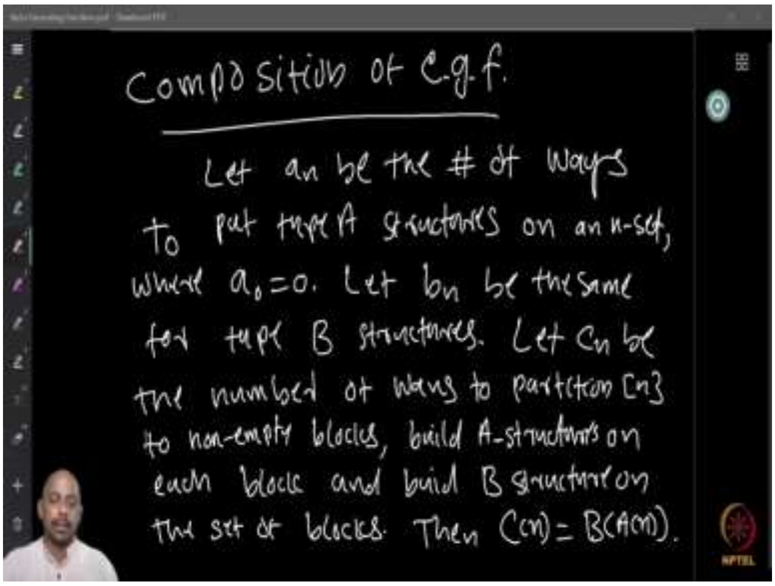
Combinatorics

Doctor Narayanan N
Department of Mathematics
Indian Institute of Technology Madras

Composition of EGF

(Refer Slide Time: 00:16)



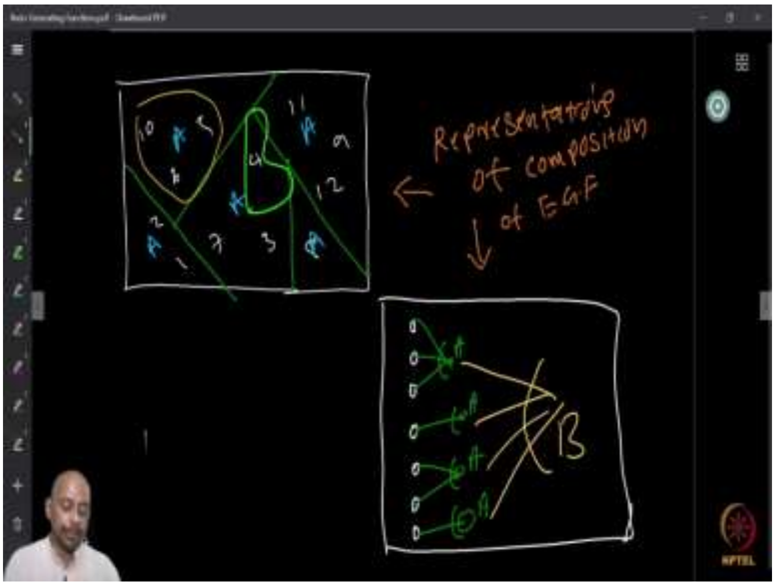
Hello, and welcome back. In the past few lectures we have been discussing generating functions, and in the last lecture we discussed the exponential generating function, and looked at the product of generating functions, which are exponential. Now, we also look at what is the combinatorial interpretation of this product.

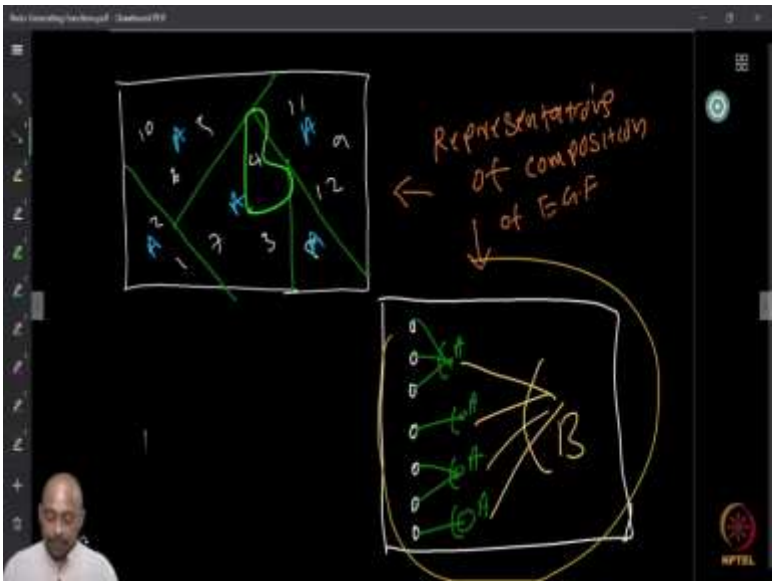
Now, similar to what we did for the ordinary generating function we can also talk about the composition of exponential generating functions. So for this let us start by assuming there are two generating functions, EGFs, where a_n counts the number of ways of making type A structures on an n-element sets, where we assume $a_0 = 0$, just like we did in the previous case.

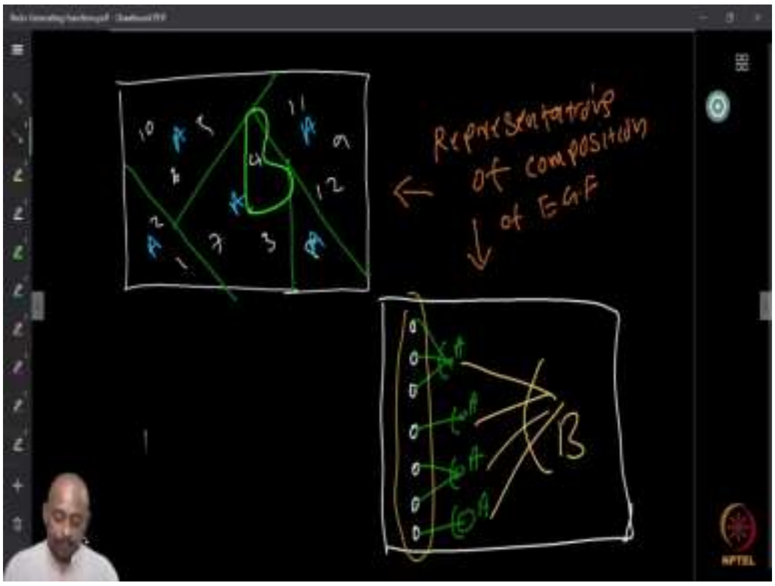
And let us also assume that b_n is the number of ways to put B type structures on an n element set. Now once we have these two, so we have A(x) and B(x) corresponding to the generating functions, exponential generating functions. So we now let another sequence c_n count the number of ways to first partition the set [n] to non-empty blocks and then build A type structures on each block, and then considering these structures as elements, build B type structures on the set of these blocks.

