**PROFESSOR:** An elementary idea that gets you a long way in counting things is this idea of counting with bijections, which is counting one thing by counting another. And we can illustrate that by example. Let's begin with looking at some stuff that is easy to count using just the simple sum and product rules.

So suppose that I'm trying to count passwords. This is a contrived, over-simplified example, but it gives you the idea. And this is what I mean by a password.

A password is a sequence of characters that are either letters or digits subject to the constraints that they are supposed to be between six and eight characters long. They're supposed to start with a letter, and they're case sensitive. So you can tell the difference between uppercase and lowercase letters.

So let's define the set L of all the letters-- uppercase and lowercase together. And let D be the set of digits from 0 through 9. Then we said that passwords are supposed to be between six and eight words long, but it's a little bit easier actually to just use links as a parameter. So let's think about words of length n that satisfy the password conditions. So Pn is going to be the length n words starting with a letter, which is one of the password constraints.

So we can express that has a length n word can be broken up into the first character, which is an L, paired with the rest of the word-- the remaining n minus 1 characters. And the remaining n minus 1 characters can be either L's or D's. So though length n passwords can be expressed as the product of L with the n-th power of L union D-- that is, L union D cross L union D cross L union D, n minus 1 times.

Well, now we have an easy way to count this, because the size of this product by the product rule is the size of L times the size of L union D to the n minus first power. And of course, L union D, since letters and digits don't overlap by the sum rule, the size of them is just L plus D. And so I get this nice formula that is 52 letters times 52 letters plus 10 digits raised to the n minus first power.

What about the passwords? Well, the passwords were then P6 union P7 union P8. And since words of length six don't overlap with words of length seven or eight, this is a disjoint union. And therefore, the total number of passwords as specified is simply the size of P6 plus the size P7 plus the size of P8.

There's the formula when I plug in. And it turns out to be a good size number, 19 times 10 to the 14th. That's one simple example where I'm translating a spec into because something that I can express easily as a products and disjoint sums of stuff that I already know the size of. Let's just do another example.

Suppose that I want to count the number of 4-digit numbers. So the elements of these 4-digit numbers are 0 through 9-- there are 10 possibilities-- with at least one 7-- the number of 4-digit sequences of digits that have at least one 7 in them. And one way to count is I can make it a sum of different 4-digit numbers containing one 7, depending on where the first 7 is. If there's at least one 7, there's a first 7. That's the well-ordering principle applied.

So if we let x abbreviate any digit-- there are 10 possible values of x-- and o represent any digit other than 7-- so there's nine possible values of o-- then the words that start with 7 can then be followed with any three digits. So 7xxx is one possible pattern when the first occurrence of 7 is first. Another possible pattern is when you have a digit that's not 7 followed by a 7. This is when 7 occurs second followed by anything at all. Likewise, here 7 occurs third, and here, 7 occurs forth.

Now, these individual patterns are easy enough to count using the product rule, because here, I have to count how many triples of any digits are there. Well, there's 10 digits, so it's 10 cubed. Here, how many sequences of where the first choice is 9 and the second two choices are 10. And it's 9 times 10 squared. Here, it's 9 squared times 10. And here it's 9 cubed.

These are disjoint, because they're distinguished by where the first 7 occurs. And so I just add them up. And I get this number. It's not especially interesting, but it's 3,439. So that's an exercise in counting something by somewhat ingeniously breaking it up into a sum of disjoint things that are themselves easier to count.

There's another way that's another standard trick that comes up in combinatorics of how do you count the sequence of 4-digit numbers with at least one 7, by counting the complement. Count the numbers of 4-digit numbers that don't have any 7's and simply subtract that number, the number of 4-digit numbers with no 7's, from the total number of 4-digit numbers. And that's going to be the numbers that are left over that have one 7.

