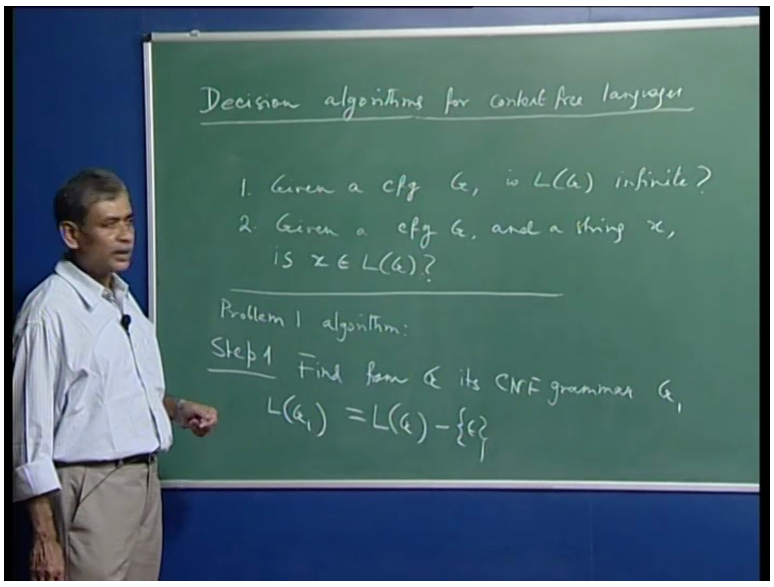
**Theory of Computation**
**Professor Somenath Biswas**
**Department of Computer Science and Engineering**
**Indian Institute of Technology Kanpur**
**Lecture 30**
**More Decision Problems**
**CYK Algorithm for Membership Decision**

Today we will provide algorithms for these two decision problems for context free languages. The first one is that we need to determine given a context free grammar G if the language generated by the grammar infinite or finite? For thisproblem so let us say problem 1 algorithm we will provide a sketch. So you are given the grammar G that is the input. So first what we do is step 1 find from G its Chomsky normal form grammar G 1.

So we know from the definition of Chomsky normal form grammar G that this grammar G 1 that we obtained,the language generated by the Chomsky normal form grammar for the grammar G will generate the language L G except possibly one string and that is the empty string. Because recall that the Chomsky normal form grammar cannot generate the empty string. So clearly G generates an infinite set if and only if G 1 generates an infinite set.

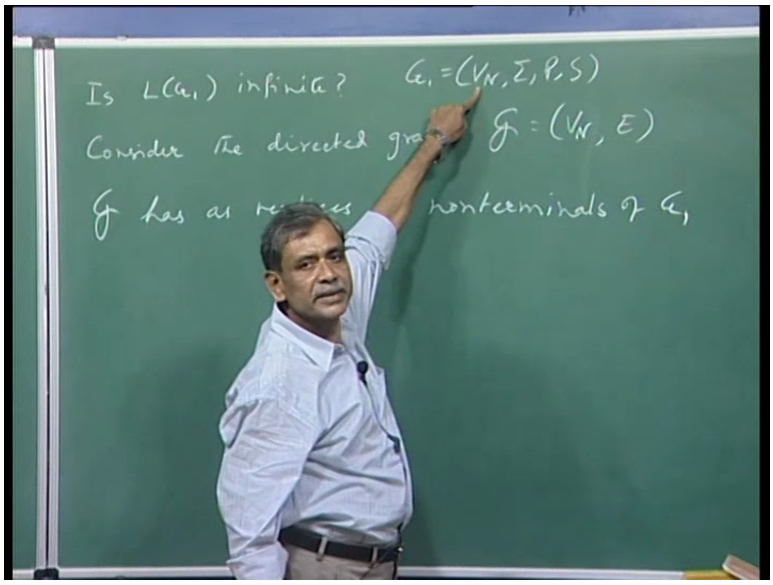
(Refer Slide Time: 02:12)




So equivalent now therefore we will look at, is L G 1 infinite? This is the question we resolved where G 1 is in Chomsky normal form. The way we do it is actually quite straight forward. We create a directed graph for the nonterminals of the grammar G 1. So let me define this. Consider the directed graph. And this sorrythis is thenonterminalset of. So let meclearly say what I mean. So G has this graph.

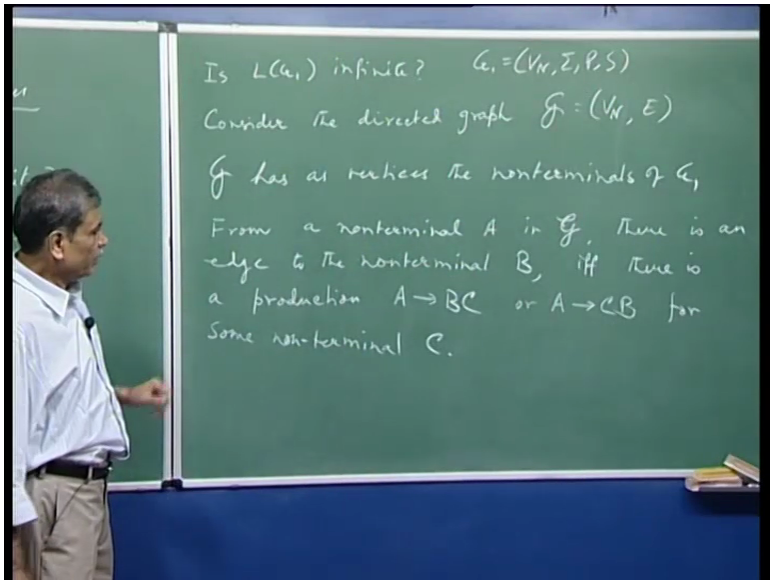
Now this script G is the graph that we are building given the Chomsky normal form grammar G 1. G has as vertices the nonterminal of G 1. So in other words if my G 1 was the set V N sigma P S, the directed graph that we are building that will have as vertices this set of nonterminals. That is what we have written, right?

(Refer Slide Time: 04:25)



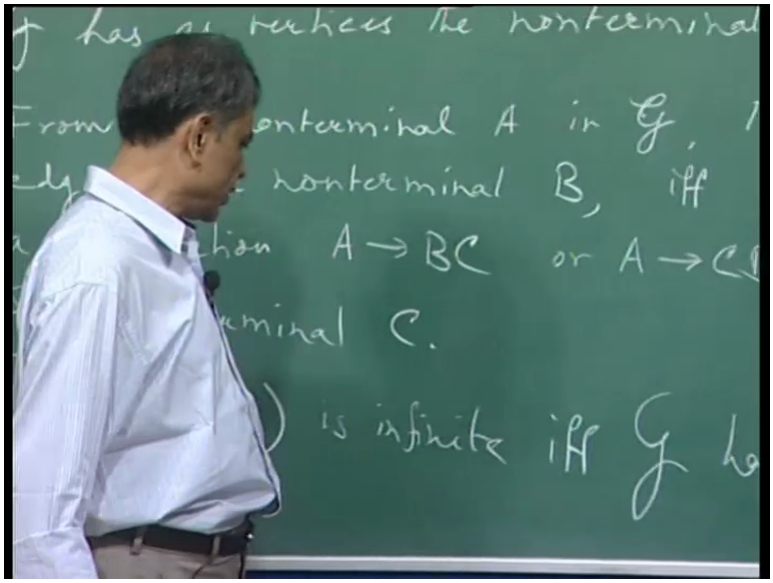

Now from let us say a nonterminal A in G. So(remem) now we can seea nonterminal has kind of two roles, one as a vertex of this graph G of course as a nonterminal of the grammar G 1. So from a nonterminalA in G there is an edge to the nonterminalB in this graph that we are constructing if and only there is a production. A goes toB C or A goes to C B for some nonterminal C, right?

(Refer Slide Time: 06:32)



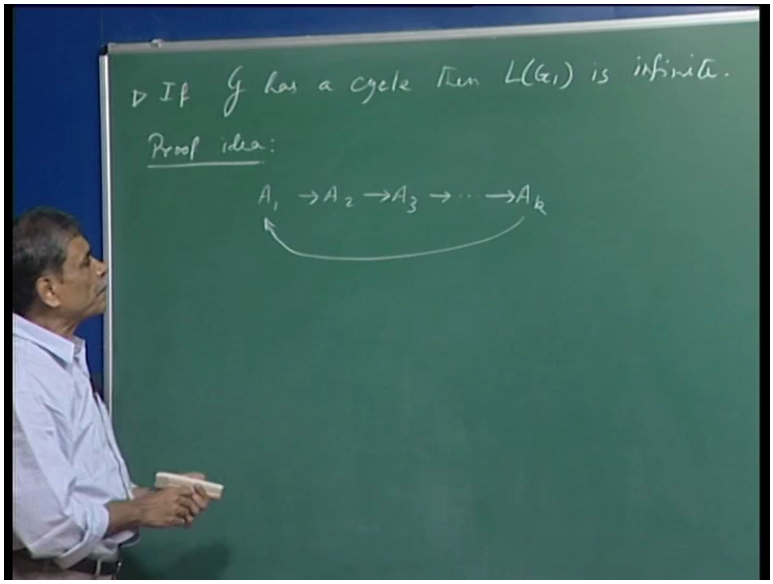
So what we are saying isin this graph that we are building there will be an edge from a nonterminal A to nonterminalB if there is a production in which A occurs on the left hand side and B occurs on the right hand side of that production. Now the claim is L G 1 is infinite if and only if this graph G has a cycle. We need to of course for proving this claim prove that if G has a cycle then L G 1 is infinite and vice versa.