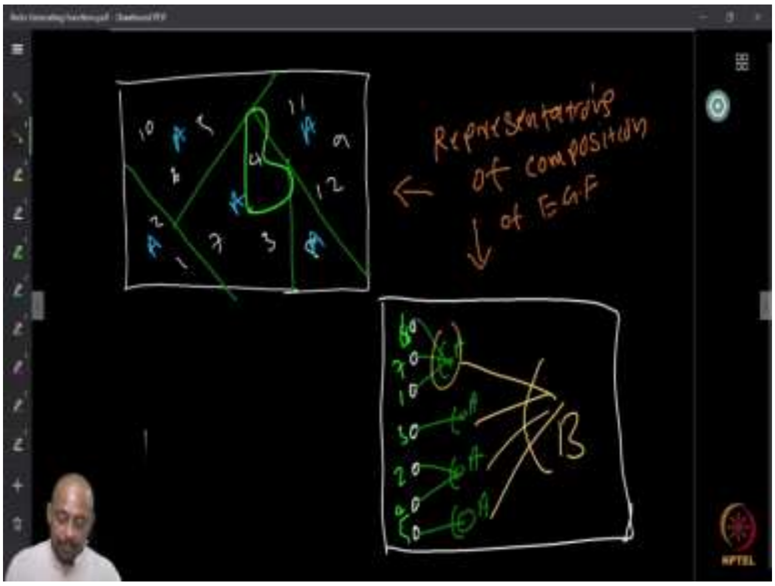
Then the claim is that C(x) = B(A(x)). That is, the exponential generating function which counts the sequence c_n is the generating function of B substituted with A(x) for x. So this is our theorem. We want to prove this theorem. But to make things a little more intuitive let us see what we are actually doing with the structures when we make the C type structures.

(Refer Slide Time: 03:10)









Composition of e.g.f.

Let a_n be the # of ways to put type A structures on an n-set, where $a_0 = 0$. Let b_n be the same for type B structures. Let c_n be the number of ways to partition $[n]$ to non-empty blocks, build A-structures on each block and build B structure on the set of blocks. Then $C(x) = B(A(x))$.

So here, here is some nice pictorial representation of what is actually happening. So in this box we have this set 1, 2, 3, etcetera, 1 to n, let us say, for n is equal to 11 or something. Then what we can say is that we have partitioned the set into different blocks like this.

So we have this, one part is like just including 1 and 2, then you have a part with 3, 6 and 10, another part with, let us say 3, 4, 7, then 8 is by itself a part, and then 12, 9 and 11, yeah, so up to 11, 12, is another part.

So we have this, we have this partition, so and each part is non-empty. Then what we do we put type A structures on each of these sets. So I can make a type A structure on the two element set. Similarly, on this 3 element set I put an A type structure. So there could be several ways of doing this.

There is precisely a_3 ways of making A type structures on this 3 element sets. That is the definition of the sequence a_n. Similarly, we have a_1 ways of doing this, a_2 ways of doing on this set etcetera.

Now we have the A structures on each of these blocks now. Now consider this as elements and then make a B type structure on this set of A structures. So then what we get is a C type structure.

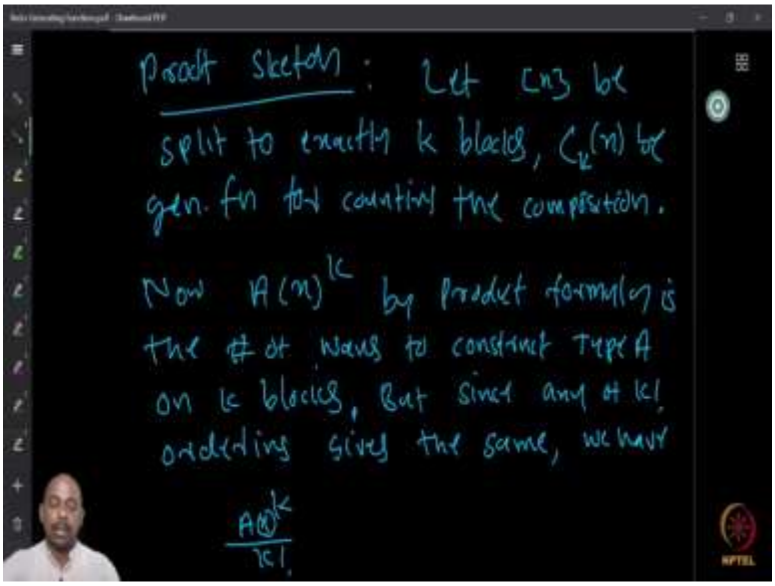
Another way to represent this is in the second figure where this figure, where we have the set, in this case we are not given any names or anything, but just to, we have the set and then we could of course name them if you want. You can make it 1, 3, 7, 8, 2, 4, 5, something like that. Now, I think I missed one thing, 8, it is 6.

So now what we do, we partition it into blocks like so we have 1, 7, 6 is a block, 3 is a block by itself, 2 and 4 is another block and 5 is a block. Now on each of this block I put an A type structure. So this represents that I am making A type structure using all these elements. So that is the meaning of this representation.