Now, the number of 4-digit numbers is easy to count. And it will turn out that the number of 4-digit numbers with no 7's is also really easy to count, because the number of 4-digit numbers

is 10 to the fourth and the number of 4-digit numbers with no 7's, there's nine possible choices for each of the remaining digits. So it's just the digits 0 through 9, leaving out 7, to the fourth power, or 9 to the fourth. And you can double check that 10 to the fourth minus 9 to the fourth is 3,4,39.

So now, with that practice using the basic sum and product rules, we can start applying and thinking about the bijection rule. So the bijection rule simply says that if I have a bijection between two sets A and B, then they have the same size, at least assuming that they are finite sets. And the only kind of things we're counting are finite sets.

Let's use an example of that, where I'm going to count the number of subsets of a set A by finding a bijection between the subsets of a set A and something that I do know how to count. In fact, we've already counted them, the binary strings of a given length. What's the bijection?

Well, suppose that A is a set of n elements, call them a1 through an. And I have some arbitrary subset of A. Say, it's got a1, and it doesn't have a2, and it has a3, and it has a4, and it doesn't have a5. And then it's got some selection of the other numbers. And it turns out it has a n in it.

Well, if I think of a subset laid out this way up against the corresponding elements in A, I can code this in an obvious way by putting a 1 where the element is in the subset and a 0 where the element is not in the subset. In effect, this is the so-called characteristic function of the subset where 1 means that that index element-- a 1 in the i-th position means that ai is there. And a 0 in the i-th position means that ai is not there.

So the second coordinate here is a 0. That means a2 is not there. And this is easily seen to be a bijection. That is, given the string, you could figure out what the subset is. Given the subset, you can figure out what the unique string is.

So we have a bijection. And what we conclude then is that the number of n-bit strings is equal to the size of power set of A. It's equal to the number of subsets of A. And of course, we know how to count the number of n-bit strings. It's 2 to the n.

So what we just figured out is, if I have a set of size n, it's got 2 to the n subsets. And a slick way to say that without mentioning n is that the size of the power set of A is simply 2 the size of A.

One more example of bijection counting that is kind of fun and interesting will illustrate the fact

that we learn something by finding a bijection, even if we don't know how to count either one yet. So what I'm interested in is, suppose I have a situation where there are five kinds of doughnuts-- five different flavors of doughnuts. And I want to sort of select a dozen. Now, I want to know how many selections there are.

So for example-- these little O's represent doughnuts-- I might choose a selection of a dozen by choosing two chocolate and no lemon-- I don't like those so much-- and six sugars and two glazed and two plain. So there are 12 doughnuts here using four out of the five possible flavors of doughnuts. This is what I'll call a selection of a doughnut. And I'd like to know how many such selections of doughnuts are there.

Well, let that be the set A, the set of all these different ways of selecting 12 doughnuts when there are five flavors of doughnuts available. Well, this is, again, an obvious correspondence between the set A of doughnut selections and the set B of 0's and 1's of length 16 that contain four 1's. What's the correspondence? Well, here's my doughnut selection.

And of course, the reason why I use those O's for doughnuts is that they also correspond to 0's. I can just put in 1's as delimiters between the groups of flavors. So after the chocolate doughnuts, I put a 1. And then after the lemon doughnuts, that happen to be none, I put another 1. And then after the six sugar doughnuts, I put a 1.

And then I kind of consolidate and I extract from the doughnut selection this 16-bit word with 12 0's corresponding to 12 doughnuts and four 1's corresponding to breaking up those groups of 0's into five categories, five slots, corresponding to the number of doughnuts of each flavor. So the general bijection, of course, is that if I have a selection of c chocolate doughnuts, l lemon doughnuts, s sugar doughnuts, g glazed, and p plain of any number really, a selection of doughnuts with this number of chocolates, lemons, glazed, plain corresponds to a binary word with c plus l plus s plus g plus p 0's and four 1's.

And so what we can say is that the set of 16-digit words with four 1's is exactly the same size as the number of doughnut selections, even though at this moment we don't know how to count either one. We will see in the next lecture an easy way to count the number of those 16-bit words with four 1's. But for now, our conclusion from bijection counting is that these two sets are the same size, even though I haven't counted yet either one.