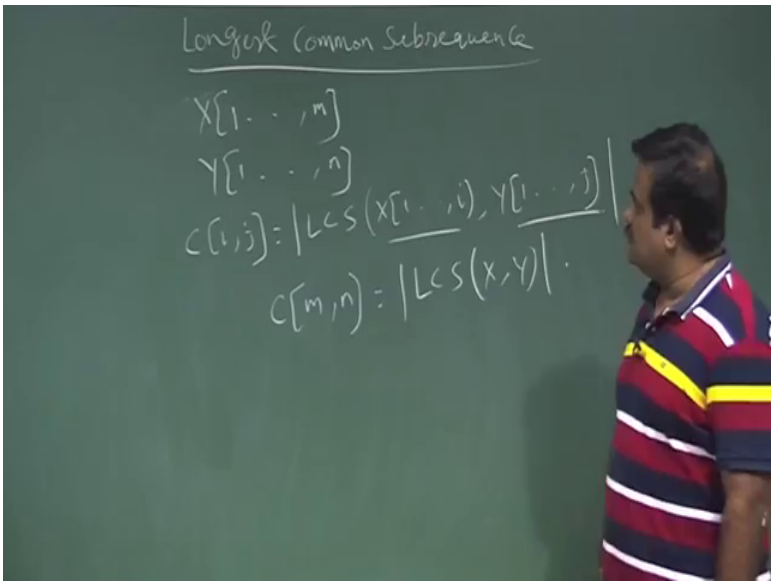
An Introduction to Algorithms
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 37
Longest Common Subsequence

So we are talking about longest common subsequence. So, we will just to recap. So, we have given 2 Sequence x and y.

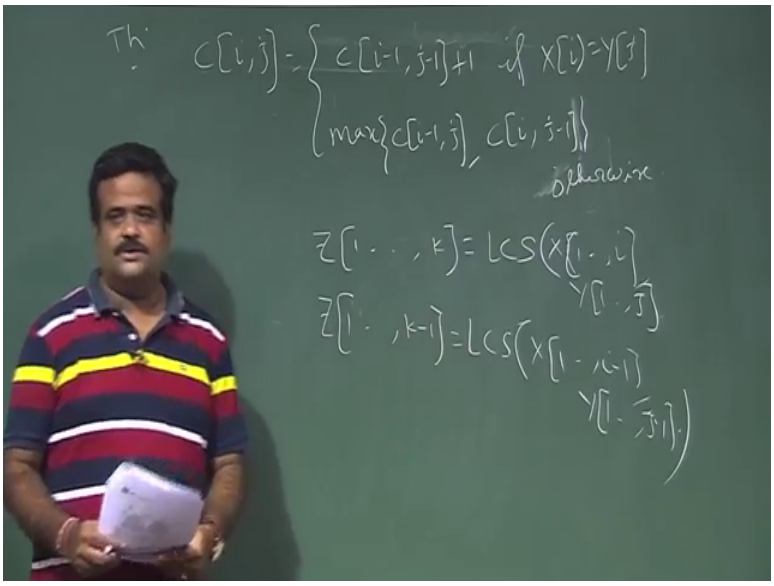
(Refer Slide Time: 00:24)



You have given 2 sequence one is of length m, other 1 is of length l. And you need to find a longest common subsequence between this. So, for that we have if we simplify this problem to finding the length of the longest common subsequence, and we have defined c i j which is basically prefix is length of the longest comma subsequence of a x 1 to i and y 1 to j and we have. So, this is the length of the longest common subsequence up to prefix 1 to i after prefix 1 to j.

Now, we can find out c i j for I all i j then we are done because c m n length c m n is basically length of the longest common subsequence of x and y. So now, we in the last lecture we have seen a recursive formulation for this c i j. So, that is basically c i j.