(Refer Slide Time: 07:36)



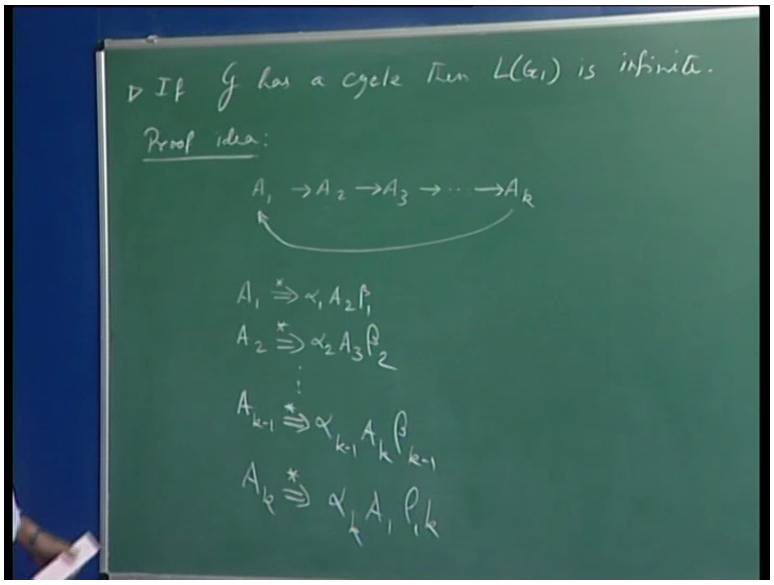

So let us do this one by one. If and the proof idea is supposeone cycle that we have in this graph is of the form the cycle is from A 1 to A 2 to A 3 A k and back to A 1, right? So whatdoes it mean? This means that from A 1 you can derive some string which will have A 2 in it.

(Refer Slide Time: 08:58)



So we can say that you know this first edge in the graph this means that A 1 derives let us say alpha 1 A 2 beta 1 and similarly A 2 derives alpha 2 A 3 beta 2 and so on. A kminus 1 derives alpha A minus 1 A k beta k minus 1. And then because of this edge back we have that A k in turn derives let us say alpha k A 1 and beta k, right?

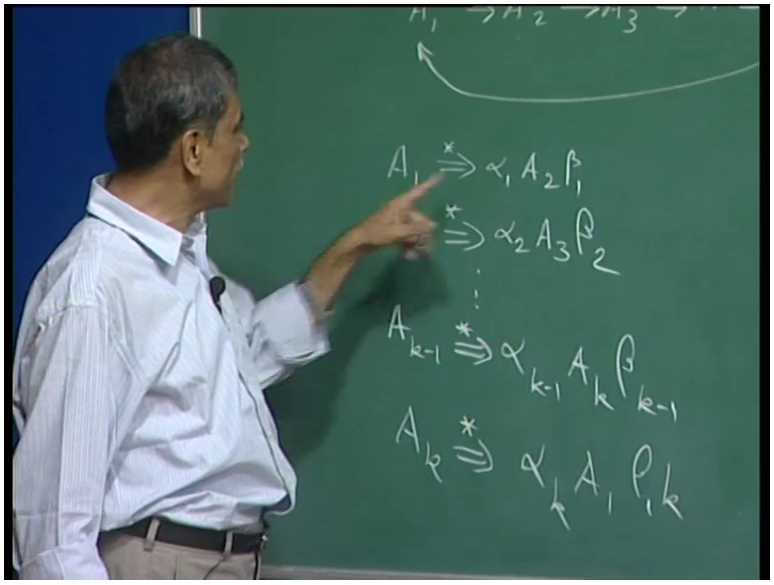
(Refer Slide Time: 10:22)




Now the crucial point is because the grammar G 1 is a Chomsky normal form grammar, this is something you can verifythat it is not possible for each i alpha both, so let me write it this way both alpha i and beta i are empty strings, okay. So in fact it is quite easy to see at least in one for example right in the beginning when we say A 1derives alpha 1 A 2 beta 1 and suppose both alpha 1 and beta 1 they were empty strings.

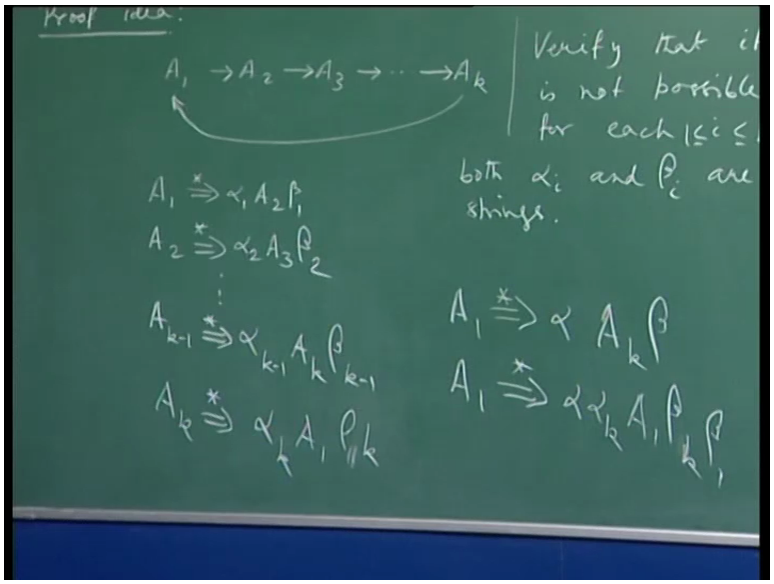
So that would mean what? That A 1 derives A 2. Is that possible? Because that is not possible in Chomsky normal form grammar. Because remember the Chomsky normal form grammar does not have unit productions as well as epsilon productions. So in the absence of unit productions and epsilon productions A 1 derives A 2 is just not possible for two distinct nonterminals A 1 and A 2.

(Refer Slide Time: 12:03)



So basically from this what we get if you see what is happening is that you know from all this we can see A 1 derives some alpha A k beta, you knowin this A 2 I substitute this and so on and then, right? Then this A k I can of course for this A k I substitute this. So what will happen? Alpha alpha k A 1, right? Thenbeta k beta 1, right?

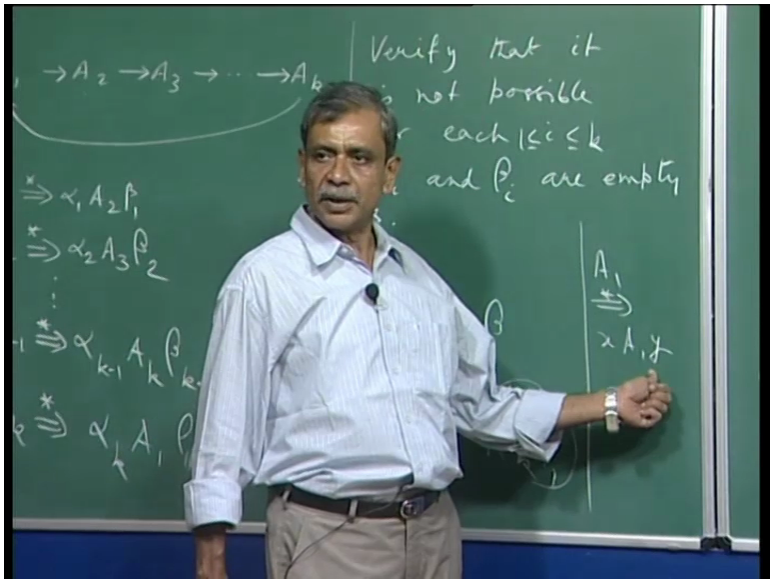
(Refer Slide Time: 13:09)



So as before both of these cannot be empty, right? So that means A 1 derives A 1, a string with A 1 and plus some non-empty things. Now both these things because againthe grammar is a Chomsky normal form it is not possible that both of them finally they derive the empty string. This both of these alpha alpha k together could be a string of terminals andnonterminals. But it is not possible that both these things derive only empty strings.

So in that case what it means is that I will have A 1 deriving something, let us say x A 1 y finally when x and y I should be able to find terminal strings x and y such that A 1 derives x A 1 y where both x and y are not empty, right?

(Refer Slide Time: 14:26)

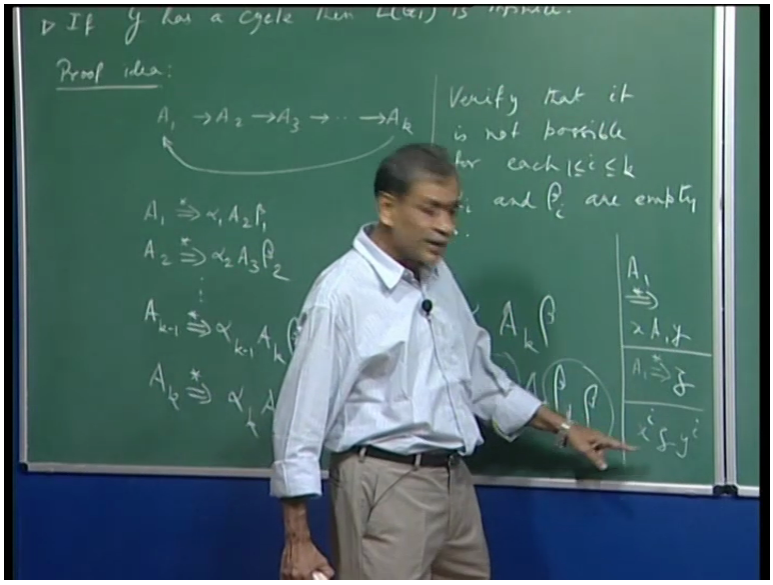



And now also because seethis fact I am getting because of the cycle, right? Nowof course we have another fact about A 1 that A 1 does generate some terminal string which is non-empty. Why? Because A 1 is not a useless symbol. So A 1 has to derive some terminal string because A 1 is a nonterminal of the Chomsky normal form grammar. So now put these two together. What does it mean? Just see right away this is very simple that from A 1 you can get x A 1 y.

Then youcan just keep on doing it as many times. You know this A 1 then you can again replace it by x A 1 y. So you will get x x A 1 y y. Then x x x A 1 y y y. Something in that form of our u v w x y theorem and then finally this A 1 you can rewrite by the terminal string z. And so therefore A 1 will derive what?X i z y i for all values of i greater than equal to 0. And what also we know so we can say A 1 derives this string.

