We started by reviewing the idea of **function composition**.

Let \( f : A \rightarrow B \) and \( g : B \rightarrow C \) be functions (ie. \( f \) is a function from set \( A \) to set \( B \), and \( g \) is a function from set \( B \) to set \( C \)) then we write the **composition of \( g \) and \( f \) as \( g \circ f \)**. \( g \circ f \) is a function from \( A \) to \( C \) (in notation \( g \circ f : A \rightarrow C \)) such that \( \forall \, a \in A, \, (g \circ f) (a) = g(f(a)) \)

In plain English, when we see \( g \circ f \) we just have to remember that it means, “apply \( f \), then apply \( g \) to the result of that”. The key thing to remember is that the first function we apply is the last one listed.

Note that when \( g \circ f \) is well-defined, \( f \circ g \) may not be defined at all. To compose two functions, the “target set” of the first one we apply must match the “input set” of the second one we apply.

Now on to permutations. We’ve seen the word “permutation” before, in the context of counting the number of different linear arrangements of \( n \) distinct objects. For the next couple of classes we are going to define the concept of a permutation precisely, using our established understanding of relations and functions. We’ll discuss the rudiments of a system of mathematics in which permutations are the fundamental objects. The idea of creating meaningful mathematical systems for things that are not numbers is of fundamental importance in discrete mathematics.

**Definition:** A **permutation** is a bijection from a set to itself.

For example, let the set \( A = \{ a, \text{red}, 3, \alpha \} \). One permutation of \( A \) is the bijection defined by the ordered pairs \( \{ (a,3), (\text{red},a), (3, \alpha), (\alpha, \text{red}) \} \) -- make sure that you agree that this is a bijection.

When we are studying permutations the objects in the set don’t usually matter – what really matters is the size of the set. For this reason, when we talk about permutations the set \( A \) is usually just \( \{ 1, 2, 3, \ldots, n \} \) for some value of \( n \).

We use \( S_n \) to represent the set of all permutations of the set \( \{ 1, 2, 3, \ldots, n \} \)
One of the first questions we can ask is, what is $|S_n|$? It is easy to answer: The number of ways to create an ordered pair $(1, x)$ (where $x$ represents an element of $\{1, 2, \ldots, n\}$) is $n$. For each of those there are $n-1$ ways to create an ordered pair $(2, y) \ldots$ and so on. The result is $n!$

Consider the permutation of $\{1,2,3,4\}$ defined by $\{(1,4), (2,1), (3,3), (4,2)\}$ Notice that under this function, $3$ maps to itself. This is perfectly fine. In fact, there is a permutation that changes nothing: $f(x) = x$ for all $x$. For $\{1,2,3,4\}$ the ordered pairs for this permutation are $\{(1,1), (2,2), (3,3), (4,4)\}$. This is called the **identity permutation**, and we represent it with the Greek letter $\iota$ which looks like this: $\iota$. It’s basically $i$ without the dot.

In fact we almost always use Greek letters to name permutations: $\pi$ (pi), $\sigma$ (sigma), and $\tau$ (tau) are among the favourites.

Permutations can be represented in a variety of ways. So far we have just listed the ordered pairs, but we can also use an $n$-by-$n$ matrix, a diagram that shows the mapping of the set onto itself, or a 2-by-$n$ matrix. For example, the permutation $\{(1,4), (2,1), (3,3), (4,2)\}$ can also be represented as

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

in which each row corresponds to the first element in one of the ordered pairs, and each column corresponds to the second element. A “1” in the matrix indicates that the elements represented by the row and the column form an ordered pair. For example, there is a “1” in the second row and first column, so we know $(2,1)$ is one of the ordered pairs in the permutation.

The diagram representation is very hard to show in a text document, but you can find it in the text: two vertical sets of labelled points, with arrows going from points in the first set to points in the second.

The 2-by-$n$ matrix representation of this permutation looks like this:

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

in which each column represents one of the ordered pairs in the permutation.
It’s important to understand that each of these representations contains exactly the same information (they define the same permutation) and that if we are given any one of them we can construct all the others.