(Refer Slide Time: 01:36)

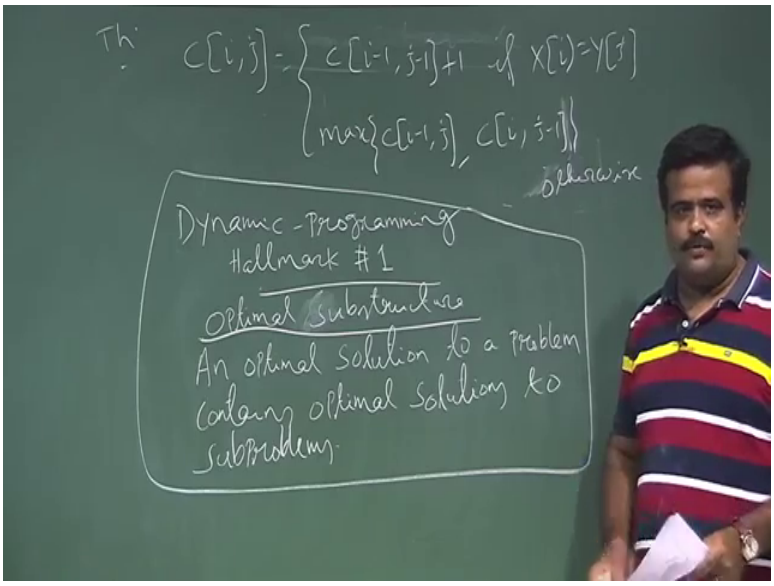


So, that is basically c i j is c i minus 1 j minus 1 plus 1 if x i can to i j and it is basically maximum of c i minus 1 comma j comma c i j minus 1 c i j minus 1. Where otherwise if x i is not equal to x j. And we prove this part and to prove this part we have taken a longest common subsequence z 1 to k which is a longest common subsequence between x 1 to i comma 1 to j. And then we have seen that any subsequence.

So, this is also a 1 to k minus 1 is a longest common subsequence between x 1 to i minus 1 y 1 to j minus 1. So, this is one of the hallmark of dynamic programming problem. So, for dynamic programming problem basically you have 2 hallmarks this is the first hallmark. So, this is telling us a of a solution of a problem content the solution of the sub problems and optimal solution to a problem contain the optimal solution of the sub problem. So, this is a longest common subsequence up to k. So, it is up to k minus 1 is longest common subsequence of this ok.

So, this is a hallmark of dynamic programming problem.

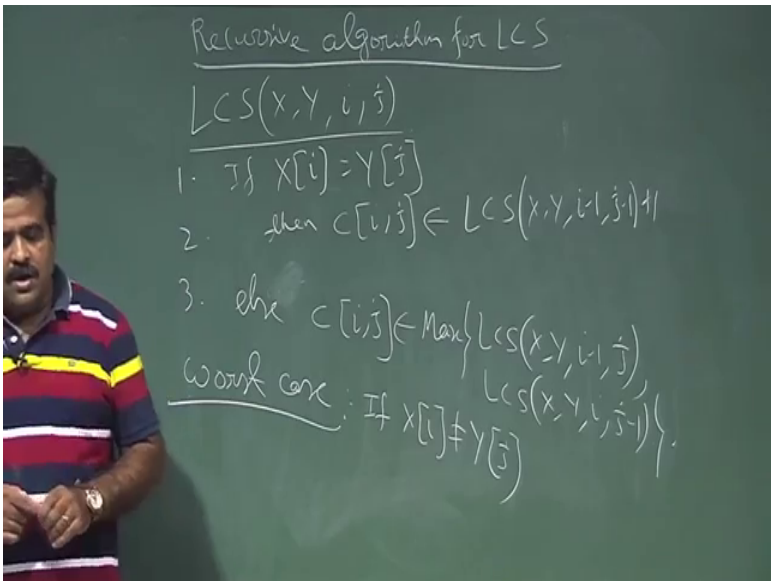
(Refer Slide Time: 03:30)



So Let us write this dynamic programming. So, programming hallmark this is hallmark number 1. So, what it is telling it is telling the optimal substructure the optimal substructure. So, it is telling an optimal solution an optimal solution to a problem content contains optimal solutions to sub problems. So, like if z up to k is a LCS then any subsequence any sub problem in his if we take j up to k minus 1 that is the optimal solution that is the LCS of x up to in k my i minus 1 y up to j minus 1.

