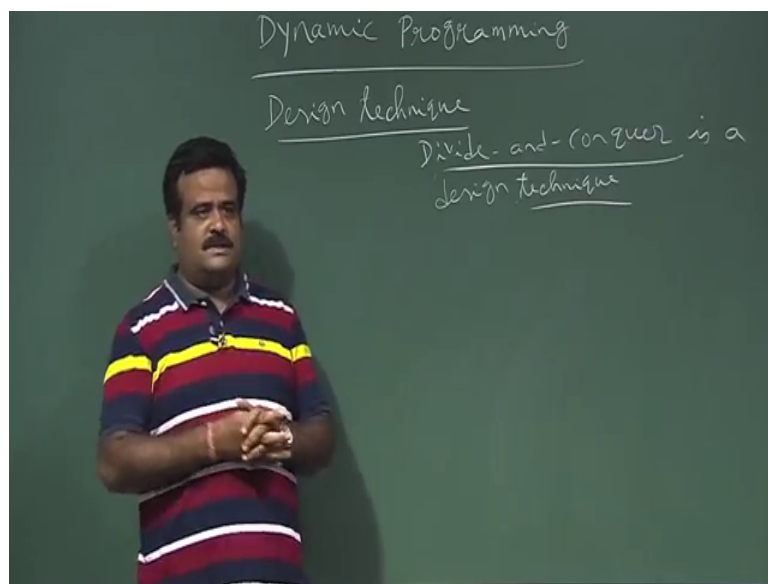


**An Introduction to Algorithms**  
**Prof. Sourav Mukhopadhyay**  
**Department of Mathematics**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 36**  
**Dynamic Programming**

So, we talked about dynamic programming. So, it is a design technique like divide and conquer.

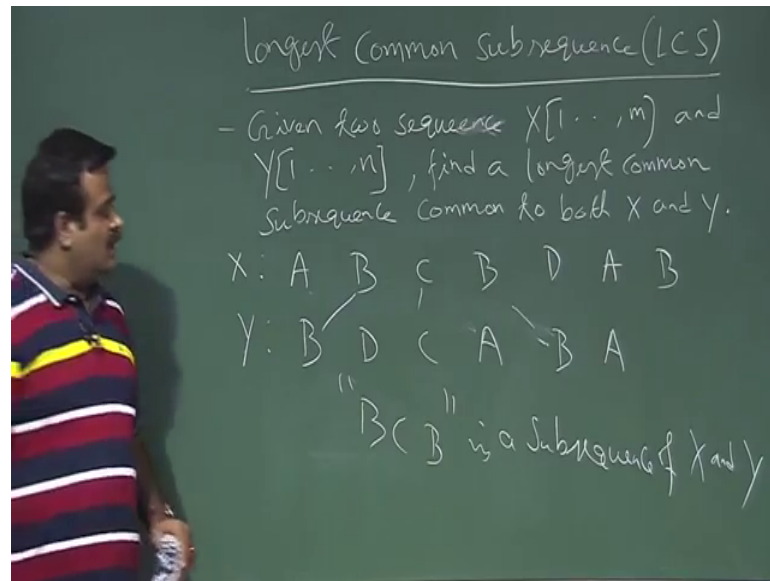
(Refer Slide Time: 00:29)



So, we know one designing technique is divide and conquer technique. So, there this is also a; this is a design technique we know. So, we have seen merge sort quick sort all that divide and conquer technique. So, like we have a problem of size and we reduce the problem into sub-problems of lesser size and that is the divide step and in the conquer step, we solve recursively solve this sub-problems and once we got the solution of the sub-problems then we combine the solution of the sub problem to get the solution of the whole problem. So, this is the divide and conquer technique. So, dynamic programming is also a design technique.

So, we will discuss this through an example through a problem which is called longest common subsequence problem.