And then once we have this, take each of these A structures and then considering them as elements I make a B type structure on the set of all A structures that we have. So this is what we mean by the composition, in the exponential generating function.

So now one can ask, what is the proof that B(A(x)) precisely counts the number of C type structures. So this is something that we want to do.

(Refer Slide Time: 07:18)



So I will give you a sketch of the proof. So like in the previous case, now what is the idea here? So we know that we start with set 1 to n, and then we split it into different blocks. Now let us start with the assumption that, I mean when I split into different blocks, it must be some number. So let us say that there is exactly k blocks. So, k could range of course, from 1 to n itself.

We will fix k for some value and then assume that we have now split into exactly k block, and then see what happens. Now if I do it into exactly k blocks, there could be several ways of doing that itself, and then let us say $C_k(x)$ is the generating function for counting the composition in this particular case when I split it into exactly k blocks.

I do not look at other values of k. Only for this particular value. And then I say that okay, now if I do this, I take the set 1 to n and then partition into exactly k parts, then put A structures on each part, and then put a B structure on the A structures. So if I do that, let us see what happens.

Now, we know that there is exactly k blocks. So each block is basically a subset. So take the set on this block, then we put an A type structure. Now, we are doing this for each of the k blocks. So if you want to talk about the, generating function of making A type structures on each of the blocks independently, then we know that we can obtain this by taking the product of the generating function.

So if for example we said that if I want to make like partition into two parts then put A structure on the first type, and then B structure on the second type, then we said that A(x)B(x) will give the generating function for doing this. So now we are partitioning into k parts, and then putting A structure on each of the parts.

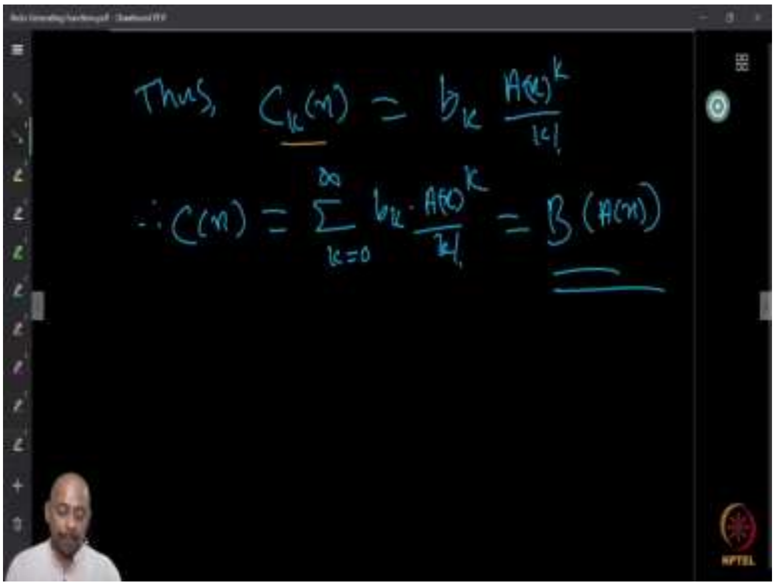
So therefore A(x) .A(x)… A(x), k times will give the generating function for doing this. But there is only a small catch. So the catch is that, see, when we were doing this, the different type of objects A type, B type, C type etcetera, so when I take, when I partition and then I put an A structure on the first, B structure on the second, and then C structure on the third, the ordering of these parts actually matters.

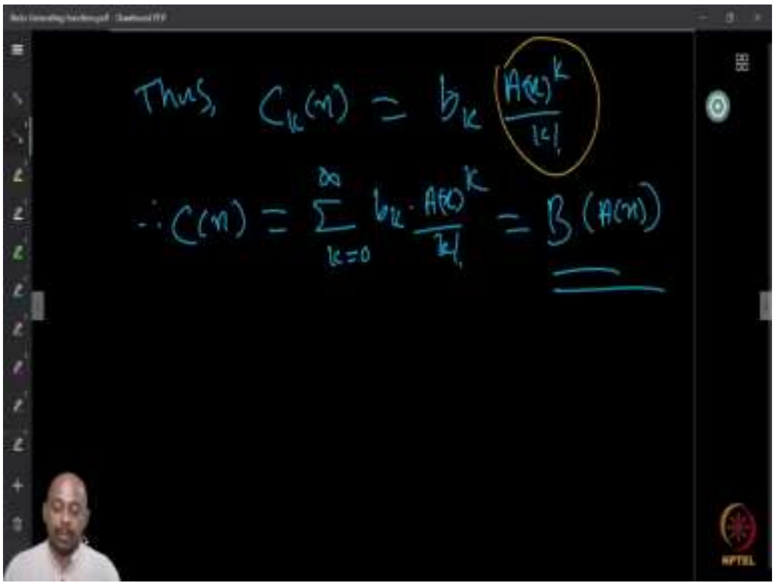
So if I have k parts the ordering of the parts will matter because on the first part I am going to put A structure, second part I am going to put B structure, and the third part I am going to put C structure. But since I am going to put A structure on each of these parts, at the first block I am putting A structure, second block I am putting A structure, third block also I am putting A structure.

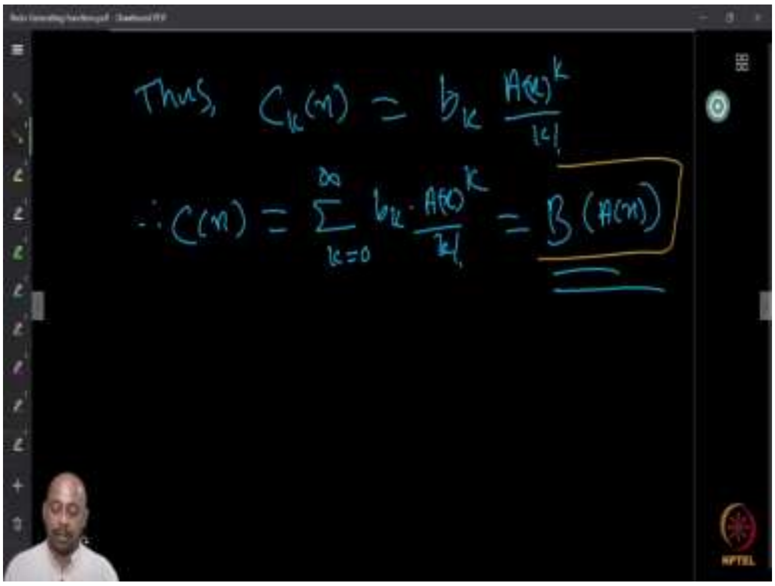
So therefore it does not matter which order I have these blocks. What only matters is that how many blocks are there. So therefore when I look at $A(x)^k$, we are basically considering all the possible permutations of the blocks also. So there is exactly k! permutations and each of them will give the same object because we are putting A structures on them.