So, this is one of the hallmark for dynamic programming problem. So, if we have such a hallmark then maybe you have to be fix our mind that maybe you have to go for dynamic programming approach. So, this is one of the hallmark. So, you have second hallmark also to reach that let us just write a algorithm for this so recursive formula. So, this is the LCS algorithm, recursive algorithm.

(Refer Slide Time: 05:52)



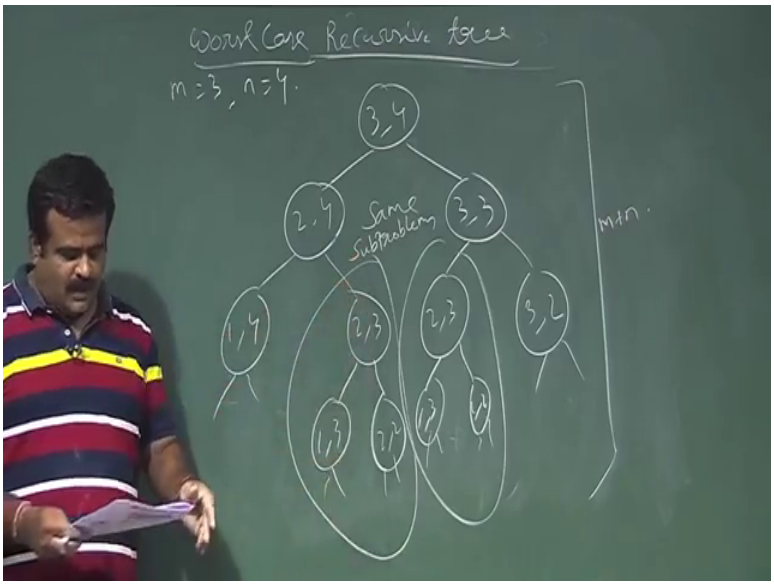
Recursive algorithm for LCS. So, basically we have finding the l s c of x y i j this is basically length of the longest common subsequence c i j.

So now if x i equal to y j then we know it is basically then it is c i j is basically l s c of I mean length of LCS of x y i minus 1 j minus 1 plus 1 this is the this case else. So, else 2 3 else we have c i j is max of max of the length of this 2, LCS of x y 1 index less and other way i j minus 1. So, this is the this is the algorithm for finding the c i j this is the recursive call. I mean now what is the worst case for this algorithm? You want to analysis the worst case of this algorithm.

So, when is the worst case. Worst case is if worst case will happen if x i is not equal to y j. Because in that case we have 2 call 2 call then you have to take the maximum. And also size if x y equal to x j then we have only one column size is also both the index reduced by 1, but if we have if xi is not equal to x j then you have to take we have to get we have to have 2 call recursive call, and then we need to get the maximum of that. And even the both the call we have only one index less. So, this is the worst case. So now, draw the worst case recursive tree for this for a some example some values of m and n.

So, let us draw the worst case recursive tree.

(Refer Slide Time: 08:50)