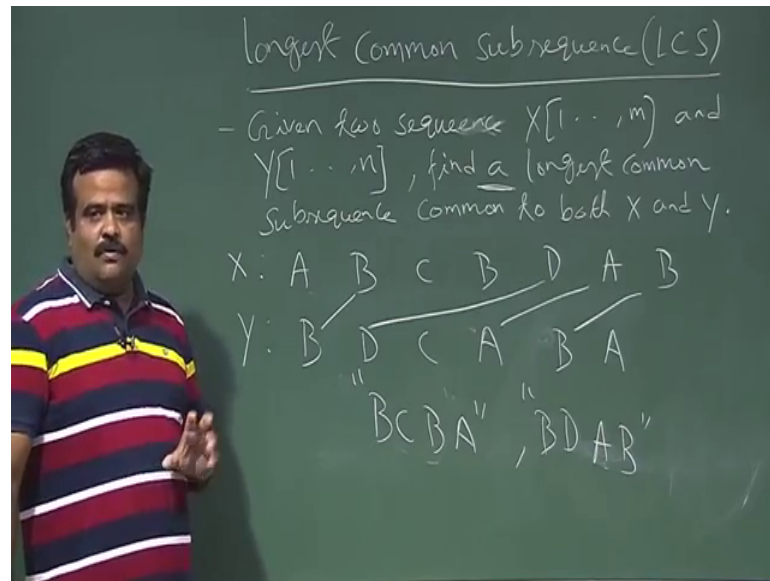
(Refer Slide Time: 01:54)



So, we learned this technique through an example which is called longest common longest common subsequence problem LCS. So, what is the problem? Problem is we have given 2 sequence X and Y. So, given 2 sequence X is of size m and Y is of size n. So, we need to find a find a longest common subsequence which is common to both X and Y. So, this is the problem. So, this is called LCS problem. So, finding a longest common subsequence between 2 sequence X and Y, suppose we have let us take an example suppose X is a sequence like this A, B, C, B, D, A, B and why is there subsequence B, D, C, A, B, A. So, suppose these 2 are my given sequence.

Now, we need to find the longest common subsequence. So, first of all, we need to find a subsequence. So, for example, A, B is a subsequence of this because this is A then B. So, A B is a subsequence A B is common to both of them, but A B is of length 2. Now do you have any length 3 subsequence. So, like B, C, B. So, B, C, B is also is a subsequence of X and Y, but this is of length 3. now we want to see whether do you have any subsequence of length 4.

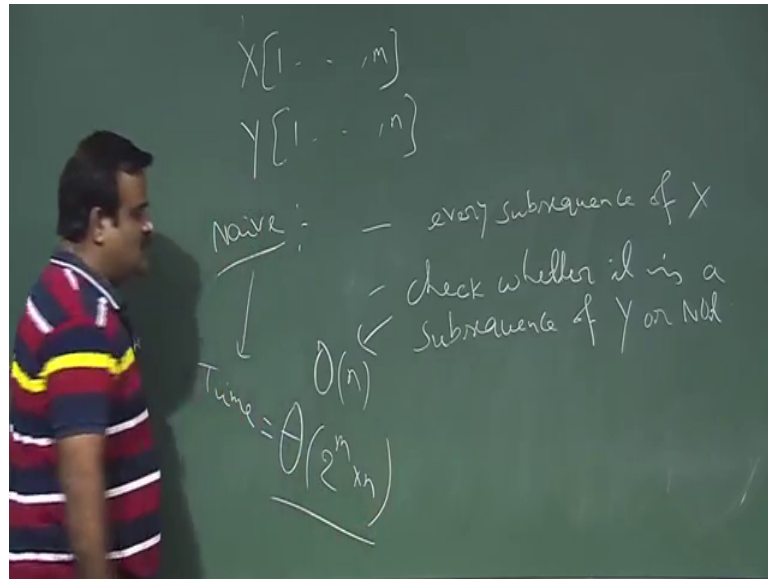
(Refer Slide Time: 04:45)



So, let us try that. So, do you have any subsequence of length 4 yes B, C, B, A. So, our B, C, B, A; this is a subsequence of length 4. Now we can check is there any subsequence of length 5. So, that we can verify there, we can check that there will be no subsequence of length 5 so; that means, the length of the longest common subsequence is 4 and this is one of them because we may get another subsequence like this. So, D, D, A, B. So, B, D, A, B. So, this is also subsequence of length 4.

