hmm, I think there are some symmetries we can take advantage of to optimize the problem. I went ahead and took a look at $n = 4$. I took to writing out a tree of all permutations, but that quickly got annoying and I noticed that some orders of flips will lead to the same results.

$$T(1,2,3,4) = (4,3,2,1) \quad F(1,2,3,4) = (1,3,2,4) \quad G(1,2,3,4) = (4,3,2,1)$$
$$G = T \circ F = F \circ T$$

So we can think of this as transformations on our original list `[1,2,3,4]`.

Now we can represent these permutations as the matrix transformation:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \end{bmatrix}$$

Now we know that $4! = 24$, but in fact there are actually less transformations necessary to describe all the permutations. These are all the matrices we can form for the transformations:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

These **7** transformations form a group under matrix multiplication and any possible permutations of (1,2,3,4) is the result of the composition of these 7 transformations.

Now let's do the same for 3x3:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Now if you notice, all of these transformations are essentially subtransformations in our n=4 transformation Group.

We can represent this as the direct sum of our matrices:

$$A \oplus B = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$$

So for example:

$$[1] \oplus \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And for our n=3 group, we can also represent these as the direct some of n=1 and n=2 matrices, and n=2 as the direct sum of n=1 matrices. Now this is where things get interesting because we can represent some matrices in our n=4 group as the direct sum of smaller matrices of n=1, n=2, and n=3. So for example:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \oplus \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

So what we've done is found a way to describe the permutation transformations for n=4 in terms of n=3,2,1. Without loss of generality we could have essentially continued this for n=5 and in n=k.

Additionally, some of the transformations are *flipped* versions of each other:

$$\begin{aligned} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} &= \text{fliph} \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right) \\ &= \text{fliph} \left([1] \oplus \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \oplus [1] \right) \\ &= \text{fliph}([1] \oplus \text{fliph}([1] \oplus [1]) \oplus [1]) \end{aligned}$$

Which is equivalent to multiplying by the matrix:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

There are also some vertical flips, but we can essentially write all the transformations in n=4 as the horizontal and vertical flipped direct sums of 4 n=1 transformations.

Now let's take a look at n=3 again:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

If you notice we only actually have 2 unique transformations and they're both flipped.

In fact we can reduce n=3 to just these transformations:

$$\{[1] \oplus [1] \oplus [1],\ [1] \oplus H([1] \oplus [1])\}$$

I simplified the notation for vertical (V) and horizontal (H) flipping.

This group is acted on by the group of V and H under composition:

$$\{e, V, H\}$$

where e is our identity, so no flipping essentially. Compositions of V and H acting on our groups will give us the other two transformations for n=3.

Now one thing to note here, let's take a quick look at n=2:

$$\{[1] \oplus [1]\}$$

All transformations on n=2 come from applying {e,V,H} on this set. And if you look even closer you'll see that it's the direct sum of `[1]` on each element of the action of our flipping group on the group of transformations of n=2.

We can even do this for n=4:

$$\begin{Bmatrix} [1] \oplus [1] \oplus [1] \oplus [1], & [1] \oplus H([1] \oplus [1] \oplus [1]), \\ [1] \oplus H([1] \oplus [1]) \oplus [1], & [1] \oplus [1] \oplus H([1] \oplus [1]) \end{Bmatrix}$$

I'm going to simplify this notation to be:

$$[1] \oplus \begin{Bmatrix} [1] \oplus [1] \oplus [1], \\ [1] \oplus H([1] \oplus [1]) \end{Bmatrix}$$

Which is:

$$[1] \oplus \{[1] \oplus \{[1] \oplus [1]\}\}$$

We can basically construct the transformation group for n=k as the direct sum of `[1]` and the transformation group n=k-1:

$$\text{Perms}(k) := 1 \oplus \text{Perms}(k-1)$$

Okay so I know this kind of got a bit abstract so let's look back at an example: So remember, all permutations of n=3 look like:

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

I'm going to abuse notation and just write these like this:

$$1 \oplus \begin{matrix} 2 & 3 \\ 3 & 2 \end{matrix} \quad 2 \oplus \begin{matrix} 1 & 3 \\ 3 & 1 \end{matrix} \quad 3 \oplus \begin{matrix} 1 & 2 \\ 2 & 1 \end{matrix}$$

and we can express the n=4 group as:

$$1 \oplus \text{ perms}(2,3,4) \quad 2 \oplus \text{ perms}1,3,4 \quad 3 \oplus \text{ perms}1,2,4 \quad 4 \oplus \text{ perms}1,2,3$$

Notice how this is the direct sum of the value and all permutations with that value removed?

As you can see we've found a recursive way to express our transformations, and it's important to notice that this won't generate any dublicate values.

So now it's only a matter of figuring out how to code this. Well this is actually really simple.

We'll start by defining a permutation function

```
def perm(l: list) -> list:
  ...
```

Then we set the base case for a list of length one and two:

```
if len(l) == 1:
  return l
elif len(l)==2:
  return [l[0],l[1]], [l[1],l[0]]
```

For 2 numbers we really are just taking our numbers and then swapping them.

Now what we want to do next is make a function that will remove a value from a list, here's what I came up with:

```
def remove(i,l):
  return [x for x in l if x!=i]
```

This function will be used to remove the specified value so we can add it to our list.

Now we have enough to cover n>2:

```
else:
  return [[i]+p for i in l for p in perm(remove(i,l))]
```

So our full function looks like:

```
def remove(i: int,l: list) -> list:
  return [x for x in l if x!=l]

def perm(l: list) -> list:
  if len(l) == 1:
```

```
    return l
  elif len(l)==2:
    return [l[0],l[1]], [l[1],l[0]]
  else:
    return [[i]+p for i in l for p in perm(remove(i,l))]
```

This will work for any list of values `[a,b,c]` too.