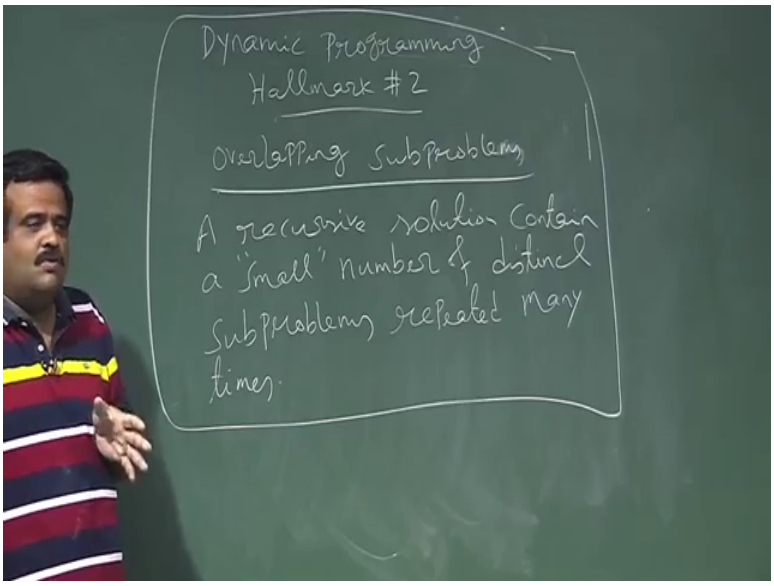


So suppose m is 3 n is 4. And we are in worst case. So, worst case means we have to go for x i is not equal to x j. So, give a call with m 3 m 4. So, since it is worst case we have to we have 2 calls which one index less. So, 2 4 then 3 and then after calculating this c 2 then we take the maximum. Again we are in worst case. So, this is index is 1 1 4 and this is 2 3 again here 2 3 3 2. So, like this, again here 1 3 2 2 again here 1 3 2 2 like this ok.

So now this is common this as. So, this tree is common. So that means, we have same sub problems, we have same sub problems over here. Because there common I mean. So, unnecessary we are doing the work double basically. So, too said that we need to have we need to memorize that I may whether we have calculated, that if you have calculated corresponding c 1 3 then we will not calculate again for this 3. So, there are many repetition over here. So, this is the what is the length of this height of this trees m plus n. This is the height of this tree. So, this is another hallmark of the dynamic programming problem. This is the many repetition of the overlapping sub problems. So, this is the repetition. So, let us write that second hallmark of dynamic programming problem.

So, this is basically telling us overlapping sub problems.

(Refer Slide Time: 11:24)

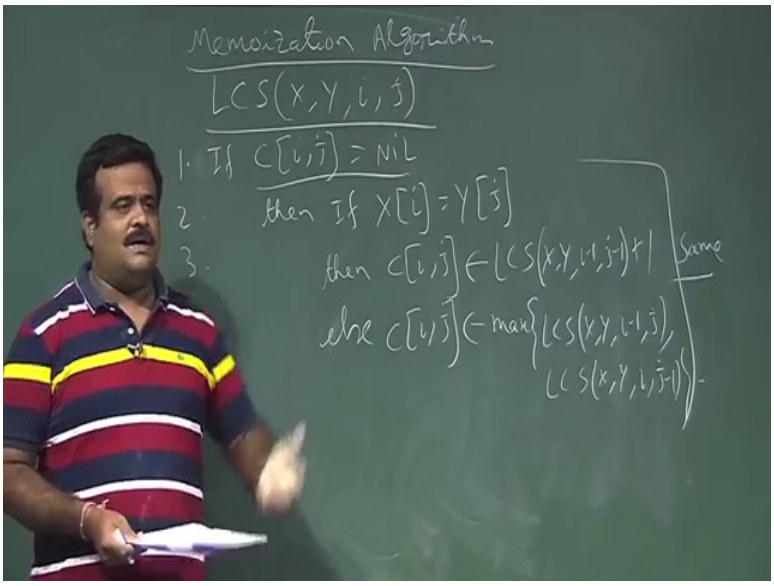


So dynamic programming this is hallmark number 2. And this is telling us overlapping sub problem, over lapping sub problems sub problems. So, what it is telling us? It is telling so a recursive solution contained a small number of a distinct sub problems a repeated many times a repeated many times. So, this is this is one of the hallmark for dynamic programming problem like we if we. So, we have many overlapping sub problems.

So, if we have. So, that is that is one indication that now we must go for the dynamic programming technique. So, this is the second hallmark. So, if we see that our we have some recursive for formula and they are we have this type of overlapping sub problems. So, many repetitions are happening, then we must be ready for now it is time for go for the dynamic programming technique. So now, let us let us modified that algorithm which recursive algorithm to avoid this repetition. So, that is the memoization. So, we will memorize the value which we have calculated. So, if you have calculated some value we will memorize that we will not re calculate again.

