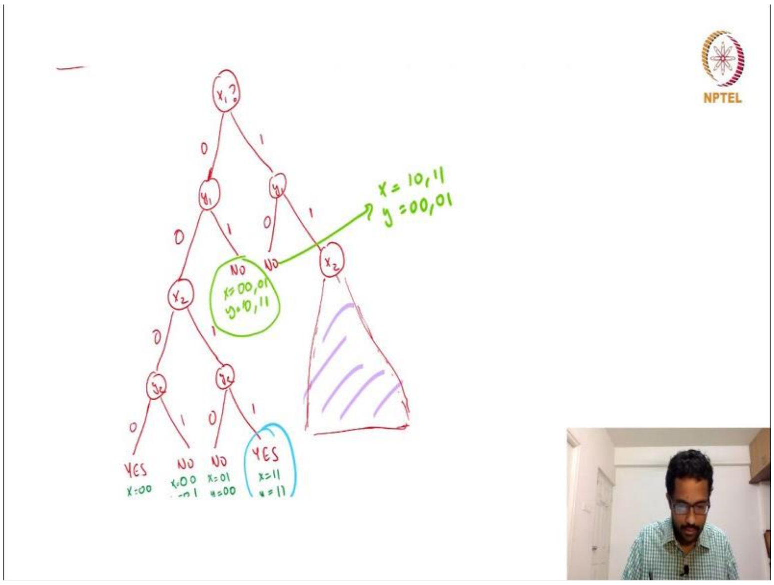
Computational Complexity
Prof. Subrahamanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Lecture - 55
Introduction to Communication Complexity: Part 2

(Refer Slide Time: 00:15)

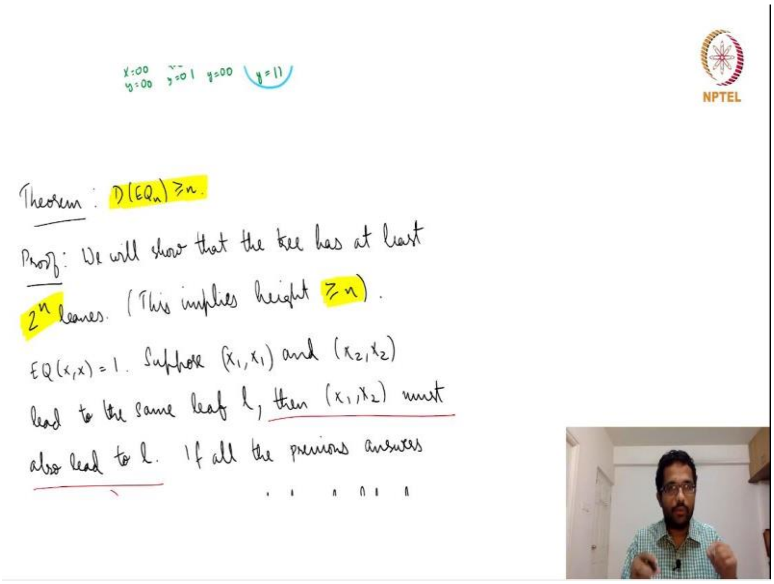


Hello and welcome to lecture 55 of the course computational complexity. In the past lecture we saw the communication complexity model and we saw a bunch of examples. And we define what is called the protocol tree and we saw a protocol for equality of 2 bit inputs and we saw the illustration of that protocol in the protocol tree model. So, now let us try to understand what is happening here.

As I observed in the previous lecture as we observed in the previous lecture some leaves have correspond to multiple input pairs. So, this particular no leaf has that is encircled with green colour has 4 input pairs that reach here. There are some other leaves like this yes input yes leaf has only one input pair that reaches here. So, which input pairs which where and how are they divided.

So, one thing must be clear any specific input pair it reaches only a specific leaf it cannot go to 2 different leaves. Because each of these questions has a deterministic question and have deterministic answers.

(Refer Slide Time: 01:40)



So, one specific input pair has follows a certain path it cannot go into two parts, that possibility is not there. So, what we will do is to first see so I already explained I already saw a proof that deterministic communication complexity of equality is at most n + 1, which was the n + 1 bound was the trivial bound where Alice sends everything to Bob and Bob replies by 1 which is applicable for any function.

And now we will see that this bound is actually very close to being optimal or in fact it is optimal. So, we will see a proof the deterministic communication complexity of equality is actually at least n, so how will we see that we will show that any protocol tree for equality has at least 2 power n leaves. So, in this tree we have 4 leaves at the bottom here and another set of 4 leaves in this red triangle.