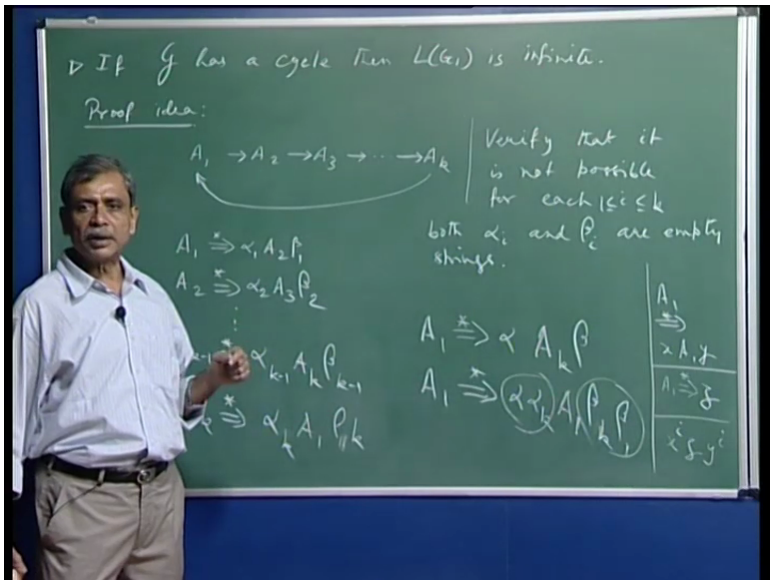
And for two different i's, let us say i and i plus 1, it cannot be that x i y i and x i plus 1 y i plus 1, these two strings are same. Why? Because x and y cannot be both empty together. So therefore it is very clear that in this case this A 1 will generate infinitely many strings.

(Refer Slide Time: 16:32)




And then of course there will be starting from the start symbol S because A 1 is reachable because G 1 is a Chomsky normal form grammar. You will have some string starting from S with A 1 in it. And now this A 1 can derive infinitely many strings so therefore S itself can derive infinitely many strings. The idea is fairly simple and all that depended on is the fact that we have such a cycle and we have crucially used the fact that the grammar G 1 is a Chomsky normal form grammar.

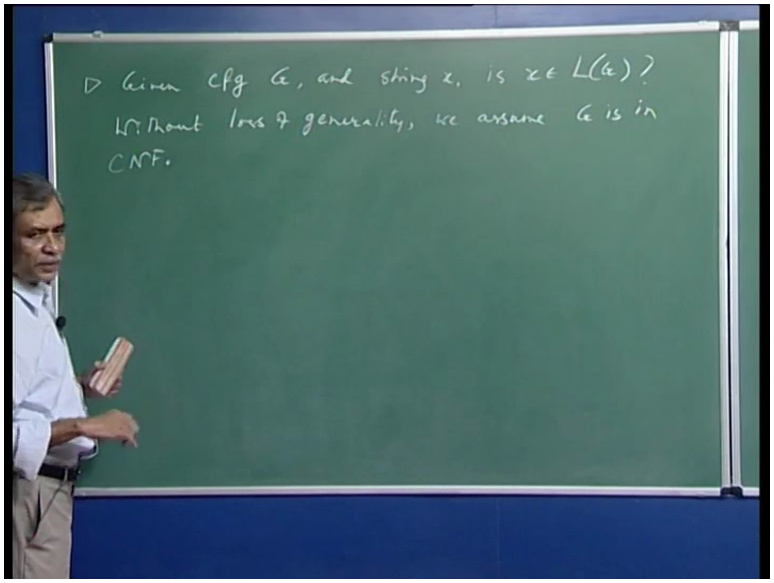
(Refer Slide Time: 17:19)




So now we consider a very important problem, the so called parsing problem that given a context free grammar G and a string x you would like to know if x is in the language generated by the context free grammar G. Now let me write this that without loss of

generality we assume that this is in Chomsky normal form. See any grammarGwe can createanother equivalent grammar inChomsky normal form such that the two grammars will generate the same language except possibly the empty strings.

(Refer Slide Time: 18:36)



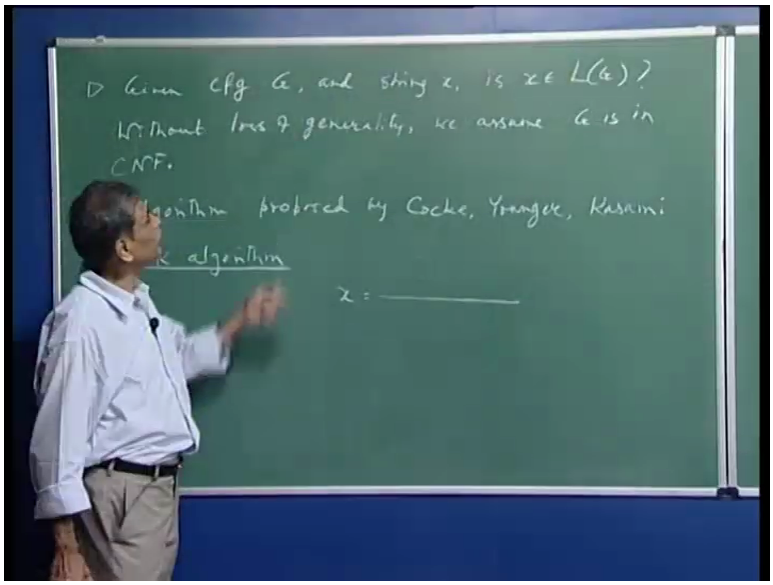

So the only case that isleft out will be the case of when your x is the empty string. But that can be handled separately and we will not spend any time on that, right? So it is not too difficult to see whether epsilon is produced an arbitrary grammar G because all it means is you remember that we defined the set of nullable nonterminals andthat means all those nonterminals which can derive the epsilon string. And if the start symbol happens to be nullable then of course the language will (con)(con) contain epsilon.

Socase of epsilon whenx is epsilon that can be taken separately. Otherwisewe can take that the grammar G is in Chomsky normal form. And now the idea is, actually there are two algorithms we can talk of. One is a kind of brute force algorithm and therefore it is extremely inefficient. Sowithout writing anything let me just say what is that brute force algorithm?

So basically what you can do is you can keep generating keep enumerating the parse trees of the grammar G in increasing order of its length of its frontier, right? Soit is of course messy but conceptually it is fairly simple, right? And so as you keep generating the set of all parse trees of the grammar G and once you cross the length of the string x then of course no further parse tree could have produced x. By then if you have notgenerated x as a frontier of one of the parse trees then of course x is not in the language.

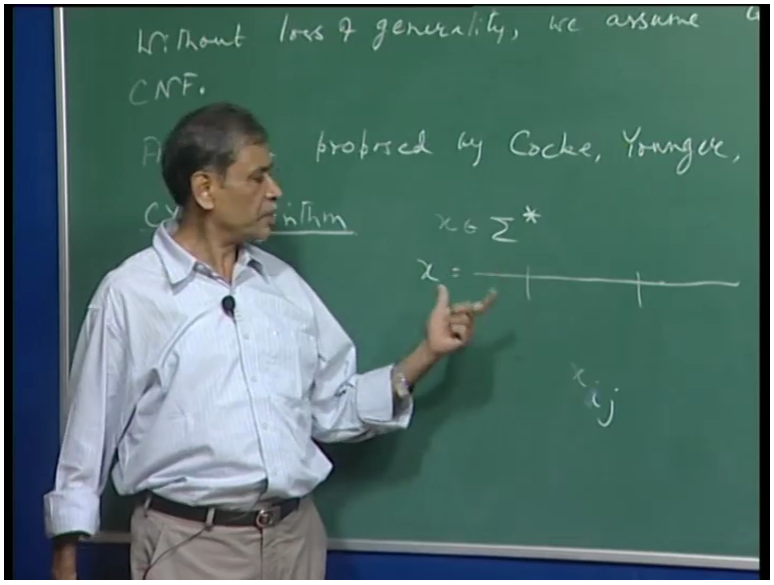
However you can see this will be an exponential time algorithm. As it happens we have a far better polynomial time algorithm for this problem. And that is an interesting algorithm. It wasoriginally proposed byCocke, Younger and Kasami, three people. Sotherefore this algorithm is called the CYK algorithm. So we will describe what CYK algorithm is? Essentially this is a dynamic programming algorithm. So suppose your x is this string and without againloss of generality we can assume x is a string of nonterminals.

(Refer Slide Time: 22:03)



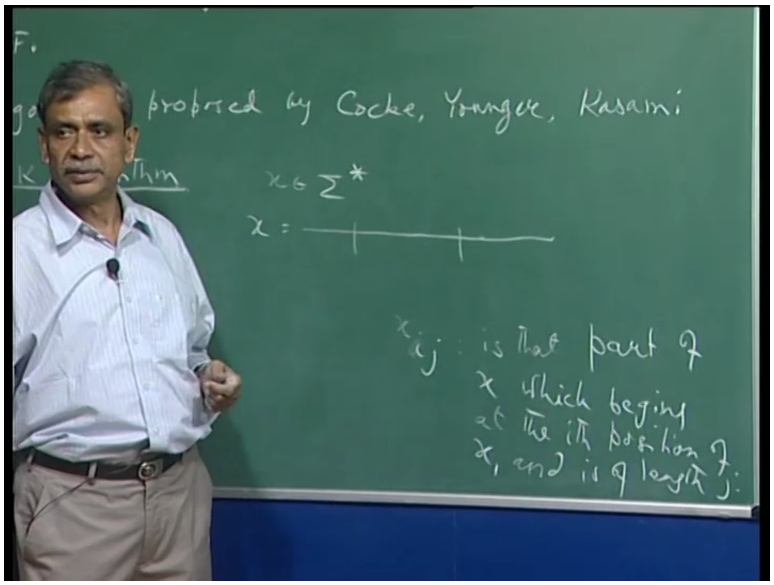

Although I have just said the string x but really speaking we are interested when x is astring of terminals. Solet me right away say that x is in sigma star where sigma is your set of terminals. So this is the string x, right? And let me identifyyou know various portions of this string as let me call this saying as x i j, this part provided and what is this way of naming substring contiguous substring of the string x?

(Refer Slide Time: 23:01)




What me mean is x i jis that part of x which begins at the ith position of x and is of length j, okay. Soyou know this is indexed by this x i j. So what I mean is x i j is that substring of x, in fact the unique substring of x which starts at the ith position of the string x and is of length j.