So, that is why it is a longest common subsequence not the longest common subsequence. So, there could be many subsequence of length 4, but the longest length is unique. The length of the longest common subsequence is unique which is here is 4, but we may have many such subsequence. So, that is our problem. So, how to get that such longest common subsequence; how to find a longest common subsequence from a given 2 sequence  $X$  and  $Y$ ? So, what is the main approach? So, what is the main approach how we can proceed?

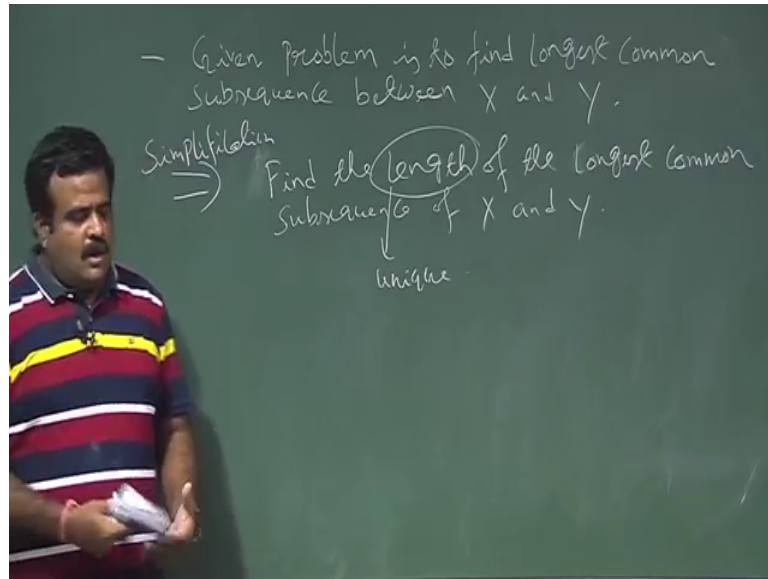
(Refer Slide Time: 06:37)



So, main approach could be we can take a subsequence we have given to sequence X and Y. X is of length m and Y is of length n. So, this is the neighborhood. So, what we do? We take a subsequence from X and we check whether that is a subsequence of Y or not. So, this is the approach check every subsequence of X is. So, we take a subsequence, we take every subsequence of X and check whether it is a subsequence of Y or not subsequence of Y or not. So, we take the subsequence from X and then we check whether that is a subsequence of Y or not. So, to check whether it is subsequence of Y or not, it will take order of n time because this is the length of the Y sequence. Now how many subsequences we can have from X. So, there is that is the power set basically.

So, basically this approach there could be 2 to the power n subsequence. So, this approach will take order of 2 to the power m into n exponential time algorithm. So, which is not acceptable this is very expensive this is exponential time algorithm to the per m not even polynomial time. So, , but this is this is one this is a would force method where one can take one can try for all possible subsequence of X and can see try to see whether this is a common subsequence of Y or not. So, now, we want to do something better. So, to do better; what we do? So, we transfer we convert this problem into we want to simplify this problem. So, basically our problem is to find the longest common subsequence between X and Y.