If we look at the 2-by-n matrix representation for different members of \( S_n \) such as

\[
\begin{bmatrix}
1 & 2 & 3 & 4 \\
4 & 1 & 3 & 2
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
1 & 2 & 3 & 4 \\
2 & 3 & 4 & 1
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
1 & 2 & 3 & 4 \\
3 & 4 & 1 & 2
\end{bmatrix}
\]

we can see that the first line is always the same. So we can leave it out! We represent those permutations by

\[
\begin{bmatrix}
4 & 1 & 3 & 2 \\
2 & 3 & 4 & 1
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
3 & 4 & 1 & 2
\end{bmatrix}
\]

I will call this the **standard notation** for a permutation of \( \{1, \ldots, n\} \) because it is used very widely ... but as we will see, there is another notation that is often more useful in practice.

There are some basic facts about \( S_n \) that we need to have in hand:

1. If \( \pi \in S_n \) and \( \sigma \in S_n \) then \( \pi \circ \sigma \in S_n \)

2. If \( \pi \in S_n \) and \( \sigma \in S_n \) and \( \tau \in S_n \) then \( \pi \circ (\sigma \circ \tau) = (\pi \circ \sigma) \circ \tau \)

3. If \( \pi \in S_n \) then \( \pi \circ \iota = \iota \circ \pi = \pi \)

4. If \( \pi \in S_n \) then \( \pi^{-1} \in S_n \) and \( \pi \circ \pi^{-1} = \pi^{-1} \circ \pi = \iota \)

Each of these follows from the definition of permutation and the properties of functions. I recommend that you do some examples and convince yourself that these are true.

We should stop for a moment and think about what \( \pi \circ \sigma \) means. Remember that permutations are functions. For example, if \( \pi = \begin{bmatrix} 2 & 3 & 1 \end{bmatrix} \) we can think of \( \pi \) as a function where \( \pi(1) = 2, \pi(2) = 3 \) and \( \pi(3) = 1 \). So if \( \pi \) and \( \sigma \) are permutations, they are functions, and \( \pi \circ \sigma \) is also a function ... specifically, it is the function defined by \( (\pi \circ \sigma)(x) = \pi(\sigma(x)) \)
Cycle Notation for Permutations

Now we introduce another representation for permutations ... one that makes it possible to work with permutations very easily.

Consider this permutation:

\[ \pi = [4 \ 1 \ 5 \ 2 \ 7 \ 3 \ 6] \]

What happens if we imagine composing \( \pi \) with itself? Let’s trace what happens to the element 1. We are going to apply \( \pi \) twice: the first application maps 1 to 4, and the second application maps that 4 to 2. If we compose with \( \pi \) again, that 2 is mapped back to 1. Treating \( \pi \) as a function, we see \( \pi(1) = 4 \), \((\pi \circ \pi)(1) = 2\), and \((\pi \circ \pi \circ \pi)(1) = 1\)

Composing with \( \pi \) even more times will cycle through 4 then 2 then 1 then 4 then 2 then 1 etc. We can write this behaviour as \( 1 \rightarrow 4 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 1 \) etc.

If we trace what happens to 2 when we repeatedly compose \( \pi \) with itself and apply the resulting function to 2, we see exactly the same pattern: \( 2 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 2 \) etc. The same thing happens if we start with 4 and trace what happens to it when we repeatedly compose \( \pi \) with itself and apply the resulting function to 4: we get the pattern \( 4 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 1 \rightarrow 4 \) etc.

So in this sense, 1, 4 and 2 form a cycle: 1 goes to 4, 4 goes to 2, and 2 goes to 1. We write this cycle as \((1, 4, 2)\) - it is a notational device that describes the three ordered pairs \((1,4),(4,2),(2,1)\) which do indeed belong to \( \pi \).

What about the rest of \( \pi \)? Since we have dealt with 1, 2 and 4, let’s see what happens to 3. Following the same analysis as we did for 1, 2 and 4 (but skipping over some of the details) we see this pattern: \( 3 \rightarrow 5 \rightarrow 7 \rightarrow 6 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 6 \rightarrow 3 \) etc. which we write as \((3, 5, 7, 6)\). Again we can see that this is a non-ambiguous shorthand way to represent the four ordered pairs \((3,5),(5,7),(7,6),(6,3)\) that make up the rest of \( \pi \).

Thus we can express the entire definition of \( \pi \) with the two cycles \((1,4,2)(3,5,7,6)\) - we call this the cycle notation for \( \pi \) All of the information that defines \( \pi \) is there, expressed in a different way (in other words, we can reconstruct the standard representation from the cycle notation).