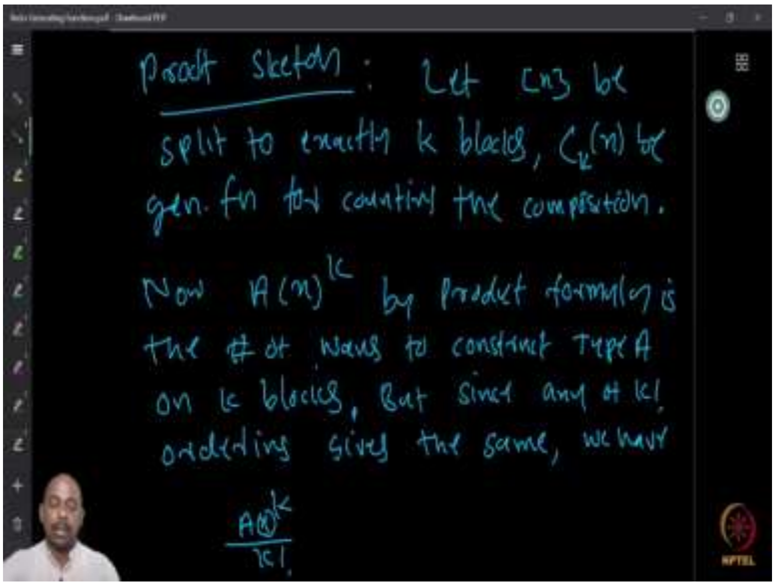
So therefore we have to divide by k! to make sure that we do not over count. So this gives the idea that $\frac{A(x)^k}{k!}$ is the generating function for $C_k(x)$. And whatever is the number that we started with, n elements, you look at the coefficient of x^n in that will give you precisely the number of ways of obtaining C type structures on n-element set by doing this.

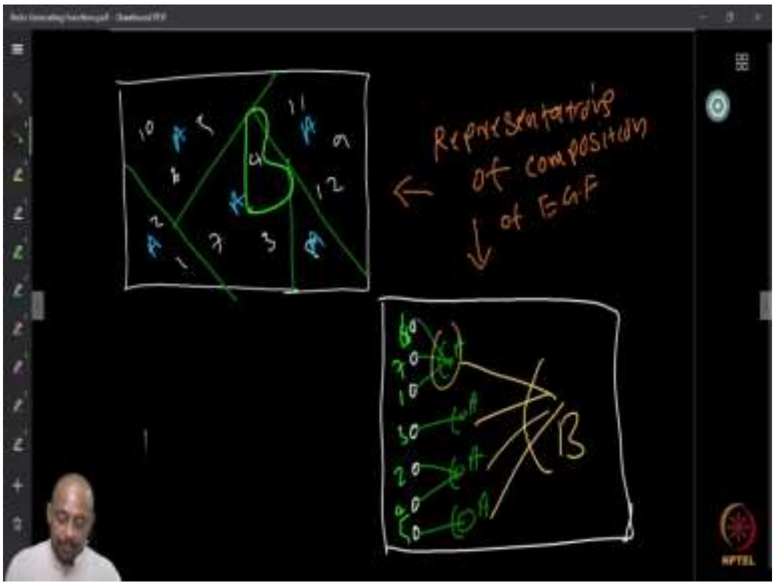
(Refer Slide Time: 11:54)











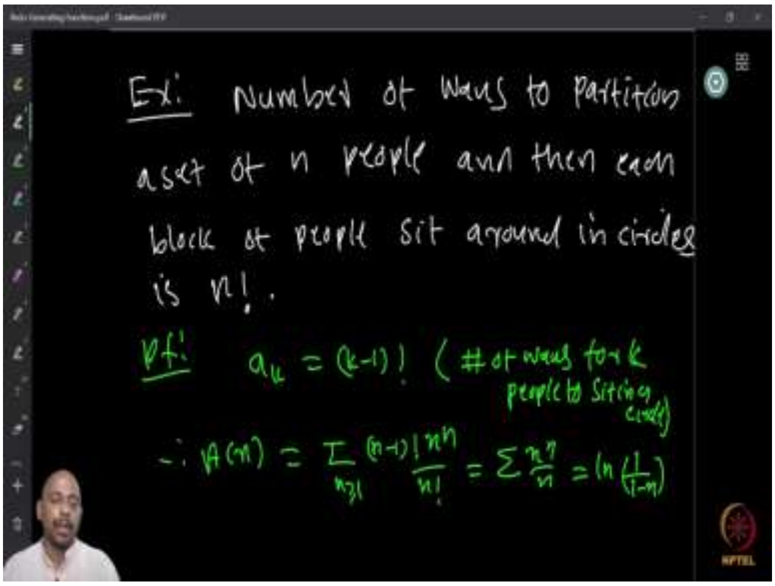
There is another thing that once you make this we have to put B type structures. So it is not $C_k(x)$, it is just to do the A type structures on these blocks.