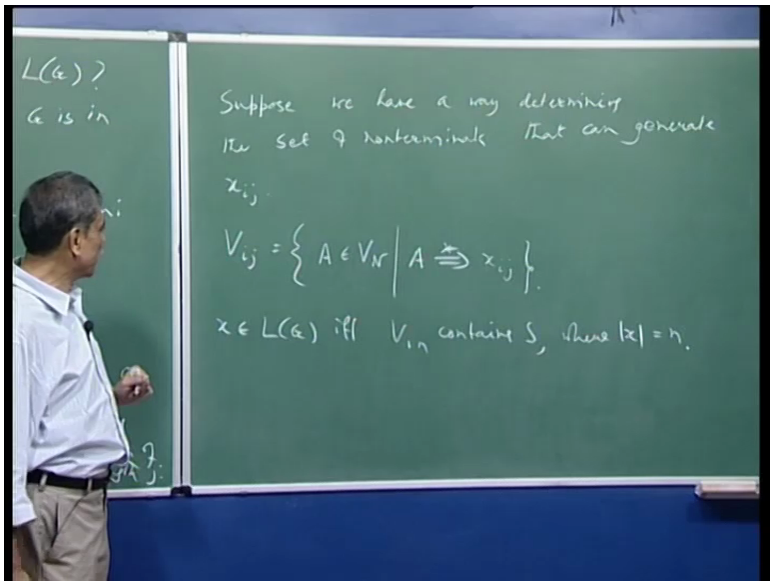
(Refer Slide Time: 24:11)




Nowin some way suppose we have a way of determining the set of nonterminals that can generate x i j. So let me call this set as V i j. Solet me write it kind of formally. V i j is a set of all nonterminals such that A,all thesenonterminals derive the string x i j. And suppose we managed to do it for all possible values of i and j, right? Then it is quite clear that the string x is in L G if and only V 1 n contains the start symbol where the input string x is of length n.

I mean this is reallyjust coming from the definition of the set V i j. So you know it is saying V i j is a set of all nonterminals which can generate x i j and the entire string x is of coursestarts at the position 1 and is of length n. So x i j is nothing but in this way of looking at x is nothing but x 1 n. So by definitionif S derives x 1 n then of course the string x is in the language. And similarly if x does not derive x 1 n then it is not in the language, right? So really we have not done anything much by stating this way.

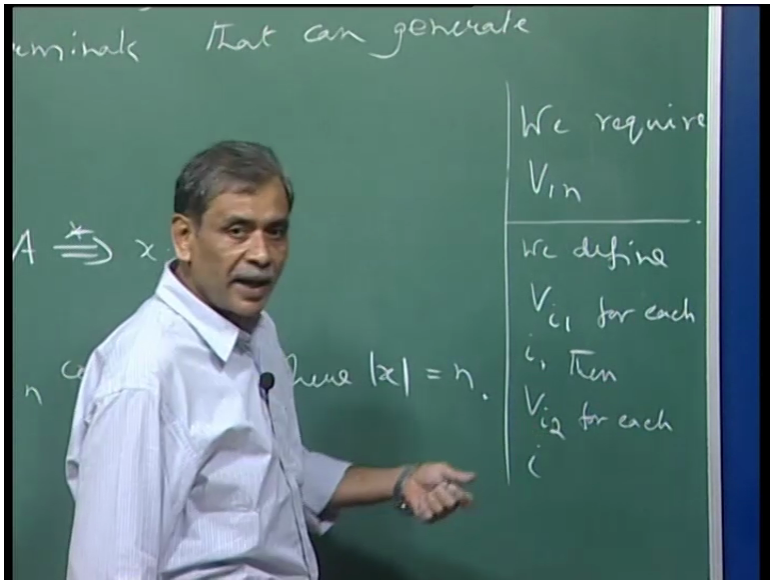
(Refer Slide Time: 27:18)




Sohowever interesting thing is what we require is of course we require therefore V 1 n, is not it? In order to see whether the string x is in the language or not? Now what we are going to do and that is done in a typical dynamic programming paradigm that we will start with the smallest value of j for whichyou knowthese symbols are defined. And then so essentially first we define V i 1 for eachi, then V i 2 for each i.

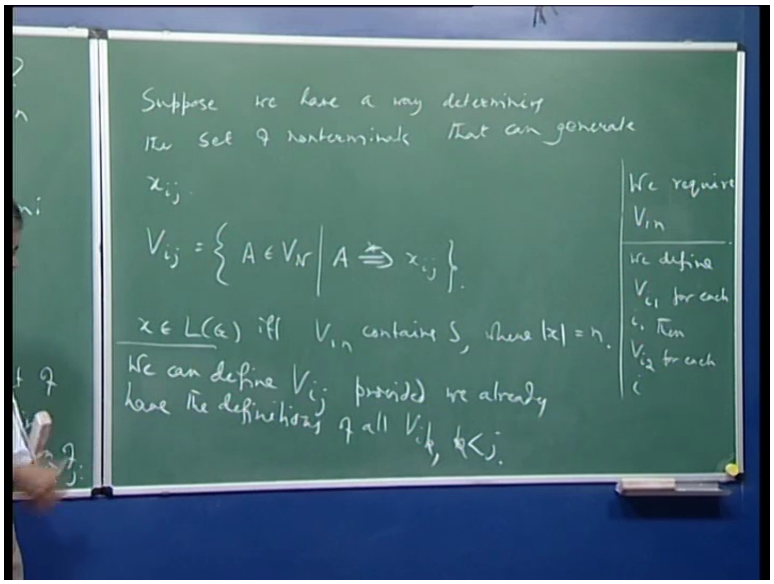
See the range of i in these two cases will be different. And this way we go up, right? Here i will range from 1 to n. Here i will range from 1 to n minus 1 because you know at position n you cannot have asubstring of length 2 starting at position n.

(Refer Slide Time: 28:44)




There can be only a substring of length 1. So point I ammaking is that this is the crucial point that given V i j, I mean let uswrite it slightly differently. We can define or determine V i j provided we already have the definitions of all V ik where k is strictly less than j. Ofcourse cannot be less than 1, right? So that is the range.

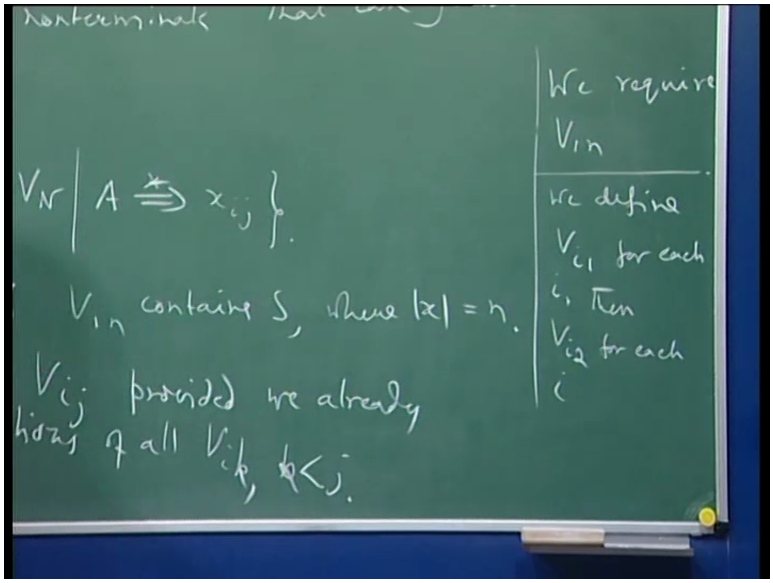
(Refer Slide Time: 30:18)




Sowe willget into the details a little later but alreadyyou can see the dynamic programming flavor of what we are trying to do because you see these things are very easy todetermine. What are V i 1? We will see that. Sothe point is that from V i 1, I will be able to determine V i 2. From V i 1 and V i 2, I will be able to determine V i 3. From V i 2, V i 2, V i 3 I will be able to determine V i 4 and so on.

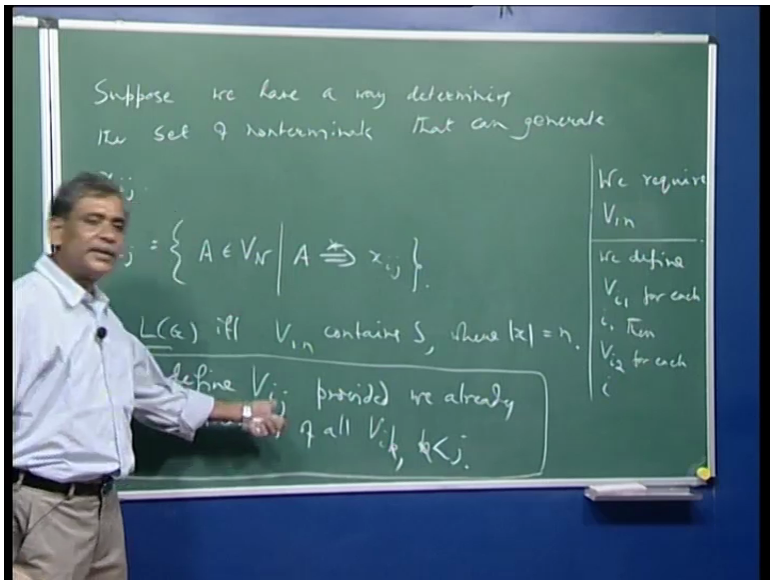
So this way we keep on building upwards for this index and then finallywe will be able to determine what is this set when j is n, and then we are through, right? We require V 1 n and that is the dynamic programming approach as you can see.

(Refer Slide Time: 31:19)




Let us therefore prove this crucial thing. I mean you see that if we can manage to prove this that we can define value for a value j provided we have the definitions for all the j's which are less than this particular j, right?