(Refer Slide Time: 09:05)

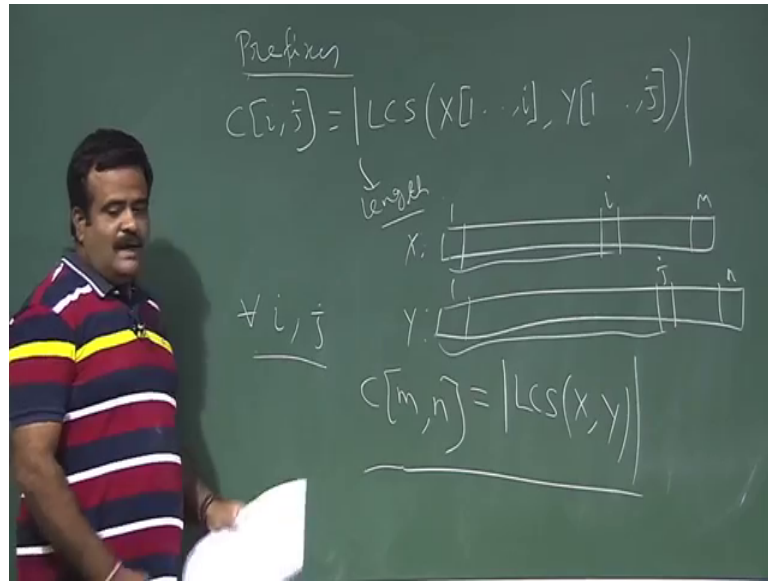


The given problem is problem is to find longest common subsequence between X and Y. So, what we do? We just simplify this problem instead of finding the longest common subsequence we want to find the length of the longest common subsequence. So, we take the simplified version of this simplification.

So, we want to get find the length. Length of the longest common subsequence of X and Y and that length is unique this length is unique in the example, we have seen there are many longest common subsequence there are many subsequence of length 4 and that is the longest one, but the length of the longest common subsequence that is 4 that is unique. So, this length is unique. So, now, we convert this problem into the problem of problem finding the length of the longest common subsequence and from there we will try to get the greater longest common subsequence. So, this is something simplification of this problem. So, we first try to get the length of the longest common subsequence which is unique and then after getting the length from that elbow, which we will use basically we will use the dynamic programming technique then from there after getting the length.

We will try to find out the longest common subsequence. So, this is the problem. So, now, we want to find the length of the longest common subsequence.

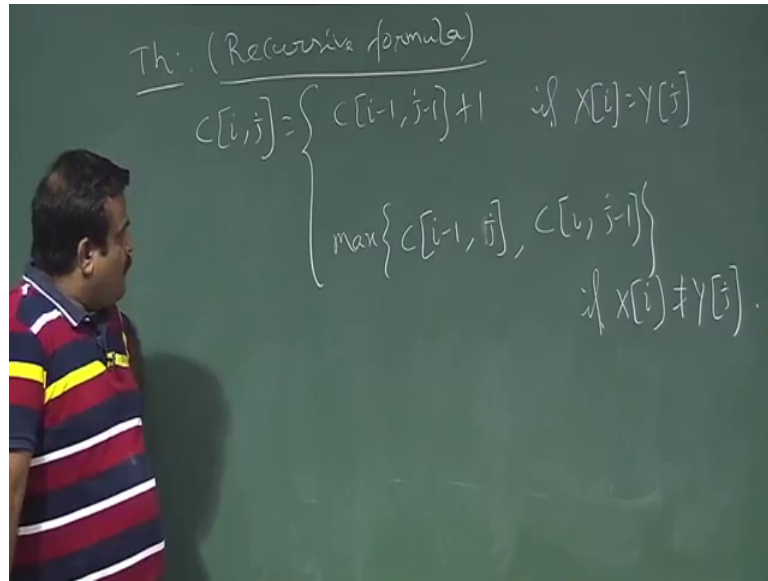
(Refer Slide Time: 11:38)



So, for that we will use some prefix notation. So, we will use some notation on X and Y. So, prefixes. So, we denote the  $C_{i,j}$  which is basically length of the longest common subsequence between  $X_1$  to  $i$  and  $Y_1$  to  $j$ . This is the basically length. So,  $X_{i,j}$ ; so, we have a X sequence which is this is the X sequence which is from 1 to m and we have a Y sequence which is from 1 to n. So, basically we take a sub; we take a subsequence of this up to  $i$  and here say up to  $j$ . Now this is  $X_1$  to  $i$  and  $X_1$  to  $j$ . So, we denote  $C_{i,j}$  is basically the length of the longest common subsequence of X up to  $i$  prefix  $i$  and Y up to prefix  $j$ . So, now, if we can find out this  $C_{i,j}$  for all  $i$  and  $j$  then we are done why because we want to find the length of the longest common subsequence of X and Y.