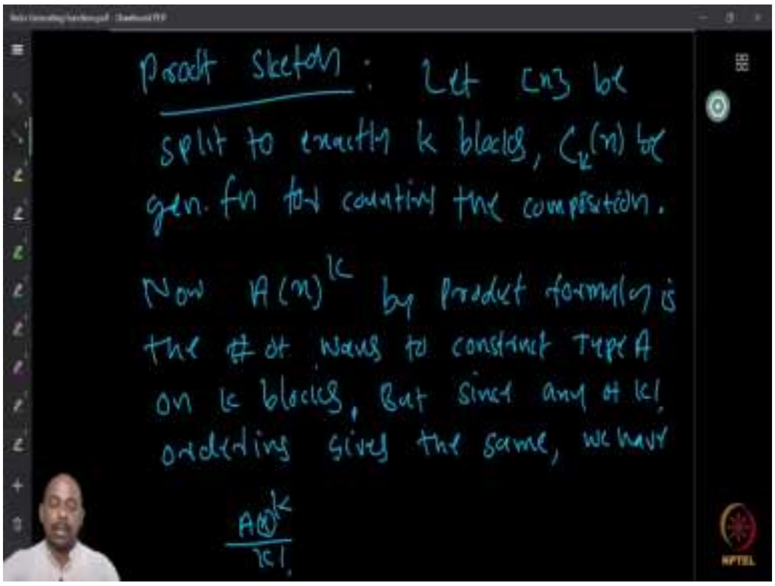
Now, then we have to make B structures on the k A structures that we have obtained. So that we can do in precisely b_k base. So therefore $C_k(x) = b_k \frac{A(x)^k}{k!}$. Because $\frac{A(x)^k}{k!}$ is the number of ways we can put A structures on the k blocks, and b_k is the number of ways to make B structures from the k distinct objects that we have obtained.

Now, if you know, for each k what is $C_k(x)$, then I can get C(x) easily. C(x), the generating function for C is basically the sum of all this, because when we separate it by the cardinality, like of the set in which we are building, our formula itself is the summation over all values of it. So, $C(x) = \sum_{k=0}^{\infty} C_k(x) = \sum_{k=o}^{\infty} \frac{A(x)^k}{k!} b_k = B(A(x))$. So this is the idea of the proof.

So we have shown that if I take a set, partition into non-empty blocks, put the type A structures on the set, then put type B structures on the set of A type structures what we get, if it is counted by the sequence c_n then C(x) is a generating function for c_n. So you basically substitute A(x) into B. So that is the composition of generating functions.

(Refer Slide Time: 14:57)





Now let us see some application for this. Our first example is the following. That we have a number of ways to partition a set of n people. And we want to count the number of ways to partition a set of n people, and then once you partition, we want to put each block of people and ask them to sit around in circles.

And the claim is that, this is precisely n!. We are basically partitioning n persons into blocks and then ask each block of people to sit around in circular table. So you have like circular tables with the chairs around it. Now, how many ways one can do this? So the claim is that this total number is n!.

So let us prove this using the composition of generating functions. So let us denote by a_k as the number of ways the people can sit around in circles. So basically what we were doing in the exponential generating function was that you split into k blocks and then you put type A structures on each of the blocks. So here we are asking the people to sit around in circles, each block of people to sit around in circles.

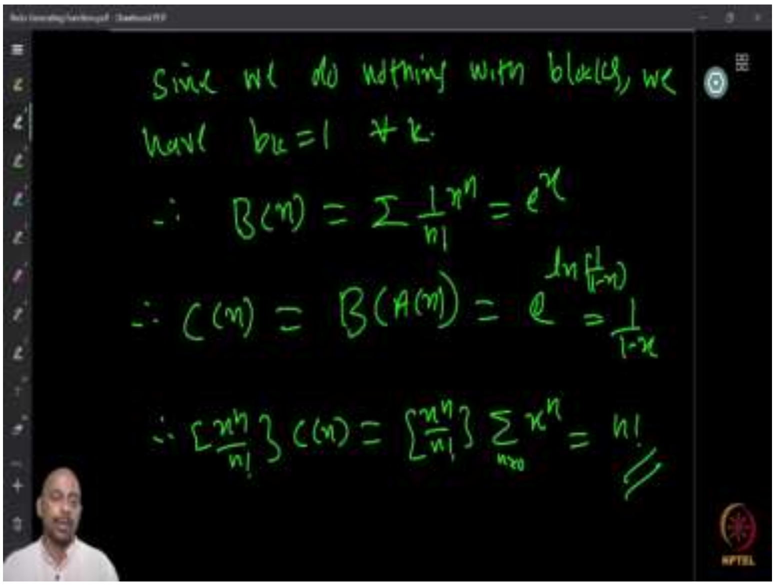
Now, if there are k people in a block, then there is (k-1)! ways to put them in circular way. So, we know that the generating function for people to sit around in circles is

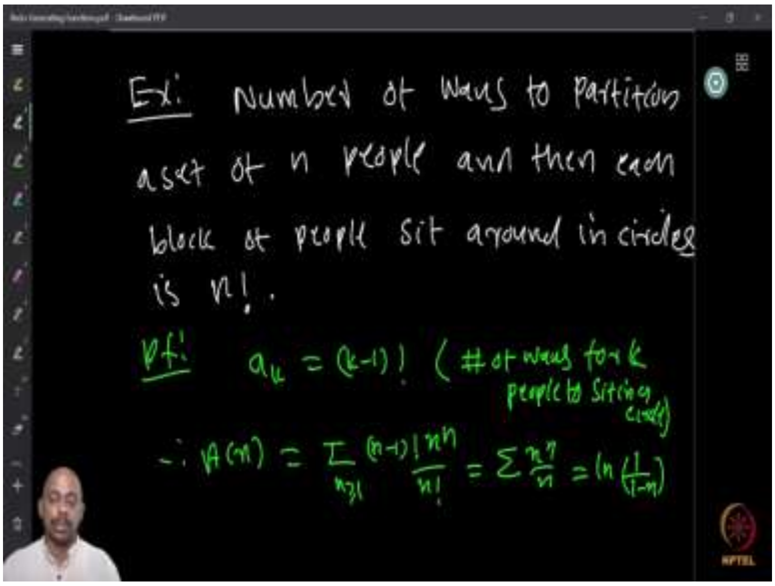
$A(x) = \sum_{n\geq1}\frac{(n-1)!x^n}{n!} = \sum\frac{x^n}{n} = \ln\left(\frac{1}{1-x}\right)$ which is the exponential generating function.