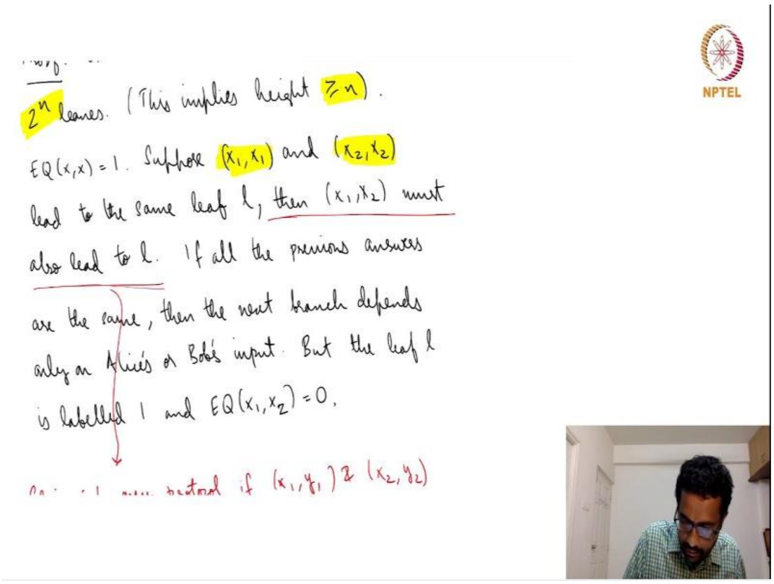
So, that is 8 and then 2 leaves from the top. So, 10 leaves there are 10 leaves and to build a tree with at least 10 leaves you need to have a tree of height at least 4, you need to height at least 4. So, 10 leaves means at least 4 height so that gives you a bound on the height a lower bound of the height and which is the same as the lower bound on the communication complexity. So, we will show that the tree that has at least 2 power n leaves.

If it computes equality so consider any input pair where Alice and Bob get the same input let us say x and x this is = 1. So, the function has to return 1. Suppose Alice and Bob get x 1 x 1 and Bob and Bob get sorry Alice and Bob get x 1 x 1 that is one possibility and Alice and Bob get both x 2 x 2. The claim is that these two input pairs x 1 x 1 and x 2 x 2 these two input pairs must go to 2 different leaves.

So, here in this picture if you see that the green leaf here that has 4 combinations 4 input pairs that but that was a no leaf. What we are saying here is that any yes input combination any yes input pair must go to 2 different leaves and we know that there are 2 power n possible binary strings of length n. So, there are two power n possible yes input pairs so starting from all 0 to all to all 1s that Alice has all 0 input Bob has all 0 input.

Let us say Alice has 0 0 0 0 1 Bob has 0 0 0 0 1 up to all the all 1 input going to both Alice and Bob. So, there are 2 power in input pairs which return yes or which return 1 as the correct answer. What the claim is that all of these 2 power n input pairs have to go to distinct leaves, no none of these pairs can go to the same leaf. So, that means that the tree has to have 2 power and leaves to accommodate these 2 powers and yes input pairs.

(Refer Slide Time: 05:00)

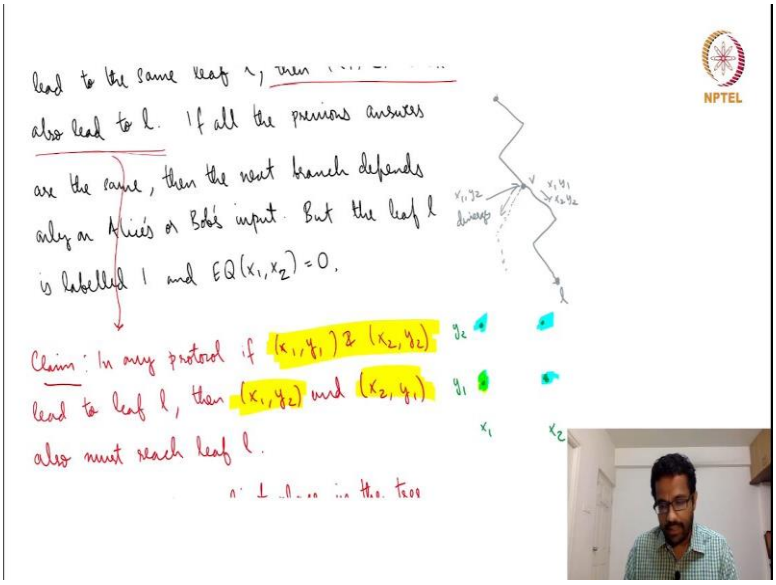


So, because of which the height should be at least n so let us see why that is the case suppose the pair x 1 so suppose x 1 and x 2 are 2 distinct values x 1 and x 2 are not the same value. So, suppose x 1 x 1 and x 2 x 2 they end up at the same leaf, the claim is that if this happens then the input pair x 1 x 2 also must go to the same leaf. But then that leaf answers yes, because x 1 x 1 the answer is yes, any leaf is a place where you output something.

