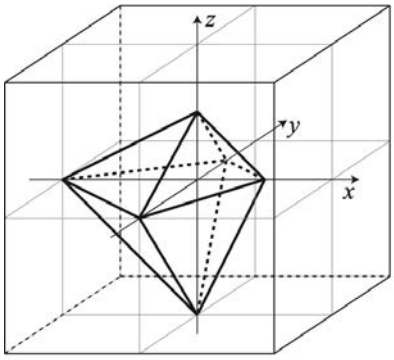
# イマジナリーキューブパズル 3H6Tの解の個数を数える

ここでは，真ん中に T が図のように配置されているか，あるいは，その上下逆さにした位置に配置されている解のみを考えます。




そのように T が置かれていない解がないということは証明できていませんが，ないのではないかと予想しています。この予想が正しいという仮定のもとで，解の個数を求めます。

箱に入れるという問題の性格上，箱の平面的な回転で同じものは，同じとみなすことにします。その上で，

1. 回転や鏡像で同一になるものも区別する。
2. x = y の面で鏡像をとる操作によって同じになるものは同一とみなす。
3. 箱の立方体の空間的な回転で同じになるものは同一とみなすか。
4. 他の面での鏡像をとる操作もあわせて同じになるものは同一とみなすか。

という，それぞれ場合の解の個数も求めます。

## 1. 回転，鏡像を同一視しない場合

まず，真ん中 T は，図のように配置されている場合を考えます。解の個数は，T を上下逆にしたものも同じ個数存在するので，その２倍です。

手前左下から，小立方体に 0,1,2,3,4,5,6,7 の番号をつけます。

H と T は，向きの異なるものに異なる番号をつけて考えます。

- T の向き：小さい三角がどの小立方体の方向を向いているか。 (0,1,2,3,4,5,6,7)
- H の向き：同じく。ただし下向きのみ。それに4 を加えた数。 (8,9,10,11)

真ん中の T によって，それぞれの箱に入るピースは制限されます。変数 possibles は，どの小立方体にはどのピースが入るかをリストで表しています。

In [ ]:

```
possibles = [[0], [1,7], [2,7], [3,5,6,8],
                  [4,7],[3,5,6,8],[3,5,6,8], [0,1,2,4,7,9,10,11]]

assign = [0]*8   #それぞれの箱に入れるピースの番号
solution = []      #解のリスト
def find(n, countT, countH):
```

```
    ''' n 個置かれた状態から先の解をさがす。countT, countH は，それまでに使われた T, H の数''
    if (n == 8):  # 解が見つかった
        solution.append(assign.copy());
        return;
    for p in possibles[n]:
        if (p < 8 and countT < 5):
            assign[n]= p
            find(n+1, countT+1, countH)
        if (p >= 8 and countH < 3):
            assign[n]= p
            find(n+1, countT, countH+1)
find(0,0,0)
print(len(solution))
```

```
256
```

よって， この 2 倍の 512 個の解が存在します。

# 2. x=y の面での鏡像を同一視した解の個数

ある解が存在した時に， 8 番の頂点の方向から見て左右を入れ替えたものは同じ解と考えるのが自然です。 それを同一視した時の個数を数えます。

まず，ある配置に対し，それの鏡像配置を返す関数を作ります。

In [ ]:

```
mirrors = [0,2,1,3,4,6,5,7,8,10,9,11]   # それぞれの鏡像位置。真ん中のTが鏡映面を唯一に定めてい
def mirror(kai):
    ''' kai の配置の鏡像の配置。場所の鏡像とピースの鏡像の両方が関わるので注意'''
    out = [mirrors[kai[mirrors[i]]] for i in range(8)]
    return(out)

# 対称性のあるもの
symmetric = [p for p in solution if (mirror(p) == p)]
print("左右対称性のあるもの:", len(symmetric))

# 対称性のないもの
asymmetric = [p for p in solution if (solution.index(mirror(p)) > solution.index(p))]
print("左右対称性のないもの:", len(asymmetric))
```

```
左右対称性のあるもの: 16
左右対称性のないもの: 120
```

よって， 合わせて 136 個の解が存在することになります。 さらに， 真ん中の T の上下をひっくり返したものを考えると， この 2 倍の 272 個の解があることになります。