So, basically we are looking for  $C_{m,n}$ . This is basically length of the longest common subsequence of full X and full Y. So, if we can find out  $C_{i,j}$  for all  $i$  and  $j$  then we can then we are done because we can get the  $C_{m,n}$  also. So, now, this is the definition. So, now, we want to find out a recursive formula for this  $C_{i,j}$  and that formula will help us to have a algorithm design technique which is called dynamic programming technique. Every dynamic programming technique we should have such kind of recursive formula. So, that we have to define the formula. So, let us have a theorem on that  $C_{i,j}$ s.

(Refer Slide Time: 14:09)

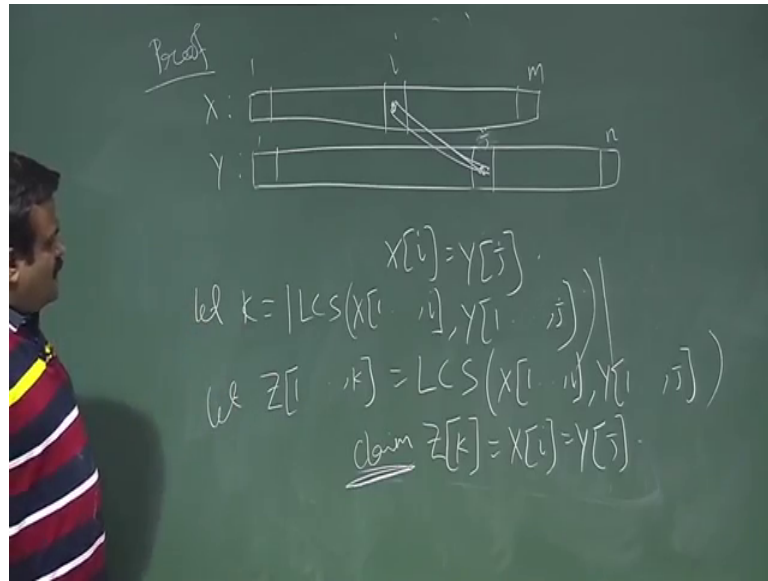


So,  $C[i, j]$ ; so, this is the recursive formula for  $C[i, j]$ . So, let us write this in a theorem. So, it is telling this is the recursive formula the recursive formula for  $C[i, j]$ .

So, what it is telling recursive formulation or formula it is telling us  $C[i, j]$  is basically  $C[i-1, j-1] + 1$ . If  $X[i] = Y[j]$  otherwise, it is maximum of this 2 term maximum of  $C[i-1, j]$  or  $C[i, j-1]$  if otherwise. So, this is the recursive formulation for  $C[i, j]$ . So, see  $C[i, j]$  is the length of the longest common subsequence of the prefix from  $X[1]$  to  $i$  and  $Y[1]$  to  $j$  and so, if  $X[i] = Y[j]$  then  $C[i, j]$  will follow this recursive formula otherwise, it will follow this recursive formula maximum between these 2. Here also we have 1 index less in  $i$  or  $j$ . So, whichever is the maximum? So, this formula will give us a design technique and that is the dynamic programming technique we will come to that so, but before that let us try to prove this formula.

Let us try to prove this. So, we will prove this part and remaining part will be similar. So, we want to prove that  $C[i, j]$  is equal to  $C[i-1, j-1] + 1$  if  $X[i] = Y[j]$ .