Notice that from each permutation, we can only get one cycle notation version. (We saw this for \( \pi \) above: no matter which elements we start with, we always get the same repeating patterns.) Similarly, from any cycle notation representation, we can only reconstruct one permutation in standard notation. This means that the cycle notation for each permutation...
is unique (up to changing the order of the cycles, because \((1,4,2)(3,5,7,6)\) gives the same information as \((3,5,7,6)(1,4,2)\) and up to rotating the elements within each cycle, since \((1,4,2)\) and \((4,2,1)\) and \((2,1,4)\) all represent the same information).

So what can we do with a permutation expressed in cycle notation?

**Computing the Inverse of a Permutation**

Suppose we have a permutation \(\pi\) and we need to compute \(\pi^{-1}\). We could do it from standard representation ...

For example, consider \(\pi = [4 \ 1 \ 5 \ 2 \ 7 \ 3 \ 6]\) To compute \(\pi^{-1}\), we could find the “1”, see that it is in the second position, so the ordered pair (1,2) must be in \(\pi^{-1}\). Similarly, “2” is in the fourth position, so the ordered pair (2,4) must be in \(\pi^{-1}\), and so on ... it's not that hard but it's a bit tedious.

But with the permutation expressed in cycle notation, computing \(\pi^{-1}\) is trivially easy: we just reverse each cycle. So the inverse of \((1,4,2)(3,5,7,6)\) is simply \((2,4,1)(6,7,5,3)\). You can check the details of this example to confirm that it works, but the logic of it is pretty straightforward: \(\pi\) contains the ordered pair (1,4) – this is encoded in the first cycle, so \(\pi^{-1}\) must contain the ordered pair (4,1) – and this is encoded in the reverse of the first cycle.

**Composition of Two Permutations**

Now suppose we have two permutations \(\pi\) and \(\sigma\) and we want to compute \(\pi \circ \sigma\). (Remember, this means “the permutation that results when we apply \(\sigma\), then apply \(\pi\”)

Once again cycle notation makes this very easy, and an example will show how this is done.

Let’s use \(\pi = [4 \ 1 \ 5 \ 2 \ 7 \ 3 \ 6]\) and \(\sigma = [4 \ 7 \ 6 \ 1 \ 5 \ 3 \ 2]\).

In cycle notation, \(\pi = (1,4,2)(3,5,6,7)\) and \(\sigma = (1,4)(2,7)(3,6)(5)\) - you should check this.

We can build the cycle notation for \(\pi \circ \sigma\) as follows:

Start with 1. Apply \(\sigma\) to it, giving 4 (that is to say \(\sigma(1) = 4\)). Then apply \(\pi\) to that 4, giving 2 (that is, \(\pi(4) = 2\)). So in \(\pi \circ \sigma\), we see that 1 → 2 (that is, \((\pi \circ \sigma)(1) = \pi(\sigma(1)) = \pi(4) = 2\)). Now start with 2. \(\sigma\) takes 2 to 7, and \(\pi\) takes 7 to 6.

So \((\pi \circ \sigma)(2) = 6\). Now start with 6. \(\sigma\) takes 6 to 3 and \(\pi\) takes 3 to 5, so \((\pi \circ \sigma)(6) = 5\).

Now start with 5. Following the same steps we see that \((\pi \circ \sigma)(5) = 7\). Then we discover that \((\pi \circ \sigma)(7) = 1\) So we have discovered that in \(\pi \circ \sigma\), \(1 \rightarrow 2 \rightarrow 6 \rightarrow 5 \rightarrow 7 \rightarrow 1\) is a cycle, and in cycle notation this is \((1,2,6,5,7)\)
We still haven’t dealt with 3 and 4. It doesn’t matter which we start with, so let’s start with 4. $\sigma(4) = 1$ and $\pi(1) = 4$, so $(\pi \circ \sigma)(4) = 4$. In cycle notation this is just \((4)\)

You can verify for yourself that $(\pi \circ \sigma)(3) = 3$, giving (3) as the last cycle in $\pi \circ \sigma$

Putting all the cycles together, we get $\pi \circ \sigma = (1,2,6,5,7)(4)(3)$

You should check this too!

**Transpositions**

A **transposition** is a permutation that, when written in cycle notation, has exactly one cycle of length 2, and $n-2$ cycles of length 1.

For example $\pi = (1)(2)(3,7)(4)(5)(6)(8)$ is a transposition.

Note that the effect of a transposition is to simply swap the positions of two numbers and leave all the others where they are. Using $\pi$ from the example just given, $\pi(3) = 7$ and $\pi(7) = 3$ and $\forall$ other $x$, $\pi(x) = x$

When we write the cycle notation version of a transposition, we just write the non-trivial cycle .... so for the transposition $\pi$ shown above, we just write $\pi = (3,4)$. (Obviously we need to know the value of $n$, so that we can figure out the trivial cycles that have been omitted.)