So, that is the memoization algorithm of that finding LCS.

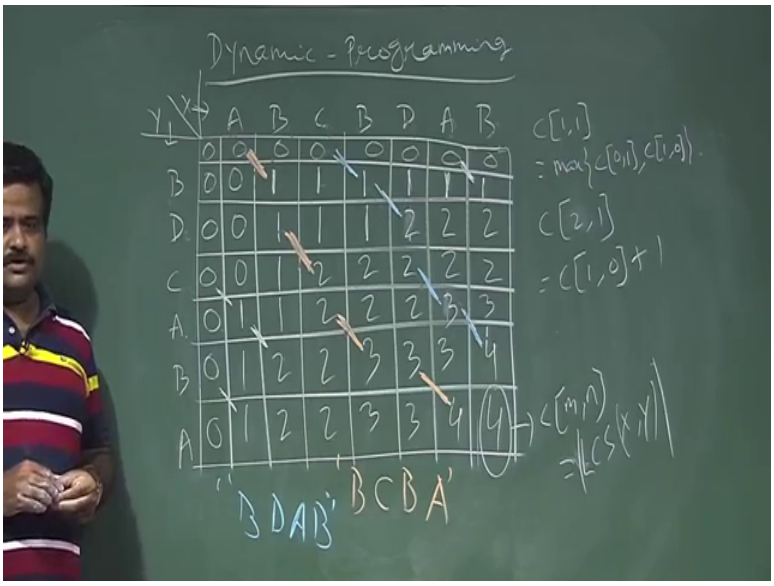
(Refer Slide Time: 14:11)



So, let us just Memoization not memorization memoization algorithm for finding the LCS. So, the remaining part is same all the thing if we have already calculated some value will not re calculate again so; that means, if c i j is nil is nil; that means, it is not calculated the only we go for calculating it yeah. So, if this then we will go for then if then remaining part is same this, then c i j is basically length of x y i minus 1 j minus 1 plus 1 else c i j is basically max of this 2 length of i minus 1 comma j.

So, this part is same as earlier only thing we have a check point over here to avoid the recalculation. If we are calculated the value will not going to recalculate again, that is why it is memoization we memorize that if we also calculated we will not be calculated. So, that you will give us the dynamic programming technique. So, it is basically the bottom of technique. So, it is a tabular technique. So, we will just try to find put this c i j's.

(Refer Slide Time: 16:31)



So, it is basically dynamic programming approach. So, idea is basically computing that table bottom of.

So, let us just take that example. So, you have the x value and y value. So, A B C let us take a x sequence and y sequence D A B. So, this is our x sequence and we have a y sequence B D C A B A B D C A B A. So, we just take this table like this. So, this is a tabular method. So, let us so, this is this we put in a table. So, this is our x sequence starting from here, this is our x sequence and this is our y sequence. And we want to calculate this c i j's c i j is basically the bottom of way we will calculate this ok.

So, these are all 0 because this y is not started yet, and there is no common thing. So, these are all 0 this means are all zeros. Now we will calculate say this field. So, this is our so, for this field we need to take these and these value. So, if you take these and these value. So, this is basically x c 1 1. This is basically c 1 1 c 1 1 is basically. So, it depends on x 1 y 1. X 1 is a y 1 is B. So, if you use the formula they are not same. So, it is basically maximum of c 0 1 comma c 1 0. So, these 2 maximum of these 2 these 2 are both 0. So, it will be 0 ok.

Now, this is basically c. So, x is 2 c 2 1. Now x 2 x 2 is b x y is. So, they are same. So, they are same means the formula is basically c 1 0 plus 1. So, this plus 1. So, this will be 1 like this. So, similarly we will take these 2 these 2 are not same if these 2 are not same then this will be maximum of these 2. So, maximum of these 2 is 1. Then again we have

this 2 is same. So, it will be this plus this plus 1 So 1. So now, we take these 2 these 2 are not same maximum of these 2 is 1, then we take these 2 it is one not same maximum of these also 1 and we take these 2 these 2 are same. So, it will be this plus 1 1.