(Refer Slide Time: 16:27)

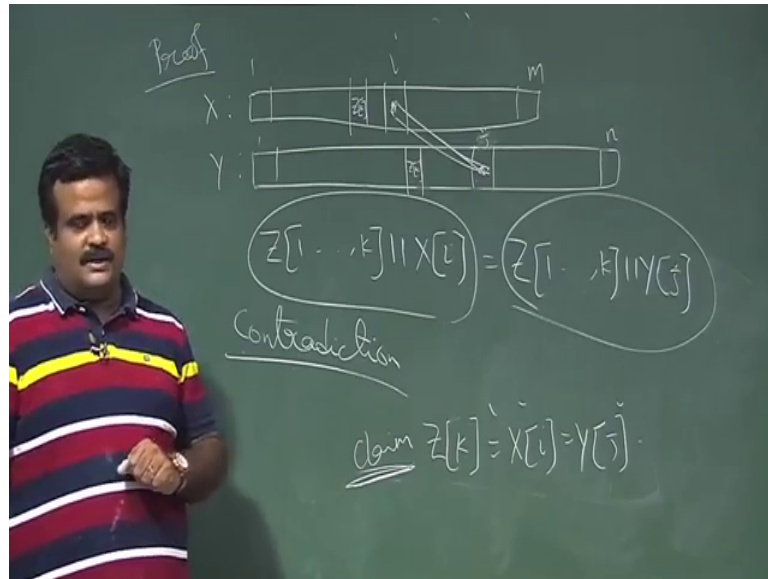


So, let us prove this. So, suppose this is our X sequence; this is our X sequence and this is our Y sequence and X is starting from 1 to m and Y is from 1 to n and suppose this is i and this is the j this is j and in this case, these 2 are same the value over here is same; this to a same value. So, X i is basically Y z. So, if X i is equal to Y j we have to prove C i j is equal to C i minus 1. So, we need to prove C i is equal to C. So, let us let K be the length of the longest common subsequence between these 2. So, LCS of X 1 to i and Y 1 to j and suppose this is we denote by j. So, this is the basically j 1 to K is basically LCS of X 1 to i comma Y 1 to j let. So, we length up the longest common subsequence is K and suppose j is the one such subsequence common subsequence between X 1 to i j 1 to j.

Now, our claim is the X j is this is our claim i, sorry, j K is basically this common value this is our claim we need to prove this, we need to prove this we need to prove that that X K is basically sorry j K the last term last common alphabet is basically X x i and Y j. So, how to prove that suppose it is not; suppose it is not; suppose the j is j K is not these 2. So, j K will be somewhere here.



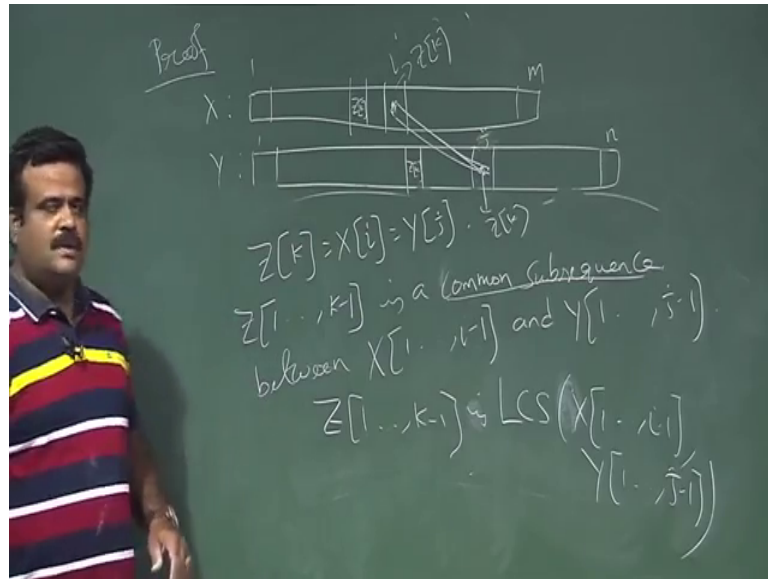
(Refer Slide Time: 19:22)