これを全て箱に入れて確認したいところですが， 1, 2, 4 の箱の位置は他と独立に考えられるので，これらは 7, 7, 7 のピースに固定したものを考えます。そして，対称性のあるのだけをリストしました。これらは，実際にパズルの箱に入れて，正しいことを確認しました。

In [ ]:

```
# 1,2,4 の位置は独立なので,7,7,7 に固定し, しかも,対称性のないものだけをあげる。
#これは, 実際に箱に入れて正しいことを確認。
u = [p for p in asymmetric if (mirror(p) != p) and p[1] == 7 and p[2] == 7 and p[4] == 7]
print(len(u))
u
```

Out[ ]:

```
14
[[0, 7, 7, 3, 7, 8, 8, 9],
 [0, 7, 7, 5, 7, 8, 8, 9],
 [0, 7, 7, 5, 7, 8, 8, 10],
```

```
 [0, 7, 7, 5, 7, 8, 8, 11],
 [0, 7, 7, 8, 7, 3, 8, 9],
 [0, 7, 7, 8, 7, 3, 8, 10],
 [0, 7, 7, 8, 7, 3, 8, 11],
 [0, 7, 7, 8, 7, 5, 8, 9],
 [0, 7, 7, 8, 7, 5, 8, 10],
 [0, 7, 7, 8, 7, 5, 8, 11],
 [0, 7, 7, 8, 7, 6, 8, 9],
 [0, 7, 7, 8, 7, 6, 8, 10],
 [0, 7, 7, 8, 7, 6, 8, 11],
 [0, 7, 7, 8, 7, 8, 8, 1]]
```

# 3. 立方体の回転での同一性

箱は上があいているので, 上下の区別がありますが, 立方体としてみなした時にはその区別はありません。 箱の上をふさいで, 箱を空間的に回転させることにより同じものは同一とみなして, 解の個数を求めましょう。 これは, 数学的には, 立方体の回転全体のなす群 (S4 と呼ばれています) の作用で同じものは同一視するという数え方です。

真ん中の T の上下をひっくり返したものは, x 軸の周りの 180 度回転で行き合うことができるので, 真ん中の T が上下逆のものは考える必要がありません。最初の 256 個の中で, 真ん中の T *を変化させないような回転で同一になるものを同一視して考えればいいことになります。それは,* T の軸の周りの 120 度回転です。右回転と左回転がありますが, 右回転で同じものをリストします。

In [ ]:
```
rotation = [0,2,4,6,1,3,5,7,8,9,10,11]
invrotation = [0,4,1,5,2,6,3,7,9,10,11]
def rotate(kai):
    out = [rotation[kai[invrotation[i]]] for i in range(8)]
    return(out)

# 回転対称性のあるもの
rsymmetric = [p for p in solution if (rotate(p) == p)]
print("回転対称性のあるもの:", len(rsymmetric))

# 回転対称性のないもの
rasymmetric = [p for p in solution if (
    solution.index(rotate(p)) >= solution.index(p)) and
    solution.index(rotate(rotate(p))) > solution.index(p)]
print("回転対称性のないもの:", len(rasymmetric))
```

```
回転対称性のあるもの: 4
回転対称性のないもの: 84
```

In [ ]:
```
rsymmetric
```

Out[ ]:
```
[[0, 1, 2, 8, 4, 8, 8, 0],
 [0, 1, 2, 8, 4, 8, 8, 7],
 [0, 7, 7, 8, 7, 8, 8, 0],
 [0, 7, 7, 8, 7, 8, 8, 7]]
```

よって, 合わせて 88 個あります。 4 + 84 * 3 = 256 なので, 計算はあっています。

# 4. 立方体の回転, 面対象で同一になるものは同一視する

回転，および面対象の操作の組み合わせにより同一になるものは同一と考えて数え直しましょう。 ここで，組み合わせとして考えないといけなおものは，中心の T を変えない変換を考えると， 6 個(0, 120, 240 度回転および，その後に鏡像をとったもの) だと分かります(すなわち，D6 という位数 6 の２面体群)。