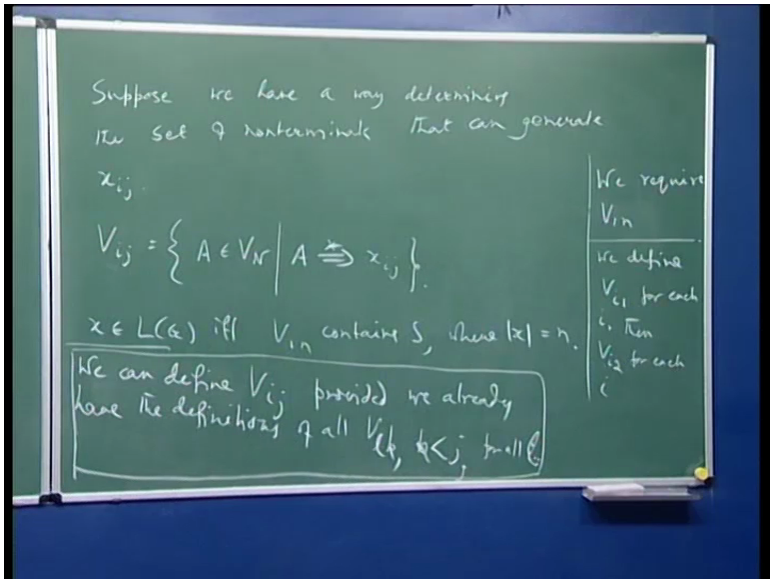
(Refer Slide Time: 31:54)




This is what we said that we can definethe set V i j. You know the set of nonterminals whichgenerate the substring starting at i of length j provided we know all those nonterminals which can generate for all V i k. Now see basically here actuallyI will not only need V i k but

also and let me just say for all i as well, right? Because I amgoing to need. Solet me write it this way. For all V L k where k is less than j. Where L is you know all possible values. So this is what we are saying.

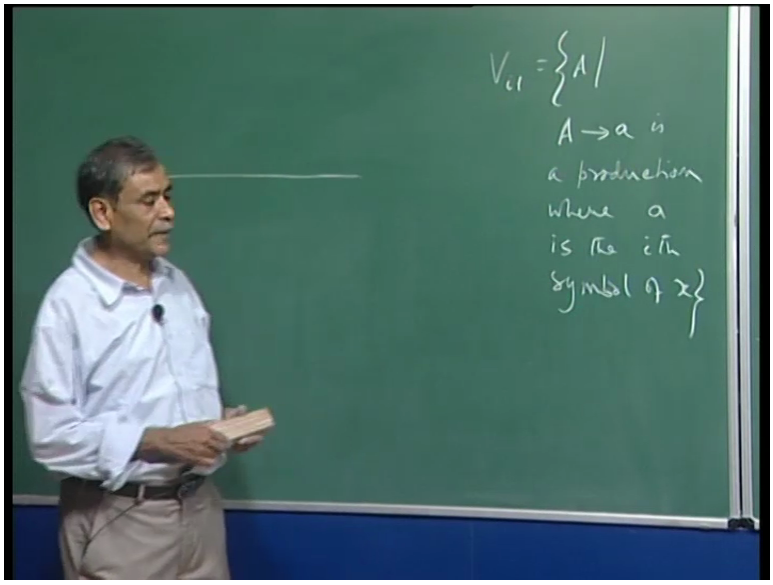
(Refer Slide Time: 33:04)




In other words thatyou know the setyou can define for a higher range provided you know the definitions for lower ranges. So that is the basic idea and let us see why it is (())(33:22)? First of all what is V i 1? V i 1 is the set of all nonterminals which generate the substring of length 1 starting at position i of thestring x. So is not it that V i 1 is essentially all those nonterminals A such that A goes to a is a production where a is the ith symbol of x.

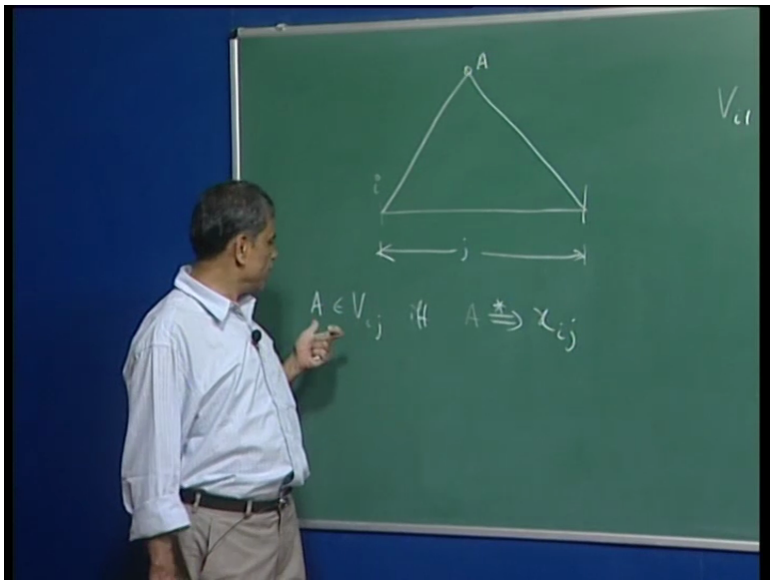
This is very simple, right? Because V i 1 is basically string of length 1 and string of length 1 can be generated by some nonterminals directly using the production. And so you have seen V i 1 is essentially all those nonterminals A such that A goes to a is a production where this a is that substringof length 1 at position i, right?

(Refer Slide Time: 34:45)



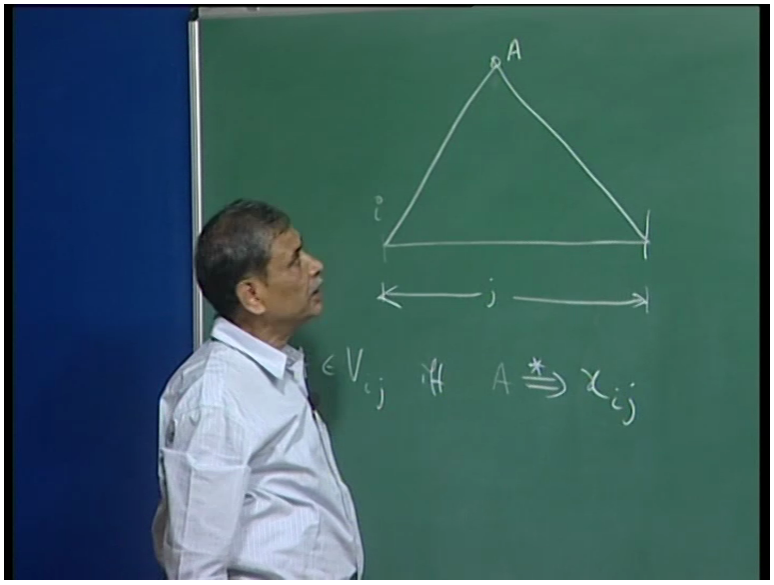

You know both i, I mean you know i, so therefore you know what is a? So you can just pick up this. Sonow let us assume that I am talking of asubstring of length j, right? And suppose this substring starts at position i, right? Which all nonterminals can produce it? So suppose nonterminalA produces this. So I will have essentially what it means is that, I am just writing it slightly differently, A is in V i j if and only A can derive x i j, right? That substring of length j starting at position i.I mean this is of course the definition of V i j.

(Refer Slide Time: 35:59)



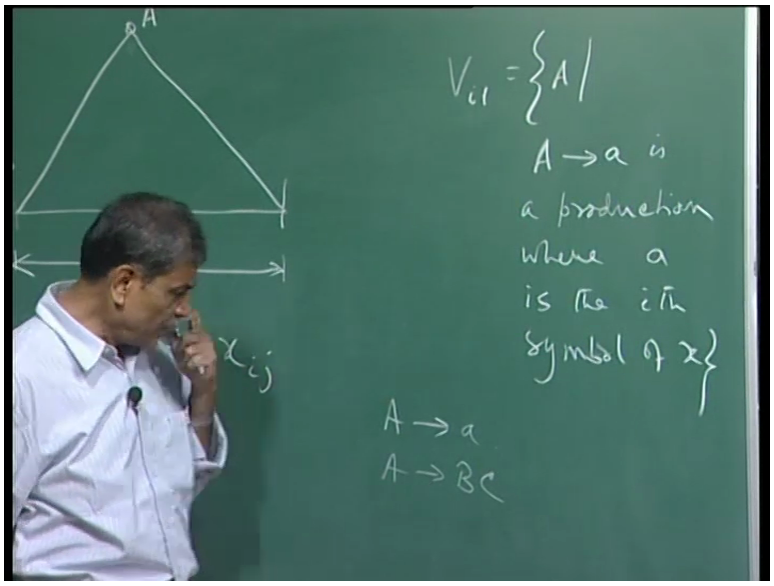

So we are saying A is in V i j if A can derive this. So consider this parse tree which generates x i j starting from the nonterminalA. Now can this (sub)derivation tree, I mean what can be the first production in this derivation tree? Can it be easy?

(Refer Slide Time: 36:28)



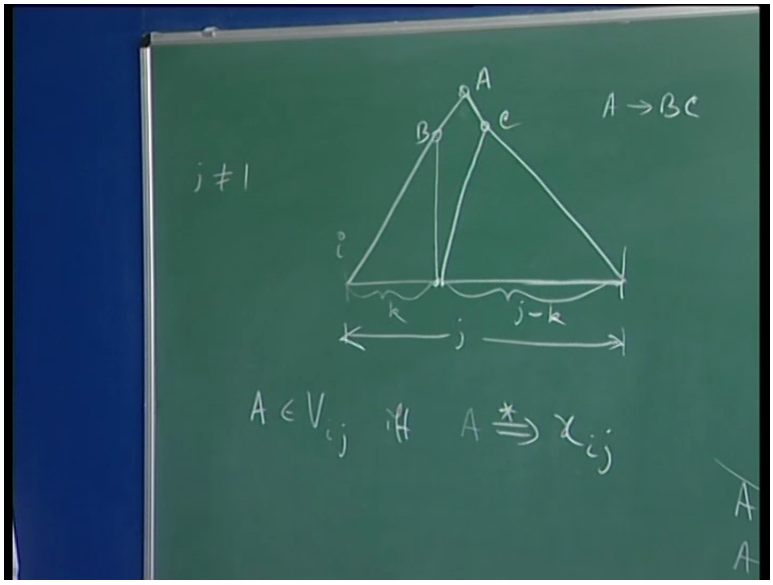

Remember in Chomsky normal form there are only two kinds of production. Either A goes to a where a is the terminal or A goes to B C where these are two nonterminals. Of courseboth of them can be same as A. So B and C in general two nonterminals.

