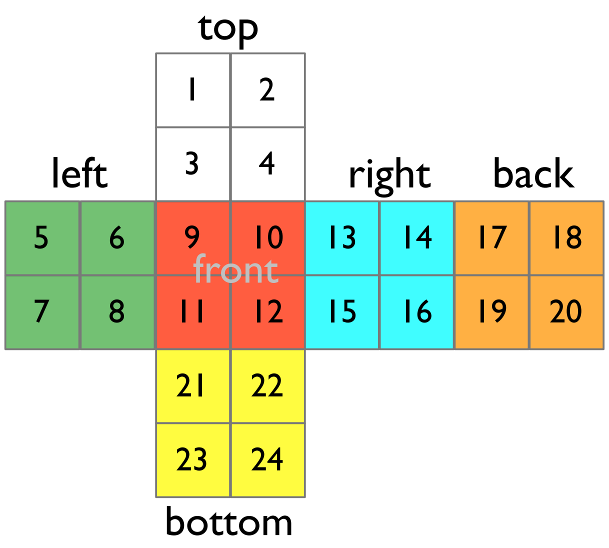
Mathematics 676 **The $2 \times 2 \times 2$ Rubik's Cube in** GAP A. Hulpke

As an example of a more involved puzzle consider the $2 \times 2 \times 2$ Rubik's cube. We label the facelets of the cube in the following way:



We now assume that we will fix the bottom right corner (i.e. the corner labelled with 16/19/24) in space – this is to make up for rotations of the whole cube in space. We therefore need to consider only three rotations, front, top and left. The coresponding permutations are (for clockwise rotation when looking at the face):

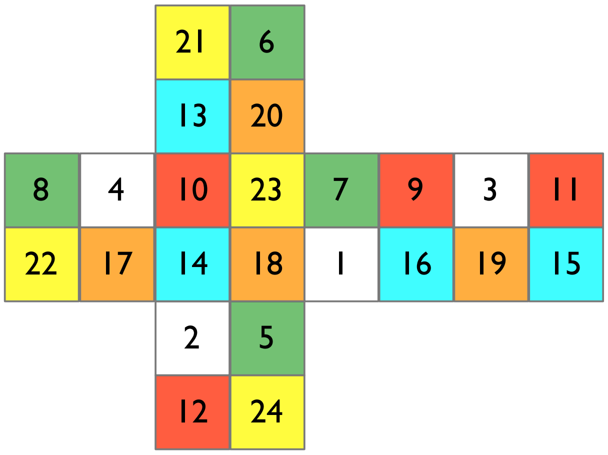
```
gap> top:=(1,2,4,3)(5,17,13,9)(6,18,14,10);;
gap> left:=(1,9,21,20)(5,6,8,7)(3,11,23,18);;
gap> front:=(3,13,22,8)(4,15,21,6)(9,10,12,11);;
gap> cube:=Group(top,left,front);
Group([(1,2,4,3)(5,17,13,9)(6,18,14,10),(1,9,21,20)(3,11,23,18)(5,6,8,7),
  (3,13,22,8)(4,15,21,6)(9,10,12,11) ])
gap> Order(cube);
3674160
```

By defining a suitable homomorphism first (for the time being consider this command as a black box – a free group is a group generated by formal symbols) we can choose nicer names – T, L and F – for the generators:

```
gap> map:=EpimorphismFromFreeGroup(cube:names:=["T","L","F"]);
[ T, L, F ] -> [ (1,2,4,3)(5,17,13,9)(6,18,14,10),
  (1,9,21,20)(3,11,23,18)(5,6,8,7), (3,13,22,8)(4,15,21,6)(9,10,12,11) ]
```

We now can use the command `Factorization` to express permutations in the group as word in generators. This is done using an orbit algorithm on elements, the group is just small enough that this is feasible.

The *reverse* sequence of the inverse operations therefore will turn the cube back to its original shape. For example, suppose the cube has been mixed up in the following way:



This corresponds to the permutation

```
gap> move:=(1,15,20,4,6,2,21)(3,17,8,5,22,7,13)(9,14,11,18,12,23,10)
```

(1 has gone in the position where 15 was, 2 has gone in the position of 21, and so on.) We express this permutation as word in the generators:

```
gap> Factorization(cube,move);
T*F*L*T*F*T
```

We can thus bring the cube back to its original position by turning each *counter*clockwise top,front,top,left,front,top.

Larger puzzles If we want to do something similar for larger puzzles, for example the $3 \times 3 \times 3$ cube, the algorithm used by `Factorization` runs out of memory. Instead we would need to use a stabilizer chain. The algorithm used then does not guarantee any longer a shortest word, in our example:

```
gap> PreImagesRepresentative(map,move);
T*L^-2*T^-1*L*T*L^-2*T^-2*F*T*F^-1*L^-1*F*T^-1*F^-1*L*T*L*
T^-2*F*T*F^-1*T*F*T^-1*F^-2*L^-1*F^2*L
```

(Note that the code uses some randomization, your mileage may vary.)