元素互相运算，可以生成Gi的所有元素，且只生成Gi的元素。）

Schreier子群引理说，给我们Gi的Generator集合，以及G(i+1)的陪集集合（G(i+1)是Gi的稳定子群），我们就以生成G(i+1)的Generator集合。这样，我们就从Gi走到了G(i+1)。

Schreier subgroup lemma:
Let G be a group with a set of generators S. Let H be a subgroup of G, and let R be the set of coset representatives of H in G. For any g in G, let [g] denote the element of R that represents the coset gH, i.e. [g]H = gH, and [g] lies in R.

Then H is generated by the set $\{ [sr]^{-1} (sr) \mid r \text{ in } R, s \text{ in } S \}$.

Proof:
Let h be any element of H. Then it also lies in G, so $h = s_1 s_2 s_3 \ldots s_k$ for some sequence of generators s_i in S.
Let $t_i = [s_{i+1} \ldots s_k]$, the coset representative of $s_{i+1} \ldots s_k$.
Note that $t_k = [e] = e$ by definition, and $t_0 = [s_1 .. s_k] = [h] = e$.
So we can therefore rewrite h as $h = (t_0^{-1} s_1 t_1)(t_1^{-1} s_2 t_2) \ldots (t_{k-1}^{-1} s_k t_k)$.
We also find that $(s_i t_i)H = s_i(t_i H) = s_i (s_{i+1} .. s_k H) = (s_i s_{i+1} .. s_k)H = t_{i-1} H$ so $[s_i t_i] = [t_{i-1}] = t_{i-1}$.
We use this to rewrite h once again, to get $h = ([s_1 t_1]^{-1} s_1 t_1)([s_2 t_2]^{-1} s_2 t_2) \ldots ([s_k t_k]^{-1} s_k t_k)$.
The t_i are coset representatives (by definition) so clearly each factor in the above expression is of the form $[sr]^{-1} (sr)$ with s in S and r in R.
Furthermore these factors are in H because $[sr]^{-1} (sr) H = [sr]^{-1} (sr H) = [sr]^{-1} ([sr]H) = ([sr]^{-1} [sr])H = eH = H$.
Any element h in H is a product of such factors, so it follows that the set $\{ [sr]^{-1} (sr) \mid r \text{ in } R, s \text{ in } S \}$ will generate exactly H.