This is the  $j$   $K$  which is the common alphabet between  $X$  and  $Y$ . So, this is the  $j$   $K$ . Now if we just take this one. So, this is these 2 are same. So, we just take one to  $K$  concatenate with  $X$   $i$  which is same as this is a subsequence in  $X$  which is same as  $j$  1 to  $K$  concatenate with  $Y$   $j$  because  $X$  is  $Y$   $j$  now. So, this is a common subsequence between these 2 are same. So, this is a common subsequence between  $X$  up to  $j$  and  $Y$  up to a  $X$  up to  $X$   $Y$  up to  $j$ .

So, this is telling us the length of the longest common subsequence is  $K$  plus 1 which contradicts that length is  $K$ . So, this is the contradiction this is the contradiction because we assume length is  $K$ , but here we are getting length is  $K$  plus 1. So, that is the contradiction so; that means this is true. So, that means,  $X$   $Z$   $K$  is basically the last term. So,  $Z$   $K$  is last term.

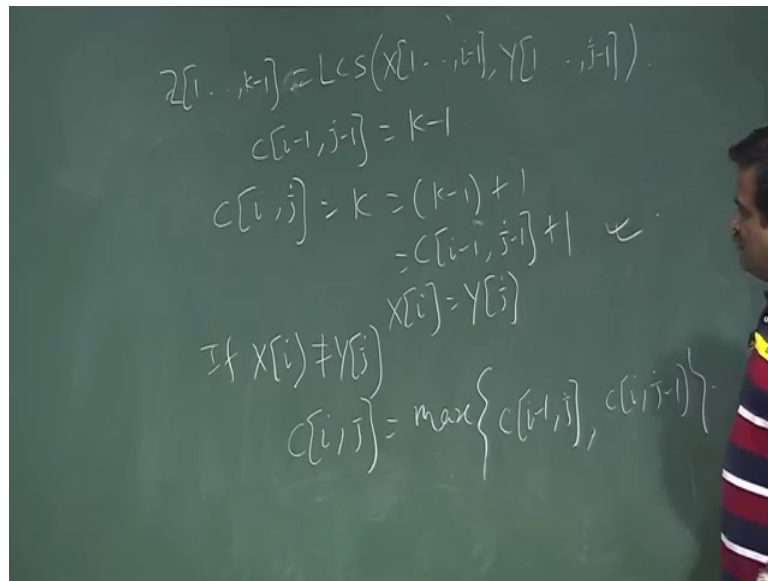
(Refer Slide Time: 20:53)



Now we take Z up to. So, Z K is basically X i which is basically same as Y Z. So, now, we take Z 1 to K minus 1. So, you just remove the last term. So, this is a common subsequence common subsequence between X up to i minus 1 and Y up to j minus 1 because we are just removing the last part. This last part is basically our it Z K. This is basically our Z K which is same as this. So, we are removing this last part, then this is a common subsequence of these 2 not only common subsequence.

We can claim that this is a longest common subsequence why. In fact, this is a LCS of X 1 to i minus 1 comma Y 1 to j minus 1, why so, because suppose this is a common subsequence. Now suppose this is; if this is not a longest common subsequence, this is of length K minus 1; that means, there is a subsequence of length K which is more than K minus 1, if there is a subsequence of length K, then we can add these 2. So, which will give us a common subsequence of up to i X up to j which is of length K plus 1. So, again which contradict the fact that K is the long length of the longest common subsequence between X and X up to Y; Y up to j so; that means, this is true. So, Z is a longest common subsequence of this so; that means the length of the longest common subsequence.

