CHAPTER 1

Permutations and Sorting

In the previous chapter, we talked about permutations. If you have a list of four letters, like $[a, b, c, d]$, you can rearrange them in $4!$ ways:

a,b,c,d	a,b,d,c	a, d, b, c	a, d, c, b	a, c, b, d	a, c, d, b
b,a,c,d	b,a,d,c	b, d, a, c	b, d, c, a	b, c, a, d	b, c, d, a
c,b,a,d	c,b,d,a	c, d, b, a	c, d, a, b	c, a, b, d	c, a, d, b
d,b,c,a	d,b,a,c	d, a, b, c	d, a, c, b	d, c, b, a	d, c, a, b

You can make Python generate all the permutations for you:

```
from itertools import permutations
all_permutations = permutations(('a', 'b', 'c', 'd'))
for p in all_permutations:
    print(p)
```

1.1 Notation

How do we define or write down a single permutation? You could say something like "Swap the first and second items and swap the third and fourth items." However, that gets pretty difficult to read, so we usually write a permutation as two lines: the first line is before the permutation and the second line is after. Like this:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$$

We can also assign permutations to variables. For example, if we wanted the variable A to represent "swapping the first and second item", we would write this:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix}$$

And if we wanted B to represent "swapping the third and fourth item", we would write:

$$B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 4 & 3 \end{pmatrix}$$

Now, we can *compose* permutations together. For example, we might say:

$$B \circ A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$$

In other words, if we have the list $[a, b, c, d]$ and we apply permutation A, followed by permutation B, we get $[b, a, d, c]$.

Important: Note that permutations are applied from right to left. $B \circ A$ means "Applying A and then B." Why does this matter? Permutations are not necessarily commutative. That is, if you have two permutations S and T, $S \circ T$ is not always the same as $T \circ S$.

Also, note that "don't change anything" is a permutation. We call it *the identity permutation*. If you have four items, the identity permutation would be written:

$$I = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

(We use a capital "I" for the identity.)

1.1.1 Challenge

Find an example of two permutations S and T, such that $S \circ T$ does not equal $T \circ S$.

1.2 Sorting in Python

One of the common forms of permutation in software is sorting. Sorting is putting data in a particular order. For example, in Python, if you had a list of numbers, you can sort it in ascending order like this:

```
my_grades = [92, 87, 76, 99, 91, 93]
grades_worst_to_best = sorted(my_grades)
```

Do you want to sort backwards?

```
my_grades = [92, 87, 76, 99, 91, 93]
grades_best_to_worst = sorted(my_grades, reverse=True)
```

Note that `sorted` makes a new list with the correct order. If you want to sort the array in place, you can use the `sort` method:

```
my_grades = [92, 87, 76, 99, 91, 93]
my_grades.sort(reverse=True)
```

1.3 Inverses

Think for a second about this permutation:

$$S = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 2 & 1 \end{pmatrix}$$

You could say this permutation shuffles a list. What is its inverse? That is, what is the permutation that unshuffles the items back to where they were originally?

$$S^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix}$$

In other words, the original moved an item in the first spot to the third spot; the inverse must move whatever was in the third spot back to the first spot.

(Notation note: In multiplication, $b \times b^{-1} = 1$, so we use "to the negative one" to indicate inverses in lots of places.)

Mechanically, how do you find the inverse? Flip the rows, then sort the columns using the top number:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 2 & 1 \end{pmatrix} \text{ flip } \rightarrow \begin{pmatrix} 3 & 4 & 2 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix} \text{ sort } \rightarrow \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix}$$

Let's say you have two permutations A and B. Permuting by B then A would look like this:

$$C = A \circ B$$

If you know A^{-1} and B^{-1}, what is C^{-1}? You would undo-A, then undo-B, so

$$C^{-1} = B^{-1} \circ A^{-1}$$

1.4 Cycles

Here is a permutation:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 5 & 1 & 3 \end{pmatrix}$$

When this is applied, whatever is at 1 gets moved to 2, 2 gets moved to 4, and 4 gets moved to 1. That is a *cycle*: $1 \to 2 \to 4$, then it goes back to 1. It involves three locations, so we say it is a *3-cycle*.

There is another cycle in this permutation: $3 \to 5$, then it goes back to 3.

Because these cycles share no members, we say the cycles are *disjoint*.

Every permutation can be broken down into a collection of disjoint cycles.

$$T = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 5 & 1 & 3 \end{pmatrix} = (1 \to 2 \to 4)(3 \to 5)$$

The first handy thing about this notation is that it makes it easy for us to describe the inverse. We just run the cycles backward:

$$T^{-1} = (4 \to 2 \to 1)(5 \to 3)$$

Starting with the list $[a, b, c, d, e]$, let's repeatedly apply the permutation T

Initial	a, b, c, d, e
T applied	d, a, e, b, c
$T \circ T$ applied	b, d, c, a, e
$T \circ T \circ T$ applied	a, b, e, d, c
$T \circ T \circ T \circ T$ applied	d, a, c, b, e
$T \circ T \circ T \circ T \circ T$ applied	b, d, e, a, c
$T \circ T \circ T \circ T \circ T \circ T$ applied	a, b, c, d, e

This permutation results in six combinations, then it loops back on itself. The number of combinations is the least common multiple of all the cycles. In this case, there is a 3-cycle and a 2-cycle. The least common multiple of 2 and 3 is 6.

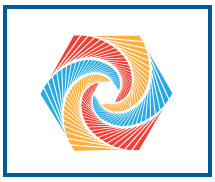
1.5 Further Learning

Check out this leetcode problem which combines permutations and programming! https://leetcode.com/problems/permutations/description/

This is a draft chapter from the Kontinua Project. Please see our website (https://kontinua.org/) for more details.

Appendix A

Answers to Exercises



Index