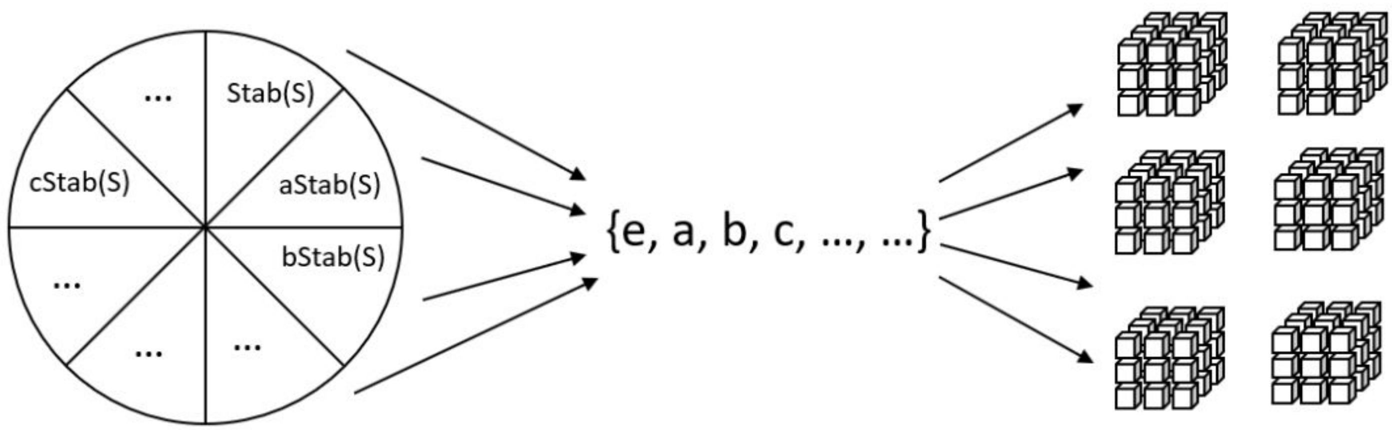
其证明方法来自巧妙的构造。有了Schreier子群引理，我们就有了迭代的桥梁。

稳定子群的商与轨道同构（G/Stab(S)<->Orbit(S)）

Schreier子群引理所需的陪集集合，从哪里来呢？

G(i+1)的陪集集合，就是商Gi/G(i+1)；它与第(i+1)个方块的轨道是同构的。我们只需判断这个方块的位置是否相等，就知道陪集是否相等了。

程序中，通过遍历Generator组合，从而探测所有陪集。陪集用其代表元表示。



另一方面，陪集集合划分了Gi的空间，为我们探明其结构，如地图一般。

给定第(i+1)个方块的某个位置，我们判断出其所属陪集，然后用该陪集代表元的逆来操作，就可以将第(i+1)个方块归位。

陪集集合Gi/G(i+1)就是第i层的地图，而陪集代表元就是每个分区的大门。

总结

求解魔方分为两部分，首先绘制地图。

我们选择一个将魔方方块逐个移回正确位置的顺序，于是有了稳定链G> G1 > G2 > .. > Gi > .. > Gn = I。

在第i层，我们输入Gi的Generator集合（G的Generator集合就是魔方的普通旋转操作），输出G(i+1)的Generator集合，步骤如下：

1. 遍历Gi的Generator集合，利用“稳定子群的商与轨道同构”，探测Gi/G(i+1)的所有陪集
2. 利用“Schreier子群引理”，得到G(i+1)的Generator集合
3. 迭代至下一层(i+1)

上述完成后，我们有了魔方群的地图，它实质是各层Gi的陪集集合。

求解魔方时，对于一个位置杂乱的魔方，我们依照稳定链的顺序，逐个归位各个方块。在第i层，步骤如下：

1. 根据第i个方块的位置，找到G(i-1)/Gi的对应陪集
2. 用该陪集的代表元的逆，操作魔方，则第i个方块回到正确位置
3. 迭代至下一层(i+1)

优化方法

稳定链方法简单强大，可适用任何群，为其绘制地图。但它有计算量问题：

1. 随着逐层迭代，Schreier子群引理生成的Generator数量呈指数增加
2. 随着逐层迭代，Generator的长度也呈指数增加

若直接实现，测试本机计算时间目测超过1年。各种优化后，缩短到5分钟左右。

增量Schreier-Sims算法（Incremental Schreier-Sims algorithm）

Generator数量呈指数增长，但并不是所有Generator都有用。因为最终绘制地图的实质是陪集集合，而没有探测到新陪集的Generator可以舍弃。

增量算法中，将逐层迭代，改成了流式算法（Online Algorithm）。每次输入一个Generator，逐层计算增量部分，尝试探测新的陪集。如果最终没有，则舍弃该Generator。

这样，相比原始算法，无效Generator不会增加后期的计算量。

Jerrum过滤器（Jerrum Filter）

关于减少Generator的数量，更有效的方法是Jerrum过滤器。

任何置换群Sn的子群，Generator集合的大小都小于n。对于魔方，随着更多方块被稳定，其稳定子群对应的n也逐渐减小。

Proposition 2.2 (Jerrum's Filter). *[2] Any subgroup of S_n can be generated by at most $n-1$ elements. Moreover, such a generating set can be found 'on-line', in the sense that if S is a suitable generating set for H and g any element of S_n, then is is possible to find a suitable generating set S' for $\langle H, g\rangle$.*

Jerrum过滤器将Generator映射为有向图中的一条边，通过保持有向图没有环，来控制Generator的总数。如果形成环，总是可以将环替换为另一Generator，而替换总次数是有限的。详见附录链接。

更简单的过滤器Sims Filter是Jerrum的简化版，但Generator总数也更多些。详见附录链接。

映射数组

魔方操作由多个旋转操作组成的列表表示。随着迭代层数更多，魔方操作的长度也呈指数增长，包括Generator。魔方操作的运算量越来越大，如何解决呢？

因为任何群都是置换群的子群，魔方操作无论多长，都可以看作一个置换。用映射数组表示新旧位置的映射，计算量就成了常数。

另一方面，为了知晓映射数组对应的旋转操作列表，也需要记录映射数组由谁运算出来，逐步追溯到原初的旋转操作。整个数据结构成了一株代数算式树，深度与稳定链相当。

其它

除以上之外，还有众多其它优化。详见解魔方程序的Commit History。

解魔方程序

本文已实现可运行程序，上传Github，链接见附录。该程序首先将魔方群分解，绘制地图；之后魔方求解只需查询地图。

此程序可扩展成通用的群算法。因为任意群的元素，都可以用映射数组表示（任意群都是置换群的子群）。不过已经有GAP库了。

稳定链方法善于群分解，但并未指出如何缩减魔方操作的长度（一些陪集代表元对应的魔方旋转次数，甚至超过INT64范围）。此处仍可继续优化，是开放问题。

附录：链接表

微信文章禁止外链，故将资料链接放于此处。

解魔方程序：https://github.com/accelazh/GroupTheory_RubiksCube

在线魔方：https://alg.cubing.net/

稳定链：https://www.jaapsch.net/puzzles/schreier.htm

Schreier子群引理：https://www.jaapsch.net/puzzles/schreier.htm

Jerrum Filter：http://www.m8j.net/data/List/Files-118/Documentation.pdf

Sims Filter：https://mathstrek.blog/2018/06/12/schreier-sims-algorithm/