(Refer Slide Time: 23:11)

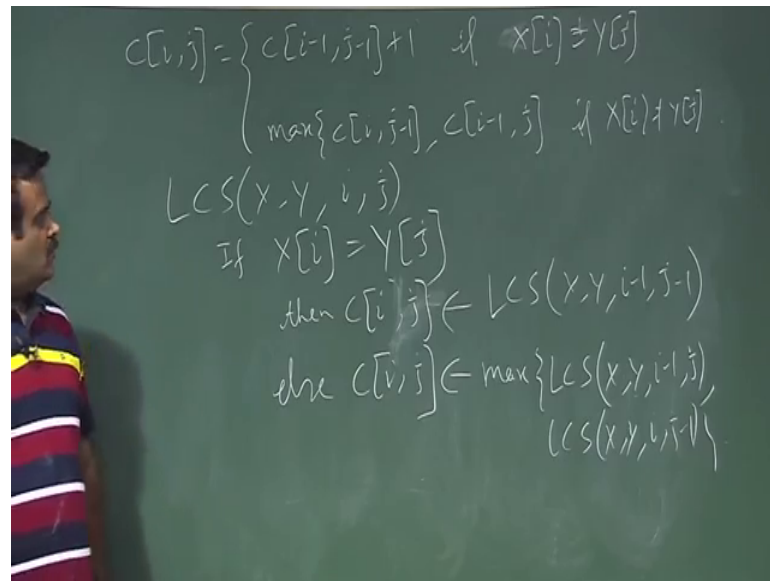


So, Z is basically one; this is basically a longest common subsequence a longest common subsequence of X up to i minus 1 and Y up to j minus 1 so; that means, length up the longest common subsequence of this i minus 1 j minus 1 basically K minus 1 so; that means, C i minus 1 j minus 1 is basically K minus 1. So, this we establish; now what is C i j? C i j is basically C i j is K which is basically K minus 1 plus 1 which is basically C i minus 1 comma j minus 1 plus 1.

So, this is the first part of the theorem and the second part will go similar way. So, second part means if this is the proof when X i equal to Y Z and the second part is if X i is not equal to Y j then we have this formula C i j is maximum of one index list C i minus 1 comma j comma C i comma j minus 1 and this prove we can for similar way one can argue this proof. So, this is the proof of this.

Now, this will give us a recursive formula. So, let us start with the; so, this is one of the hallmark of dynamic programming technique.

(Refer Slide Time: 25:24)


$$C[i, j] = \begin{cases} C[i-1, j-1] + 1 & \text{if } X[i] = Y[j] \\ \max\{C[i, j-1], C[i-1, j]\} & \text{if } X[i] \neq Y[j] \end{cases}$$
$$LCS(X, Y, i, j)$$

If  $X[i] = Y[j]$   
then  $C[i, j] \leftarrow LCS(X, Y, i-1, j-1)$   
else  $C[i, j] \leftarrow \max\{LCS(X, Y, i-1, j), LCS(X, Y, i, j-1)\}$

So, basically we have this result  $C[i, j]$  is basically  $C[i-1, j-1] + 1$ , if  $X[i] = Y[j]$  and  $C[i, j]$  is maximum of  $C[i, j-1]$  or  $C[i-1, j]$ . So, this will give us a recursive algorithm. So, this is basically LCS of  $X$  comma  $Y$   $i, j$ . So, if  $X[i] = Y[j]$ , then  $C[i, j]$  is basically sorry  $C[i, j]$  is basically LCS of  $X$  comma  $Y$   $i-1, j-1$ , else  $C[i, j]$  is basically maximum of this is the length of the LCS  $X$   $i-1, j$  comma  $LCS$  of  $X, Y$   $i, j-1$ . So, this is a recursive formula recursive algorithm. So, the worst case will be when this is not equal to in that case, we need to compute this both term and the size is all is only in the one index reduce.

So, we will continue this in the next class, where we will we will talk about how we can use this recursive formula to have a dynamic programming technique. So, you have a hallmark for dynamic programming technique. So, we will talk about this in the next class.

Thank you.