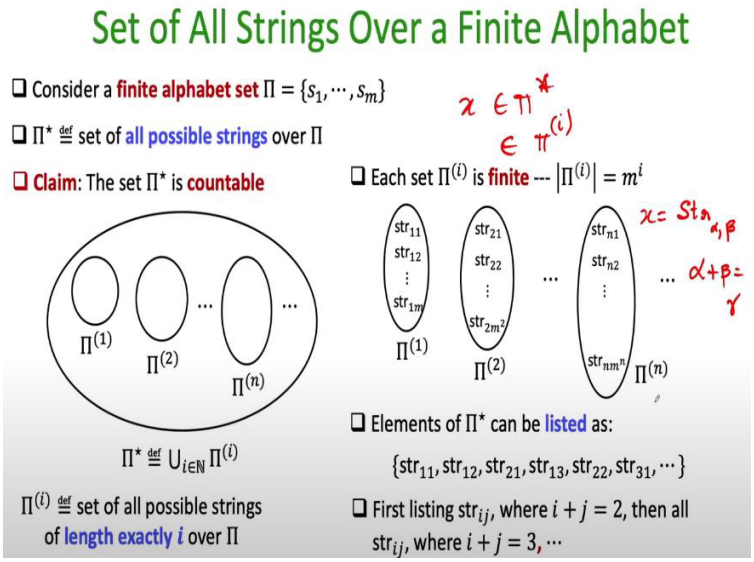


Now what we now going to prove is that the set of all strings over a finite alphabet is also countable. So what do I mean by that is just few slides back I took a binary alphabet which has only 2 symbols 0 and 1. And I proved that the set Π* which is the set of all possible strings of finite length which are binary is countable. Now I am generalizing this result to a bigger alphabet which may have more than 2 symbols or 2 characters.

So I am assuming that I have an alphabet Π consisting of m number of characters s_1 to s_m or m number of symbols. And Π* denote the set of all possible strings finite length strings over this alphabet. So my claim is that is that Π* is countable. So again what is Π*, the way we have defined Π* for the case of the binary alphabet we are going to follow the definition: Π* will be the union of the various subsets $\Pi^{(i)}$.

Where $\Pi^{(i)}$ denote the subset of all strings of length exactly i over the alphabet Π. So, for instance if my Π is consisting of alphabets a, b and c, 3 characters. Then $\Pi^{(0)}$ of course will be the empty string, $\Pi^{(1)}$ will have all the strings of length 1. So I will have 3 strings. $\Pi^{(2)}$ will have all possible strings of length 2. So I can have strings like this and so on. So it is easy to see that each subset $\Pi^{(i)}$ is finite because each subset will have m^i number strings.

And the set Π* is the union of all such subsets. So it will have infinite number of elements. But now we want to show a valid sequencing of the elements in the set Π* . So here is how we can list down all the elements of the set Π* without missing any of them. So since the set $\Pi^{(1)}$ is

finite it will have an enumeration of the elements of its set. So let that enumeration be this. So the first string in $\Pi^{(1)}$ is denoted as str_{11}, the second string is denoted by str_{12} and so on.

So in the subscript I have 2 variables. The first index here denotes the subset in which the string belong. And the second subscript denotes the ordering of that element within that subset. In the same way I will have a sequencing for the elements in the subset $\Pi^{(2)}$. So you can see here each string the first index is 2 2 2 denoting that each such thing belonging to the second subset and then we have the second level of indexing.

And the second level of indexing is from 1 to m^2 because this because the subset $\Pi^{(2)}$ will have m^2 number of elements. And in the same way if I consider the subset $\Pi^{(n)}$ it will have m^n number of strings and like that. So now what we have to do is we have to come up with a valid mechanism or valid sequencing for listing down the elements of set Π^*. And that I can do by following the sequencing by following this ordering what exactly is this ordering.

The idea is that you first list down all strings of the form str_{ij} where the sum of the indices i and j is 2. Why we are starting with the summation of indices being 2, because you can see that my first string here the least indexing I can have here is str_{11} and the summation of the indices will be 1 + 1 namely 2. So I will start with str_{11}. Then I will list out all the elements; all the strings where the summation of the indices will be 3.