Now what else we do so we partition, and then we are just asking them to sit around, but, then we are not doing anything, we are not making B type structures. So if we were making B type structures then we could use the substitution, but if we are not doing the B type structures what exactly can we do?

So if you think about it, like when we do nothing, that there is precisely one way to do it, do nothing. So we, we partition, make this A structures and then do nothing, which means that we are doing precisely one way. So the coefficient of b_n for any n is precisely 1 because no matter how many blocks are there we have exactly one way of doing nothing.

(Refer Slide Time: 19:05)





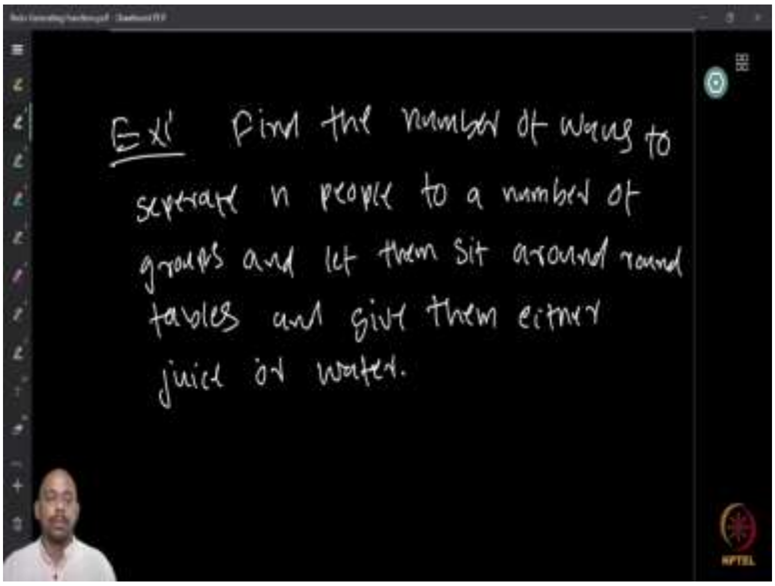
So $B(x) = \sum \frac{x^n}{n!}$ because we are looking at the exponential generating function. What is that? This is e^x. We already saw this several times. So $B(x) = e^x$. Now,

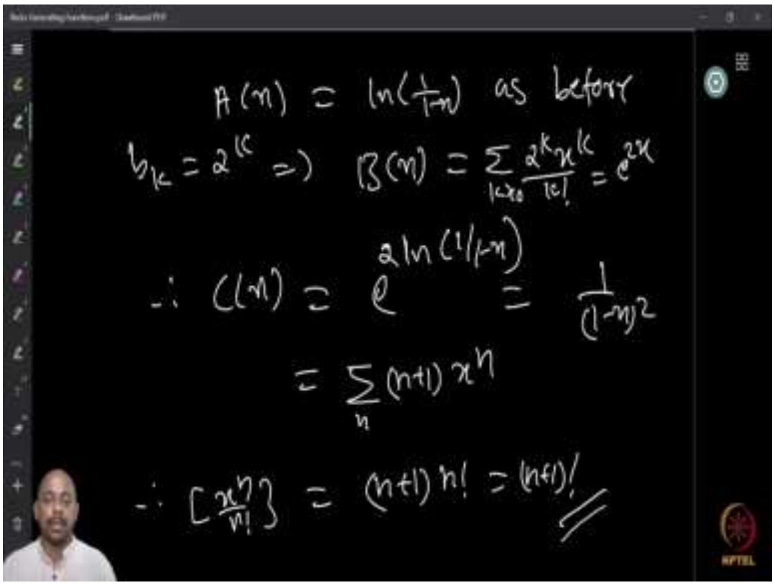
$$C(x) = B\big(A(x)\big) = e^{\ln\left(\frac{1}{1-x}\right)} = \frac{1}{1-x}$$

Now what is the coefficient of $\frac{x^n}{n!}$ in the series expansion of $\frac{1}{1-x}$? We have $\frac{1}{1-x} = \sum x^n$. But then we want to look at the coefficient of $\frac{x^n}{n!}$ in this, which is precisely n! because we want to multiply and divide by n! to make the coefficient to be 1.

And that is what we have. So, n factorial is the number of ways of making the C type structures on n element sets which is basically asking the people to sit around in circle. So this is, the, there is n factorial ways of doing this.

(Refer Slide Time: 20:41)





Now let us look at one more example, which is slightly related. So find the number of ways to separate n people into a number of groups, and then let them sit around in circles, but then after sitting them on tables, we serve them with either juice or water. How many ways of doing this? How do you count this? We can use the, the same idea.

We know that $A(x) = \ln(\frac{1}{1-x})$, as we did before, asking people to sit around in circles after partitioning them. Now what is b_k? So if, if there are k blocks and we are serving either water or juice to the tables there is precisely 2^k ways of doing this. We can decide whether to give water or juice, therefore two choices, and we have 2^k possibilities for the k, we can do that.