(Refer Slide Time: 36:49)



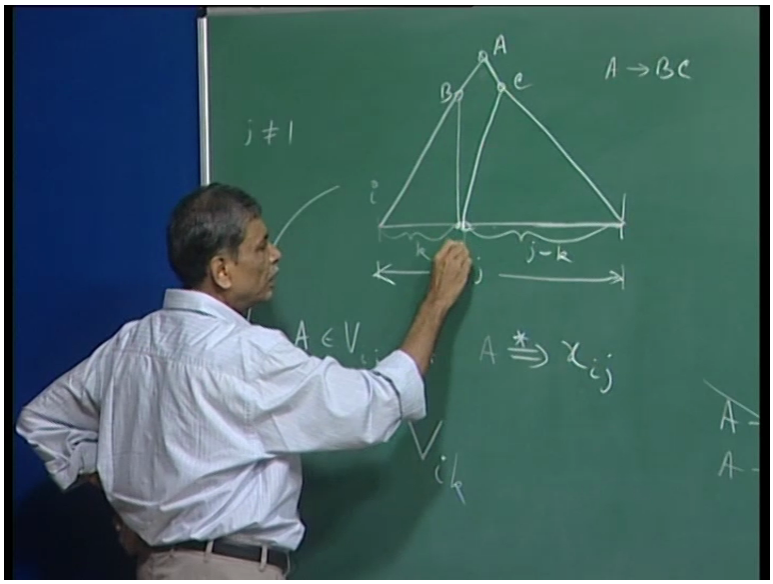

So you will agree with me that if j is not equal to 1 then clearly this production cannot be the first production that we use here in this derivation tree. So it must would be the case that we have used the production some A goes to B C in this derivation tree, okay. So then this B will generate a left half of the tree and C will generate the rest of the, sorry left half of the string x i j and C will generate the rest of it. Andlet me say that this length is k. Sothis length is of course j minus k, correct?

(Refer Slide Time: 37:53)



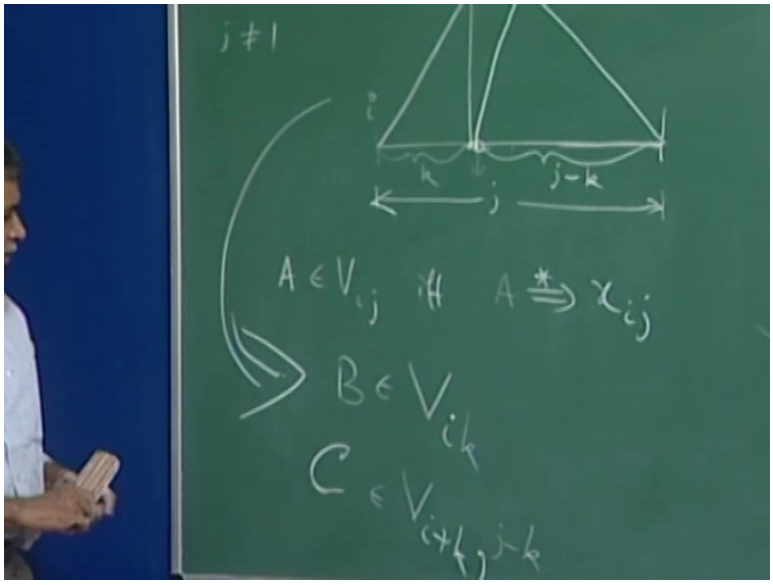

So therefore can you see what B is in terms of these V i j?So clearly B is a nonterminal which generatesthe substring of length k starting at i. In other wordsB has to be a member of, you know this implies that Bis a member of V i k. And where does this string starts from? What is this symbol? Not what is this symbol but rather what is this position?

(Refer Slide Time: 38:47)




This is clearly the position i plus k, correct? Is not it? So then similarly I mean it is not difficult to see that C has to be a member of V i plus k. And as we said thislength is j minus k, so it is j minus k, right? B is in V i k and C is in V i plus 1 k comma j minus k.

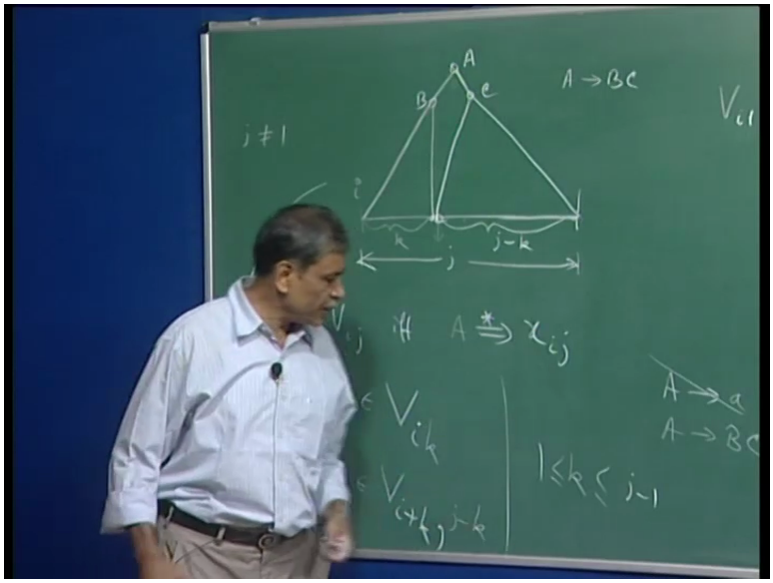
(Refer Slide Time: 39:35)




Soin other words and by the way what is k? What can k be? What is the smallest value of k? See k cannot be 0. What would that mean? K cannot be 0 because that would mean that basically this C is generating this entire string and therefore B is generating the empty string. But in Chomsky normal form grammar no nonterminal can generate empty string.

So k has to be at least 1. And how big k is? For the same reasonthe partC generates in this derivation tree cannot be empty. So this is less than equal to, you know you can go j minus 1, right? This kcan range from 1 to j minus 1.

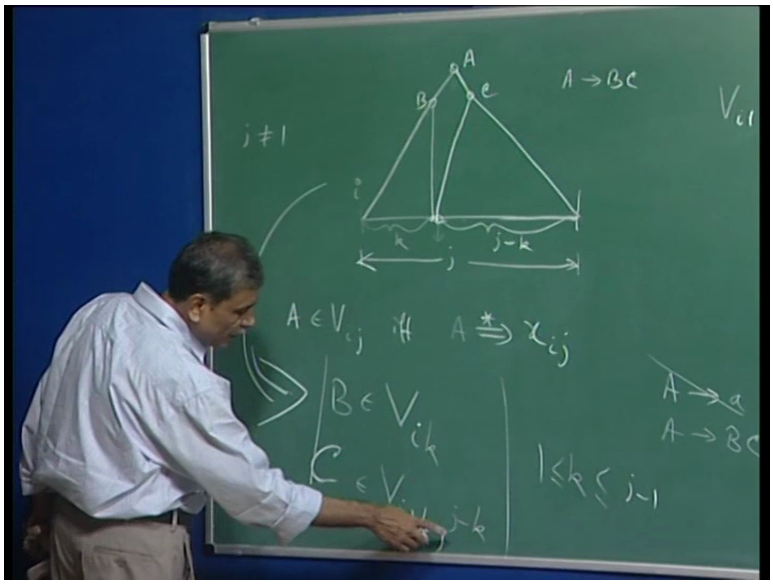
(Refer Slide Time: 40:36)




And now you see that A is in V i j if and only if there exist. So let me say this. A is in V i j if and only there exist B and C such that B is in V i k, C is in V i plus 1 k comma j minus k.

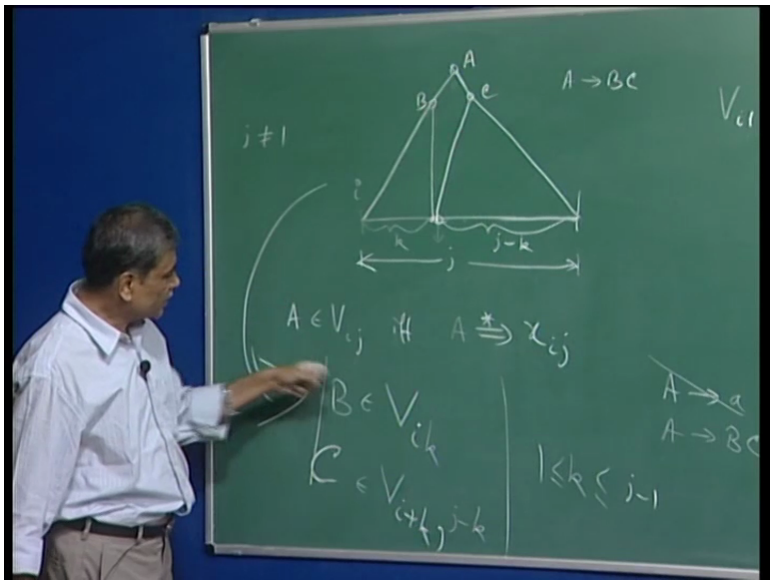
And A goes to B C is a production of the grammar. Sodo you see that suppose in other words you just think of the other way that suppose you had the definitions ofall V's where the second index isstrictly less than j, right? Then surely because you see the second index here I am using k and here j minus k.

(Refer Slide Time: 41:36)



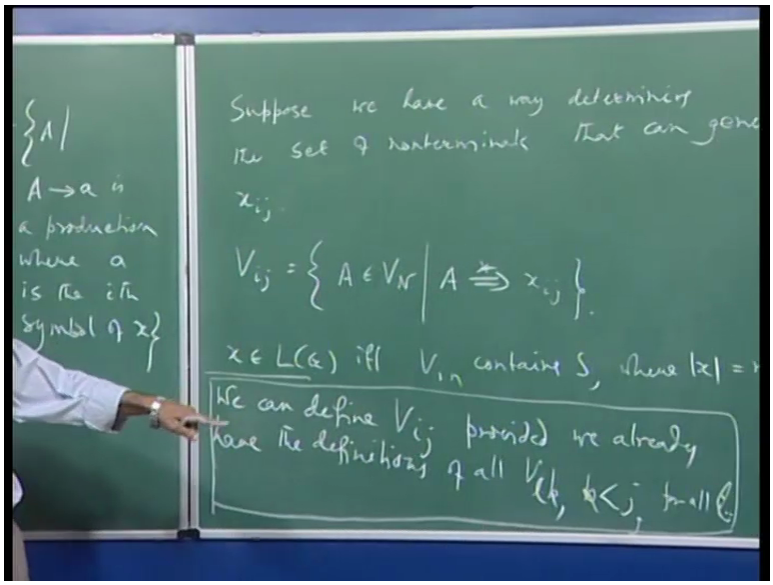

And in both these values are strictly less then j. So if I can have the definitions of V i k such that k is strictly less than j then I can determine whether or not A is in V i j and thereforeI can take each of the nonterminals, internally look at it that whether or notit will be in this set and therefore I can define the set V i j.