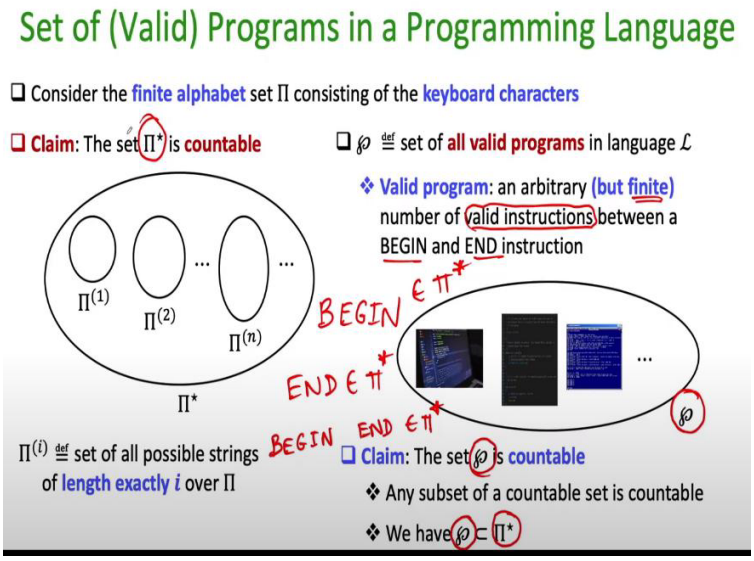
So that is why str_{12} and str_{21} because the summation of these 2 indices will be 3 and the summation of these 2 indices also will be 3. Now if you have many strings where the summation of their indices are the same value then you will follow the ordering among the subsets itself. So since str_{12} appears in the subset $\Pi^{(1)}$ and the str_{21} follows comes in as the subset $\Pi^{(2)}$ and $\Pi^{(1)}$ is appearing before $\Pi^{(2)}$ that is why I have listed down str_{12}.

And then I have listed on str_{21}. Then I will list down all strings such that the summation of the indices is 4. And again you can see here there are 3 strings. So what I have done is I have first taken the string from the set $\Pi^{(1)}$ and then I have taken the string from the set $\Pi^{(2)}$ and then I have taken the string from the set $\Pi^{(3)}$ and so on. So you can see here if I follow this ordering this is a well-defined ordering.

Why it is well defined ordering? Because you take any string x belonging to Π* it will belong to some $\Pi^{(i)}$. That means it will be appearing somewhere in the listing of the elements of $\Pi^{(i)}$ and it will have a form $str_{\alpha,\beta}$. So x will be of the form say $str_{\alpha,\beta}$. And α+β will be some integer. So say α + β is say γ. So once I have listed down all the strings where the summation of its indices is γ-1.

Next I will be listing down all the elements all the strings with such that the summation of the indices is γ and during that process x will be appearing in my sequence. So I will not be missing x and I know definitely we will not be waiting infinitely for listing down the element x. That means we will never get stuck in this process of listing down or enumerating down the elements of the set Π* and that is why this is the valid sequence.

(Refer Slide Time: 38:32)



So now based on the previous theory what we can prove here is that the set of programs or set of valid programs in any programming language is also countable. So what do I mean by that, you take Π to be the set of all keyboard characters. It is a finite alphabet because you have only finite number of keyboard characters; even if you take various combination of keyboard characters that will give you a new character.

But even if you take all such combinations the set of all the characters which you can type using the keyboard in a finite alphabet, I am calling it Π. We already proved that Π* is countable if Π is a finite alphabet. We just proved that because $\Pi^{(i)}$ will be the set of all possible strings of length exactly i and we know how to enumerate out all the elements, all the strings of the set Π*.

Now imagine you have a programming language ***L***, it can be C C+ +, java, python any programming language. And let this calligraphic ***P*** denote the set of all valid programs in your programming language. What do I mean by a valid program? I mean to say it has a start instruction or a begin instruction and it has an end instruction. And in between the begin and the end instruction or the start end instruction and you have arbitrary number of syntactically correct instructions in that programming language.

Valid instructions in the sense when you compiled the program you do not get any error you get some output. How many instructions you can have between the begin and end well that can be arbitrary large but it will be finite. It would not be the case that you have infinite number of steps between the begin and the end instruction. Why that is the case because if you have infinite number of instruction between the begin and end instruction how can your program be valid.