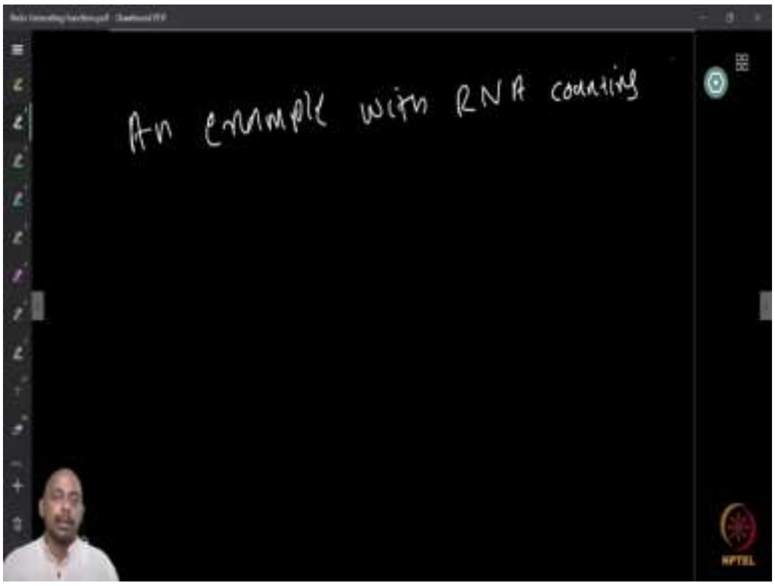
And therefore B(x) is the generating function for the type B structures on this tables is $B(x) = \sum_{k\geq 0} \frac{2^k x^k}{k!} = e^{2x}$ Now we have, $C(x) = e^{2\ln\left(\frac{1}{1-x}\right)} = \frac{1}{(1-x)^2} = \sum_n (n+1)x^n$

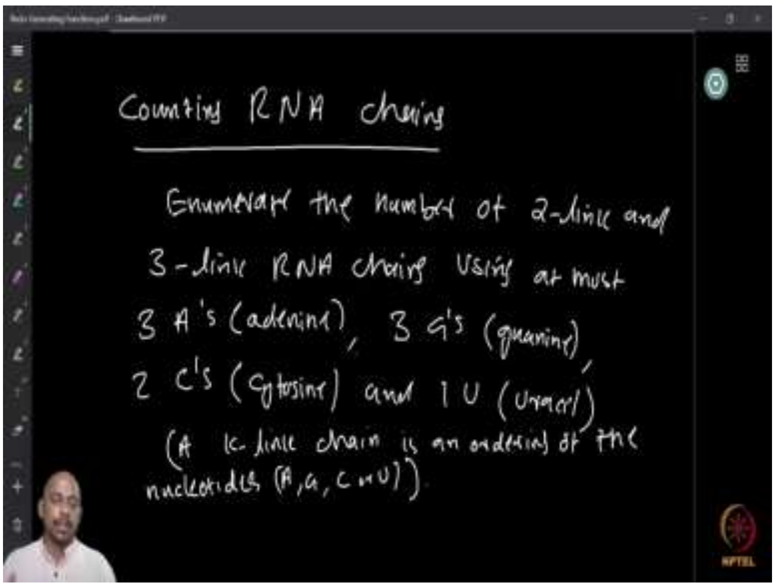
But we want the coefficient of x^n by n!. So therefore we multiply and divide by n!, I get (n+1)!. So there is (n+1)! ways of doing this, and this is kind of surprising because when we do this 2^k is coming, like when we do this serving of juice or water, that somehow we will feel that like, there is the factor of 2 must be coming.

So these are two examples of using the composition, I mean one can have several other examples, but let me now go to some, few other things.

I am going to give an application into counting RNA sequences, and similarly I want to also look at an application of generating functions at the end to wind up this topic, where we will prove Euler's famous pentagonal number theorem. So these two are the applications, and with that we will wind up this part.

(Refer Slide Time: 24:15)



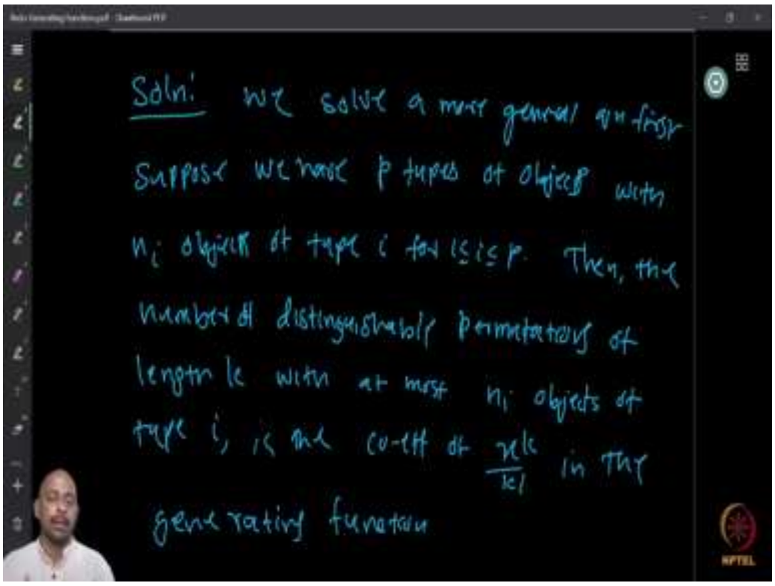


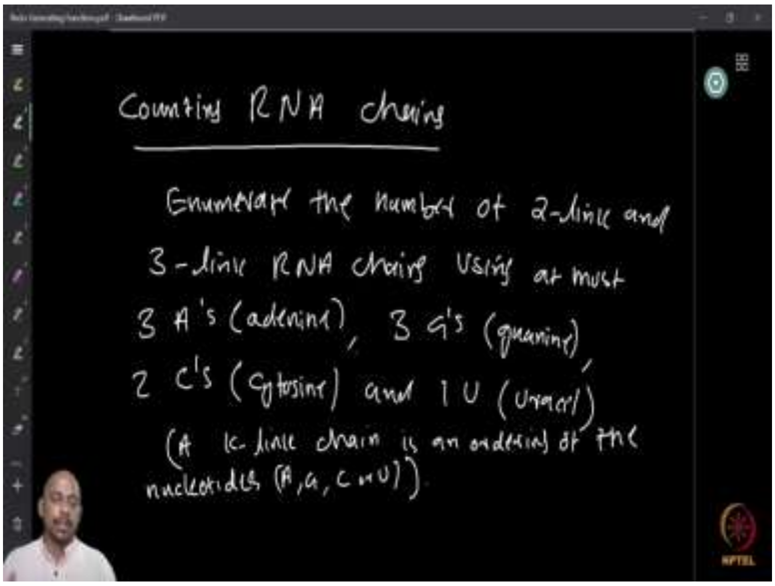
Here is an example with RNA counting. We want to count the number of RNA chains, with some conditions. A k-link RNA chain, is basically an ordering of the nucleotides. So what are the nucleotides? one is the adenine, then you have guanine, then cytosine, and uracil. These are the nucleotides which are present in RNA chains. If you are looking at DNA chain, then U will be replaced by something else. But other things are there. But then there is some differences.