(Refer Slide Time: 42:08)



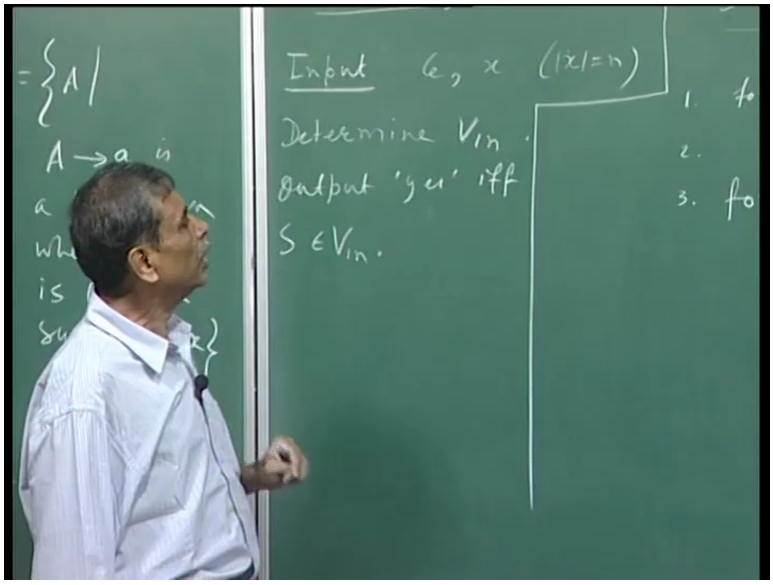
In other words this is the dynamic programming recurrence. I am not writingin that form but you should be able to see that I can define V i j provided I have the definitions of V L k where k is strictly less than j. And that is the idea. So therefore you see what we started off by saying that we need to prove this in order to get the basic crucial idea behind the dynamic programming algorithm. That we have managed to demonstrate that we can define V i j provided we already have the definitions of all V L k wherek is strictly less than j.

(Refer Slide Time: 42:55)




Therefore you can see the CYK algorithm is going to be the higher level, very simple to describe. Your input is Chomsky normal form grammar G, string of terminals and suppose the length of the string is n, then determine V 1 n and output yes. That means x is generated by G. X is in the language L G if and only if the start symbol of G this start symbol S is in V 1 n. So now all I need to do is to tell you the more precise manner how V 1 n will be determined? So how do we determine V 1 n?

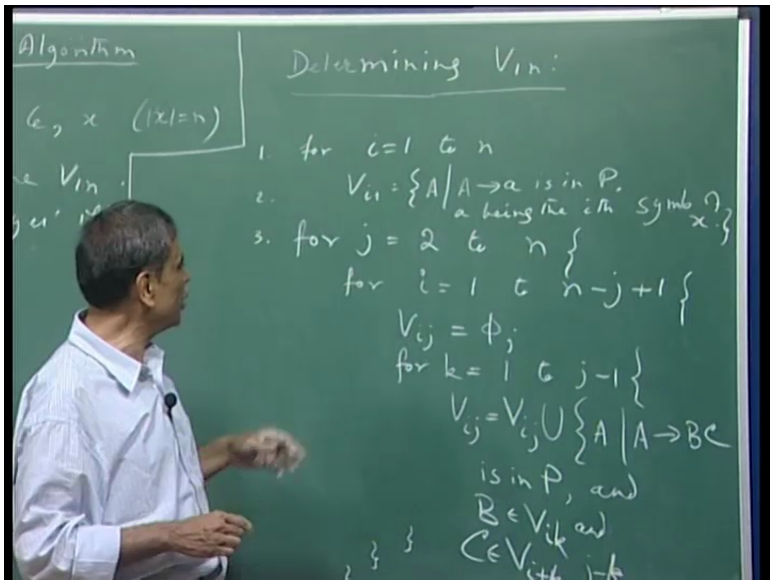
(Refer Slide Time: 43:52)




Which is what we need to do first of all we determine all the V i 1 for different values of i and that is the same thing that we have written here that V i 1 isall those set of nonterminals A such that A derives small a is in P. This small a being the ith symbol of x. This is quite clear. Now what you are going to do? So therefore we have determine all V i 1. You know this length 1 things we have determined and then we will have this j.

J will vary from 2 to n,now the bigger lengths that we are considering. Already length 1 V's are taken care of. So j will range from 2 to n and we will build up in that order. That is the dynamic programming.

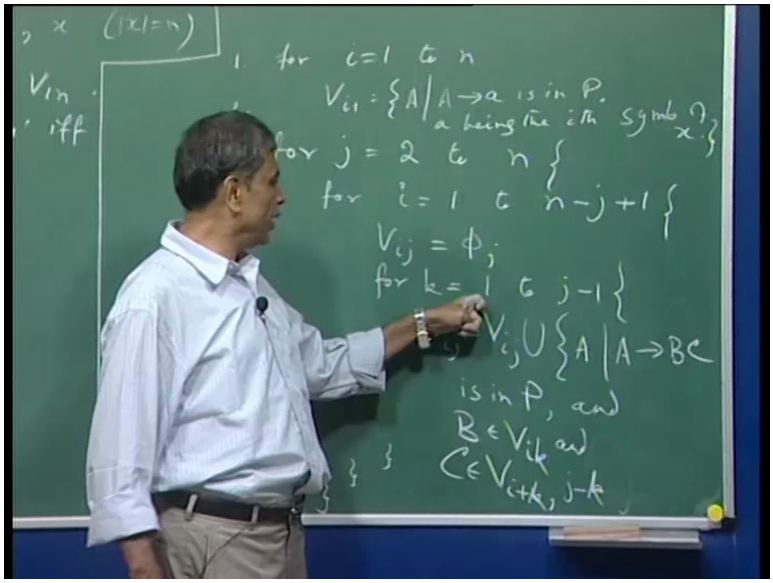
(Refer Slide Time: 44:52)



So j is(rang) ranging from 2 to n. So starting from 2 then 3 thenall the way up to n. What we do?Now once Ihave fixed a j then for that j, I define all V i j's. See that is the idea. So I can range this once j is fixedfor some value of k and clearly i can range from 1, is not it? Such a substring can start from the first position of thegiven string x and all the way up to n minus j plus 1, right?

Sonow we are defining V i j and this is just to say that V i j is empty to begin with because you know it can be empty. And then you see what I have written here is exactly the same thing that I am writing that for k ranging from 1 to j minus 1. This is what we had seen if you remember. If you go back and you can see this that k can range from 1 to j minus 1. That is exactly what we are writing here.K is 1 to j minus 1.

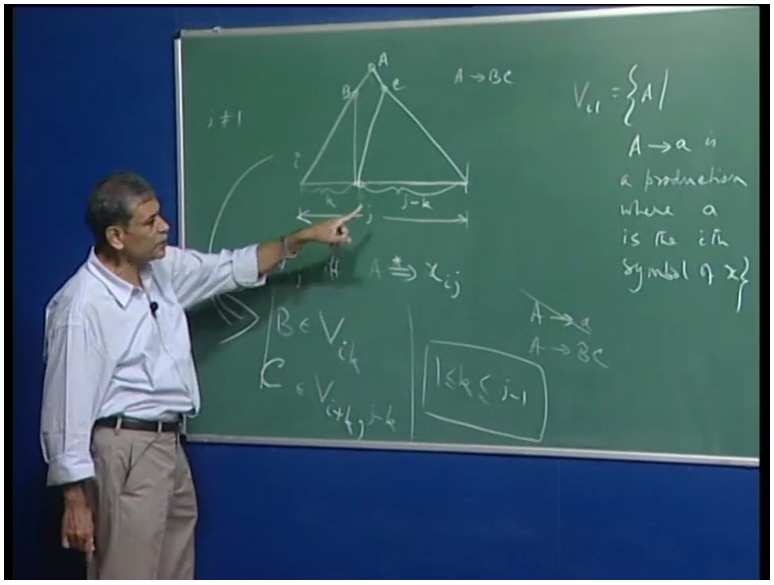
(Refer Slide Time: 46:15)




The V i j is basically this set and I am taking it with the old union so that in case this is empty this whole thing will become empty. That is fine. So is the set of all A. See it islike this that we fix a value of k and get something and all that. So what we are doing is thatall those A such that A goes to B C is a production and B is in V i k and C is in V i plus 1 k j minus k.

That is exactly what we have written here that this B part generates the string starting from i of length k for some value of k that we have set. And C is generating the rest of the string V i plus k. So therefore C has to be in V i plus k because that string will start from this position, i plus kth position and its length is going to be j minus k.

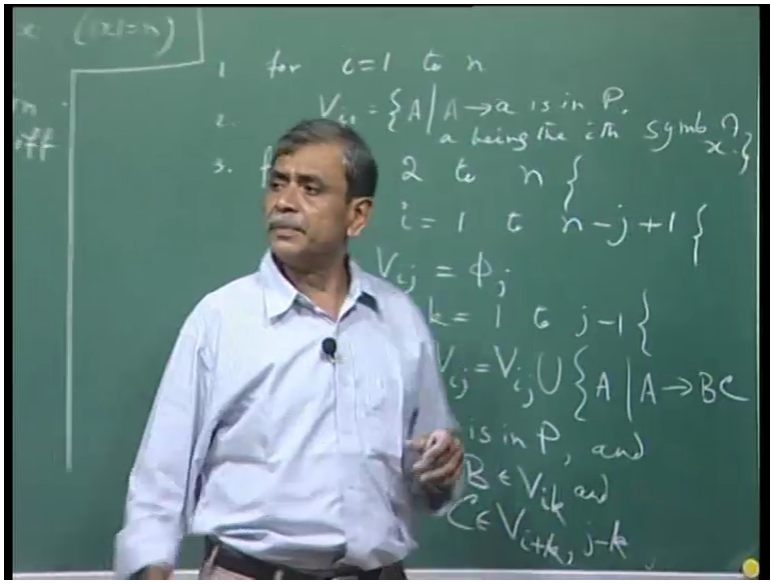
(Refer Slide Time: 47:20)




That is what we have written. So you seeessentially we were saying that first see all those A's such that it goes to B C and B generates a length 1 string and C generate rest of the string. And then with that we take the union. You knowas k is varyingwe are you know getting different sets here and(uni) I meanwe are taking their unions and therefore finally when we come out of this loop V i j would be determined, right?