How can your program will give you some output because to get the output from your program you need to reach that end instruction you compiler need to reach the end instruction. That means after parsing all the steps between your begin and end instruction the program has compiled and given you an output. And that is possible only if your number of instructions between the begin and end instruction is a finite quantity.

That means the number of steps is some natural number positive number. So this is my set ***P*** you can imagine it as many programs but the claim is that set ***P*** is countable even though the number of programs is infinite. Because you can keep on inserting, you can keep on taking existing programs and keep on increasing the size of the program by inserting a new valid instructions in the existing valid programs. That way you can keep on creating new programs, this process will never stop.

You cannot say that after this program I cannot find a new program or new valid program. There is no end point here you can always keep on coming up with new programs based on existing program. The simple thing will be just take any existing valid program and just before the end instruction insert a new valid instruction, that will give you a new program which is different from the previous program.

And that is why this set ***P*** which is the set of all valid programs in your programming language is an infinite set it is not a collection of finite number of programs. But the claim is that even though if you have infinite number of programs in your programing language that set is countable. We can list down or we can come up with an enumeration of all valid programs in your programming language. And why that is the case because we just proved that any subset of a countable set is countable.

And what exactly is the set of all valid programs in your programming language well? that is a strict subset or a proper subset of the set Π*. Why? Because I am just considering only valid programs I am not considering invalid program. My set ***P*** has only those programs which will compile and will give me some output. I am not considering programs of the form which has only a begin instruction that is all.

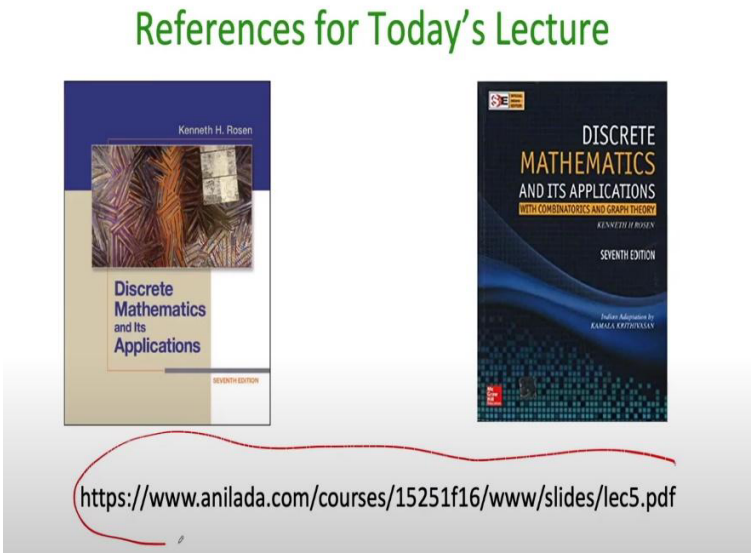
That is also string over the set Π* the string belongs to Π*. But this is not a valid program because it has no end instruction. In the same way the set in the string end also belongs to Π* but it cannot be considered as a valid program. But if you consider the string begin followed by end then that is also string belonging to Π*. But that is a valid program because you have a begin instruction and the instruction and in between you do not have anything but that is fine, this is a valid program.

So that is why the set ***P*** will have only a subset of strings from the Π* because Π* will have all the things that you have in the set ***P*** plus invalid programs as well because Π* just talks about strings over the set Π whether the string is a valid program in your programming language or not that is not necessary here. That is why the set ***P*** is the strict subset of Π*. And since we know that Π* is countable that means we know how to list down the elements of the set Π*.

Using that process we can also come up with the process of listing down all the valid programs in your programming language as well. So that proves a very interesting result. What we have proved is that even though the number of programs the number of valid programs in any programming language is infinite, we can always list down those valid programs so that we are never going to miss any program of any valid program in your programming language in that sequencing.

And it will not be an infinite process in the sense you would not be stuck for ever to find out the position of any valid program in the programming language in that sequencing.

(Refer Slide Time: 45:34)



So that brings me to the end of this lecture. These are the reference for today's lecture and again I followed some of the examples from this article in the current lecture thank you.