So what we are going to do here is to count the k-link RNA chains. So where I want to specifically count the 2-link and 3-link chains. And basically a chain is an ordering of the letters A, G, C or U where, A, G, C, U representing nucleotides, which are the molecules.

Now, how do we do this? We have the condition that we are allowed to use at most 3 adenine or three guanines, two cytosines and one uracil. These are the only things available, out of which we want to make 2-link or 3-link RNA chains. So how do you do that?

(Refer Slide Time: 25:50)



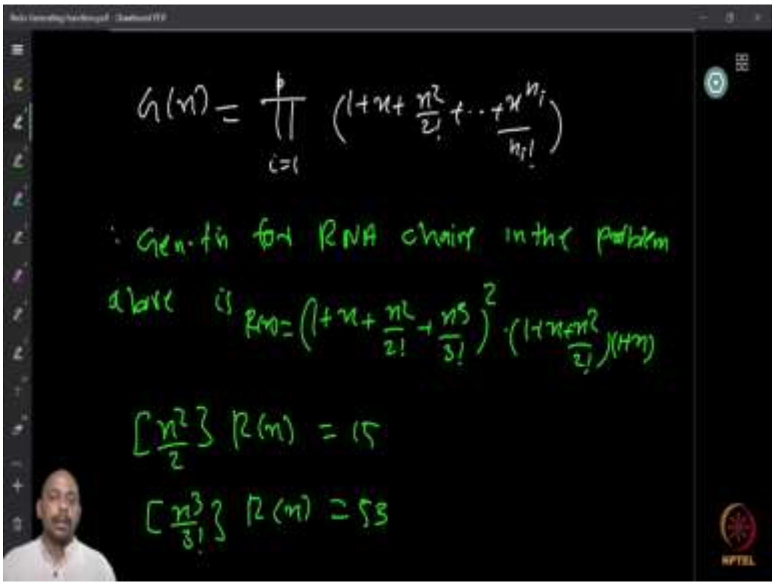


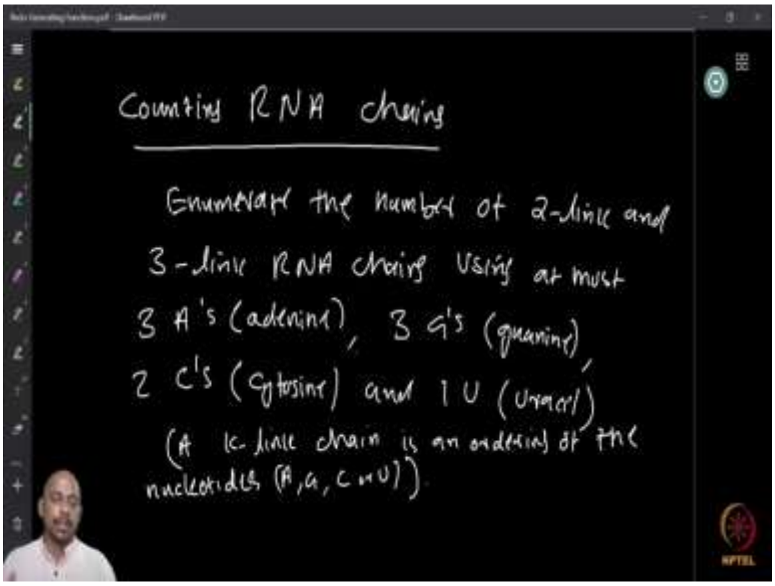
Here is a solution. We are going to prove a little more general question first, and then as a special case we will solve the RNA code. So suppose we have p different types of objects, where for each i, the type i object there is the n_i of them available.

So in this case we had, we had type A objects, there were three of them, type G objects there were three of them, type C objects there were two of them, and type U object there was one. So similarly type i object, there is n_i of them available.

Then, the number of distinguishable permutations of length k with at most n_i objects of type i to draw from is the coefficient of $\frac{x^k}{k!}$ in the following generating function. So can you guess what is the generating function for this? So think about this for some time before proceeding further.

(Refer Slide Time: 27:10)





We have $G(x) = \prod_{i=1}^{p}(1 + x + \frac{x^2}{2!} + \cdots + \frac{x^{n_i}}{n_i!})$

This will tell you that, if I am looking at most n objects, then the coefficient of x^n in this product will tell that, from the first factor which is $1 + x + \frac{x^2}{2!} + \cdots + \frac{x^{n_1}}{n_1!}$ which term I have chosen, that will tell you how many copies of the type 1 object I have chosen.

So if I take $\frac{x^2}{2!}$, that will tell me that I am choosing two copies of the type 1 object. So similarly for each type i, and therefore that is the generating function. And so the generating function for the RNA chains is basically, you substitute for each of the type i objects with the corresponding numbers. So $1+x+\frac{x^2}{2!}+\frac{x^3}{3!}$for the type A and type G objects because both of them have three of them, and therefore it is square of this polynomial.

Then we have $1+x+\frac{x^2}{2!}$ for the type C object. And for the type 1 object there is only one available so either 1 or 0, I can choose, 1+x. And hence you take the product of these polynomials, you get a polynomial: $R(x)=\left(1+x+\frac{x^2}{2!}+\frac{x^3}{3!}\right)^2\left(1+x+\frac{x^2}{2!}\right)(1+x)$.

Look at the coefficient of $\frac{x^2}{2!}$, and one can verify that it is actually 15 in this case. And if you look at the coefficient of $\frac{x^3}{3!}$, you can verify that it is actually 53. And that is precisely the way to count this. One can generalize it to other examples very easily.