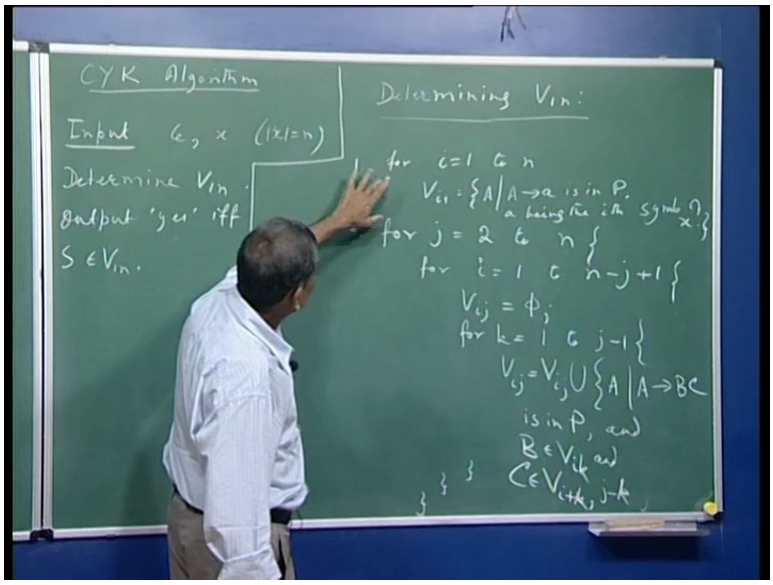
For certain value of j and then we go to the higher value of j, j plus 1 and then j plus 2 and so on. And finally when I managed to (der) find the value n for j then that is of course V 1 n. And then I look at whether the start symbol is in V 1 n. If so, I say yes the string is generated by the grammar. Otherwise I say the string is not generated by the given grammar G.

(Refer Slide Time: 48:38)



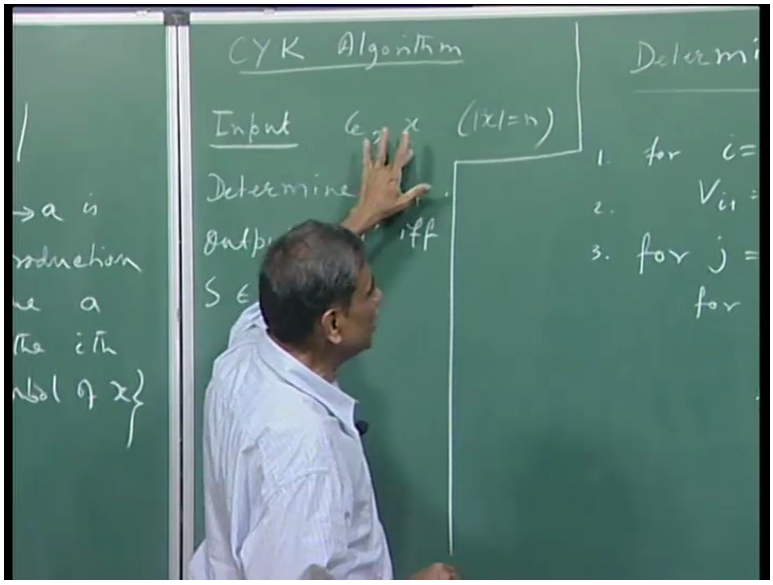
So that completes the description of our CYK algorithm. And in factas I have said that this is an interesting algorithm. And we said also that this is a fairly efficient algorithm and at leastit is a polynomial time algorithm. So the time complexity of this algorithm is not too difficult to see. You see this part, this is for loop which is what?

(Refer Slide Time: 49:08)



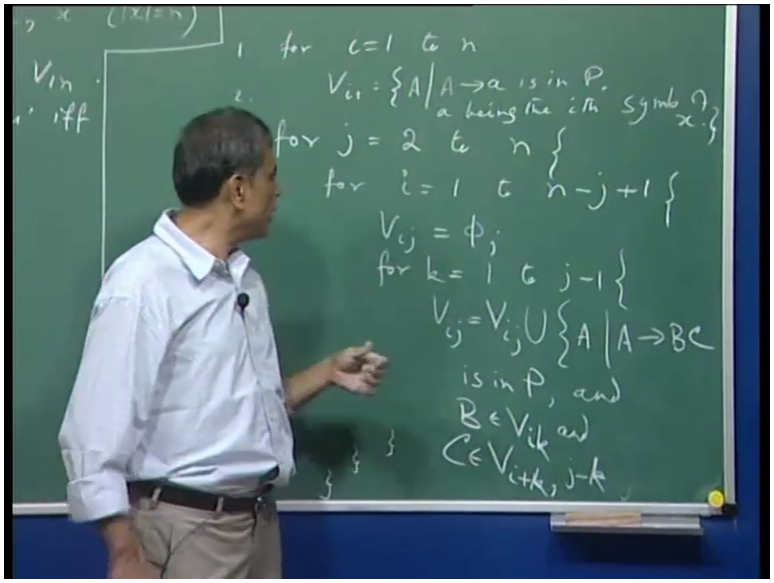

So you are iterating this for loop asyou knowthis body will be repeated n times where n is the length of the string, right? Length of the string x is n. And V i 1 is essentially what? You know how much time would you take to do this? Essentially you are looking up the production set and the symbol A and therefore you can generate this. As you can see this will be linear in the description of the grammar and x.

(Refer Slide Time: 49:48)



And now therefore what will be thetime complexity determining factor for this algorithm is essentially this for loop. But this for loop nesshas within it two other for loops, you see. This for loop, this range is n of the order n.

(Refer Slide Time: 50:18)

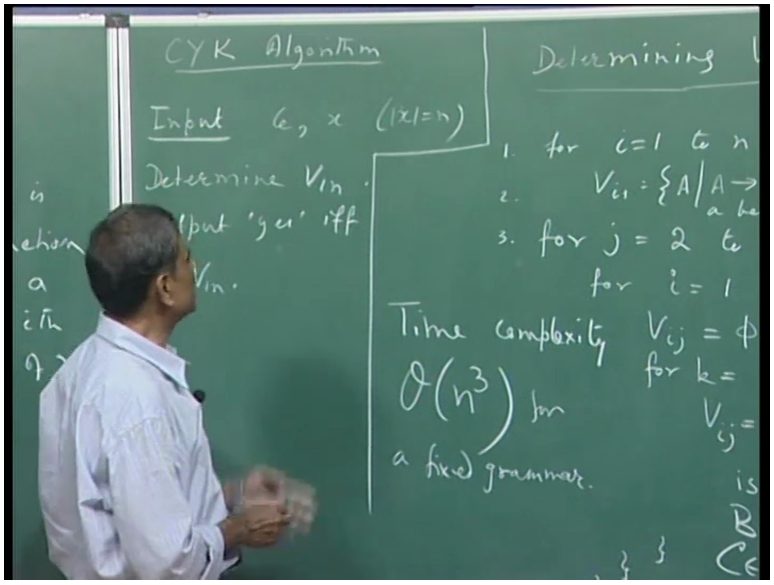



This is again iit starts from 1 and the highest value is you know whatever j is starting from 2. So n minus 1 again that is order n. And since j can go all the way up to n,so this k can be at most of order n. So you can see this entire loop thing will be done in order n cube where n is the length of the string. And this body what you are doing? You are looking up the production set, right?

And you are looking up the already defined values of V's and that again to look up the production set that will be of the order ofyou know looking at the production set would mean looking at examining G. So again we can say this is essentially the time complexity will be determined by bounded by n cube and multiplied by the size of the grammar. And if your grammar is already known because often the problems are likethat the grammar is fixed and you are essentially giving different strings.

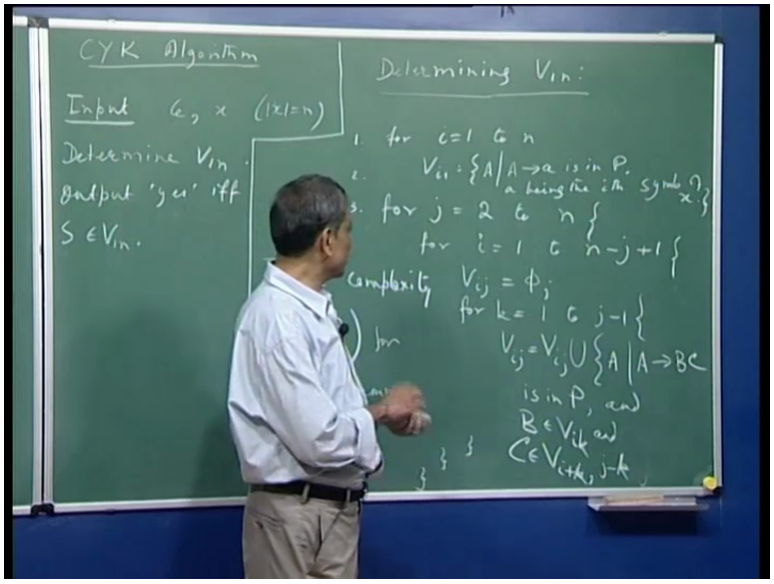
And to see whether that string is in the language or not generated by the grammar. In that case this length of the grammar G is a constant. So thereforewe can see the time complexity is order n cube for a fixed grammar G.

(Refer Slide Time: 52:15)




And when G is also in the input so basically it is size of G multiplied by this n cube. Andsee this n cube is coming from these three nested loops.

(Refer Slide Time: 52:27)




Each one is of order n and hence n cube. So that is clear. So that completes the time complexity. It is a very simple time complexity computation for CYK algorithm. Thanks.