**Decomposing a Permutation into the Composition of a Sequence of Transpositions**

This is pretty much our last topic in this introduction to permutations. I’m going to introduce a technique that creates yet another representation of a permutation – you may feel that we already have more than we need, but I promise you they are all useful in different scenarios.

I’m not going to prove that this technique always works (I hope you will trust me for now, and perhaps try to prove it on your own when you have a few spare minutes). When you see how it works, you will probably see how it could be proved.

Let’s consider a permutation $\pi = (1,3,2,4)$ (this is cycle notation, so we know $\pi \in S_4$)

Now consider the permutation $\sigma = (1,4) \circ (1,2) \circ (1,3)$

What the heck is that? Well, each bracketed pair is a transposition, and we are composing them. Since each transposition is a permutation, composing them together gives a new
permuation. Note that each transposition begins with the first element in the cycle notation for \(\pi\), and the second elements in the transpositions are the rest of the cycle, in reverse order.

Now I claim that \(\sigma = \pi\)

Let’s just check it out. To make this less confusing I need to change notation a bit. Up to this point we have used parentheses “(“ and “)“ around the argument to a function, as in \(\pi(1) = 3\). Unfortunately that makes it very hard to tell the difference between \((x)\) as an argument to a function, and \((x)\) as a cycle of length 1 in cycle notation. So for this demonstration, I will use square brackets “[“ and “]” around the argument to a function. So I will write \(\pi[1] = 3\)

Remember that if a transposition is applied to a number which is not one of the two in the transposition, it leaves it alone.

\[
\begin{align*}
\pi[1] &= 3 \quad \ldots \quad \sigma[1] &= ((1, 4) \circ (1, 2) \circ (1, 3))[1] \\
&= ((1, 4) \circ (1, 2))[3] \\
&= (1, 4)[3] \\
&= 3 \\
\pi[2] &= 4 \quad \ldots \quad \sigma[2] &= ((1, 4) \circ (1, 2) \circ (1, 3))[2] \\
&= ((1, 4) \circ (1, 2))[2] \\
&= (1, 4)[1] \\
&= 4 \\
\pi[3] &= 2 \quad \ldots \quad \sigma[3] &= ((1, 4) \circ (1, 2) \circ (1, 3))[3] \\
&= ((1, 4) \circ (1, 2))[1] \\
&= (1, 4)[2] \\
&= 2 \\
\pi[4] &= 1 \quad \ldots \quad \sigma[4] &= ((1, 4) \circ (1, 2) \circ (1, 3))[4] \\
&= ((1, 4) \circ (1, 2))[4] \\
&= (1, 4)[4] \\
&= 1
\end{align*}
\]

So by examining all cases, we see \(\sigma = \pi\)

Now if \(\pi\) has more cycles, they operate completely independently of each other so we can
decompose each of them into transpositions, and then just compose the results together.

For example, if $\pi = (1, 6, 5)(2, 4, 3)$ then

$$\pi = (1, 5) \circ (1, 6) \circ (2, 3) \circ (2, 4)$$

For most of the representations of permutations that we have seen, the representation of each permutation is unique (within some constraints, at least). But for the transposition decomposition of a permutation, this is not true. For many permutations, we can decompose them into completely different sequences of transpositions – in fact the number of transpositions in the decomposition is not even fixed. We may have a permutation that can be decomposed into 4 transpositions, and also decomposed into 6 transpositions.

But something will be consistent: for a given permutation $\pi$, either all transposition decompositions of $\pi$ will have an even number of transpositions, or all transposition decompositions of $\pi$ will have an odd number of transpositions.

This lets us partition $S_n$ into two subsets: the set of even permutations (the ones whose transposition decompositions always have an even number of transpositions in them) and the set of odd permutations (defined analogously).

And that is where we will stop for now. A whole lot of definitions, a few methods for manipulation, some interesting results, but not much in the way of applications. So I wouldn't be surprised if you are wondering why we are teaching you all of this.

It turns out that permutations are important right across the spectrum of computing. They are of great theoretical importance in the study of efficient algorithms, and they are of practical importance in such disparate areas as error-correcting codes, secure cryptosystems, and the design of unbiased statistical experiments. But all those topics must wait for another day.