Now, again here. So, we compare these and these 2 are not same maximum of this 2 is 0, this and this maximum of this 2 is 1, these and these are not same maximum of these 2 is 1. So, basically you are using that recursive formula. But this is the tabular method if we have calculated that c i j we have not recalculating is in. So, that is the memoization and this is the dynamic programming technique. So now, these 2 these 2 was also not same. So, it will be maximum of these 2 1. Now these 2 are same. So, once these 2 are same it will be this plus 1. So, this will be 2, that recursive formula. Again these 2 are not same. So, maximum of these 2 to these 2 are not same maximum of these 2 2 like this.

So, let us just fill it. So, this is not same 0 this not same one this same. So, this is 2. These 2 are not same maximum of this 2, to this 2 are not same maximum this 2 these 2 are not same 2 these 2 are not same 2. Again this 2 a this these and this is 1 they are same. This is not same this is 1 maximum of these 2 this 2 is not same maximum of this 2 not same maximum of this 2 not same maximum of this 2 these and these are same. So, it will be this plus 1. So, 3 sorry 2 plus 1 3.

So now these and this these and these are different. So, it will be maximum of these 2 3. So, we continue like this. So, this these 2 are not same maximum of these to 1 these 2 are same it will be 2 these 2 are not same maximum of these 2 these 2 are same it will be 3 this 2 are not same. So, 3 this 2 are not same again 3 now these 2 are same. So, it will be 4 this plus 1 it will be 4. Now again these 2 are same. So, it will be 1 these 2 are not same maximum of these 2 2 not same 2 not same 3 maximum of these 2 3 again these 2 is same. So, you have these 2 is basically same we have 4 and these 2 are not same maximum of these 2 4.

This is basically a one c in comma n, this is basically the LCS of length of the LCS of x comma y for the full sequence. And this is what we are looking for. We need to you want to find out c m n. So, this is basically c m n. And this is the dynamic programming technique this is the bottom of way. So, if you have calculated the c i j's will not re calculate again. So, this is the memoization. So, we will we will store this value, and we use this value to get the this thing. So, this is the length of the longest common

subsequence and length of the longest common subsequence is 4 for this 2 sequence x and y ok.

Now, the question is from here how we can find the longest common subsequence because we have simplified that problem. We need to find the longest common subsequence. So, we reduce the problem into to find the length of the longest common subsequence. Now you are interested after getting the length you are interested to find a longest common subsequence. So, for that what we need to do? We need to follow this path. So, this path we need to follow. So, maybe we can take this one, then we can take this one, we can take this one. And we can take this one. So, this will give us what this will give us which sequence this will give us A. And then this will give us B. And this will give us C. And this will give us again B. So, this is a longest common subsequence is an if you follow this path this one you need if you follows this path, this one this one this one and this one.

So, this will give us what this will give us this is for B and this is A and this is D and this is again this is again B. So, this is also longest. So, if I follow following this path we can get the along x common subsequence. So, so we have to find a longest common subsequence we simplify this problem to find the length of the longest common subsequence. So, by this dynamic programming method we got the length after getting the length we just follow this path to get the longest common subsequence. So, what is the time complexity for this algorithm for this recursive algorithm? So, basically we have a if the length of this is m.

So, the time complexities. So, if x is of size 1 to m, and y is of size 1 to n. So, basically to compute this the time is basically order of m into n. So, this is the time complexity for p p at the table. So now, also p p at the table we can get the length of the longest common subsequent. So, this is the dynamic programming technique and we illustrate this technique by an example which is called longest common subsequence problem and. So, for any dynamic programming technique we should have such kind of recursive formula, and we should have 2 hallmarks.

So, in the recursive formula we should have optimal substructure hallmark. So, that is telling us if we have a solution of a problem, that should contain solution of the sub problems. That is the first hallmark of the dynamic programming technique. And the

second hallmark is overlapping sub problems. So, if we can see the many ah many sub problems are repeating basically, then we must think that now it is time to go for dynamic programming technique ok.

Thank you.