多少汚い方法ですが，回転などで作られたもののインデックスが全て自分以上ということで，その代表元を判断することにします。

In [ ]:
```
final = [solution[i] for i in range(256) if (
    solution.index(rotate(solution[i])) >= i and
    solution.index(rotate(rotate(solution[i]))) >= i and
    solution.index(mirror(solution[i])) >= i and
    solution.index(mirror(rotate(solution[i]))) >= i and
    solution.index(mirror(rotate(rotate(solution[i])))) >= i)]
print("全ての対称性を考慮した個数:", len(final))
```

全ての対称性を考慮した個数: 52

In [ ]:
```
final
```

Out[ ]:
```
[[0, 1, 2, 3, 4, 8, 8, 9],
 [0, 1, 2, 3, 4, 8, 8, 11],
 [0, 1, 2, 3, 7, 8, 8, 9],
 [0, 1, 2, 3, 7, 8, 8, 11],
 [0, 1, 2, 5, 4, 8, 8, 9],
 [0, 1, 2, 5, 4, 8, 8, 10],
 [0, 1, 2, 5, 4, 8, 8, 11],
 [0, 1, 2, 5, 7, 8, 8, 9],
 [0, 1, 2, 5, 7, 8, 8, 10],
 [0, 1, 2, 5, 7, 8, 8, 11],
 [0, 1, 2, 8, 4, 8, 8, 0],
 [0, 1, 2, 8, 4, 8, 8, 1],
 [0, 1, 2, 8, 4, 8, 8, 7],
 [0, 1, 2, 8, 7, 3, 8, 9],
 [0, 1, 2, 8, 7, 3, 8, 10],
 [0, 1, 2, 8, 7, 3, 8, 11],
 [0, 1, 2, 8, 7, 5, 8, 9],
 [0, 1, 2, 8, 7, 5, 8, 10],
 [0, 1, 2, 8, 7, 5, 8, 11],
 [0, 1, 2, 8, 7, 6, 8, 9],
 [0, 1, 2, 8, 7, 6, 8, 10],
 [0, 1, 2, 8, 7, 6, 8, 11],
 [0, 1, 2, 8, 7, 8, 8, 0],
 [0, 1, 2, 8, 7, 8, 8, 1],
 [0, 1, 2, 8, 7, 8, 8, 4],
 [0, 1, 2, 8, 7, 8, 8, 7],
 [0, 1, 7, 3, 7, 8, 8, 9],
 [0, 1, 7, 3, 7, 8, 8, 10],
 [0, 1, 7, 3, 7, 8, 8, 11],
 [0, 1, 7, 5, 7, 8, 8, 9],
 [0, 1, 7, 5, 7, 8, 8, 10],
 [0, 1, 7, 5, 7, 8, 8, 11],
 [0, 1, 7, 6, 7, 8, 8, 9],
 [0, 1, 7, 6, 7, 8, 8, 10],
 [0, 1, 7, 6, 7, 8, 8, 11],
 [0, 1, 7, 8, 7, 8, 3, 9],
 [0, 1, 7, 8, 7, 8, 3, 10],
 [0, 1, 7, 8, 7, 8, 3, 11],
 [0, 1, 7, 8, 7, 8, 6, 9],
 [0, 1, 7, 8, 7, 8, 6, 11],
```

```
[0, 1, 7, 8, 7, 8, 8, 0],
[0, 1, 7, 8, 7, 8, 8, 1],
[0, 1, 7, 8, 7, 8, 8, 2],
[0, 1, 7, 8, 7, 8, 8, 7],
[0, 7, 7, 3, 7, 8, 8, 9],
[0, 7, 7, 3, 7, 8, 8, 11],
[0, 7, 7, 5, 7, 8, 8, 9],
[0, 7, 7, 5, 7, 8, 8, 10],
[0, 7, 7, 5, 7, 8, 8, 11],
[0, 7, 7, 8, 7, 8, 8, 0],
[0, 7, 7, 8, 7, 8, 8, 1],
[0, 7, 7, 8, 7, 8, 8, 7]]
```

これで, 真ん中に T を配置する解を全てあげることができました。

In [ ]:

---

京都大学人間・環境学研究科
立木 秀樹