So, the leaf answers yes but then x 1 x 2 is 2 inputs. So, it is not equal they are not equal we are computing the equality function so when x 1, x 2 when it goes to yes leaf that is an incorrect answer. So, this red underlined part is the is a key observation, if x 1 x 1 and x 2 x 2 go to the same leaf then x 1 x 2 must also go to the same leaf but then that is not correct. So, that means that each of each pair x 1 x 1 x 2 x 2 everything must go to distinct leaves.

Which means all the input pairs that answer yes must go to distinct leaves which means ah there are at least two power and leaves which means the height has to be at least n. So, now let us try to see why x 1 x 1 and x 2 x 2 must go to the same leaf or sorry different leaves.

(Refer Slide Time: 06:34)



In any protocol that computes equality so this whole thing is for equality. If x 1 y 1 this this particular claim is actually not for equality in any protocol computing any function. Suppose if x 1 y 1 and x 2 y 2 go to the same leave, it is possible that x 1 y 1 and x 2 y 2 go to go to the same leaf then the claim is that x 1 y 2 and x 2 y 1 also must go to the same leaf. So, in other words so you can think of some coordinate kind of setup.

Let us say this is x 1 this is x 2 this is y 1 and this is y 2. So, x 1 y 1 is this input maybe I will highlight with blue, this input x 2 y 2 is this input. So, now if these two input pairs go to the same leaf, then it means that the other inputs x 1 y 2 and x 2 y 1 also must go to the same leaf this is the claim. So, all these 4 must go to the same leaf if 2 diagonally opposite points go to the same leaf then the other diagonal the points in the other corners also must go to the same leaf, y is this the case.

So, suppose x 1 y 1 and x 2 y 2 go to the same leaf. So, let me draw a picture suppose x 1 and y 1 something this is the leaf value. Now suppose now we want to show that x 1 y 2 also goes to the same leaf suppose not x 1 y 2 diverges and goes somewhere else. Let us say this point that where I have drawn from a dotted line this is the point of divergence of x 1 y 2, x 1 y 2 diverges here.

So, why did x 1 y 2 diverge here? So, till this point x 1 x 1 y 1 x 2 y 2 and x 1 y 2 travel together to reach this point let us say this point is v. At point v they diverged y did they diverge, so this means that now what could have happened at this point. This means at this point whoever had responded maybe Alice spoke or Bob's spoke, Alice could if let us say Alice is the 1 that spoke.

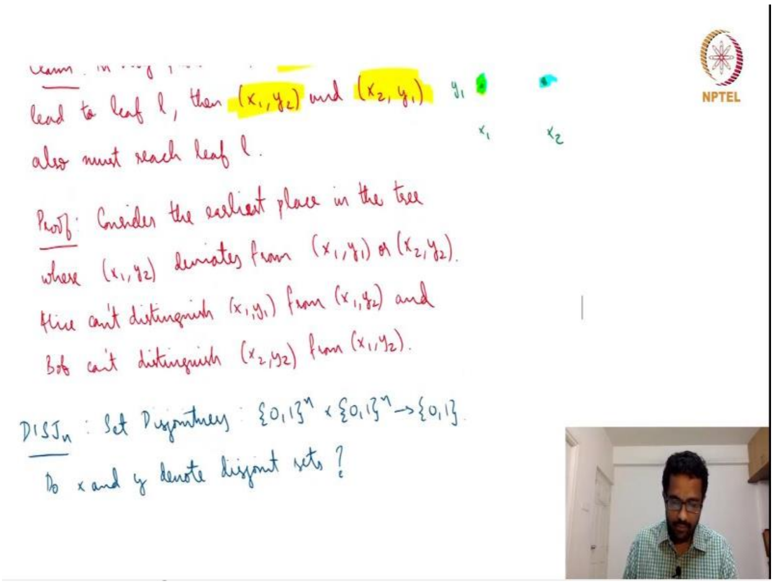
Suppose Alice spoke means Alice gave a different answer for x 1 y 2 from x 1 y 1 or x 2 y 2. So, x 2 so x 1 y 1 and x 2 y 2 go this way go to the and x 1 y 2 goes to the left. So, suppose Alice was speaking here at the at the vertex v now suppose Alice so all that Alice knows is that she said or they are at v Alice and Bob together are at v. If it was x 1 y 1, she would have gone to the right.

If, because it is x 1 y 2, she goes to the left, but then Alice does not know what is Bob's input. Except for the fact that whatever Bob's input together with Alice’s input they managed to reach the point b. Apart from this Alice has no other information on Bob's input. The only thing that she knows that Bob's input made them reach the point v. So, at this point x 1 y 2 and x 1 y 1 these 2 scenarios Alice does not have the power to distinguish because all she knows is her input is x 1 and that they are at the vertex v.

She does not know what she cannot distinguish x 1 y 1 and x 2 x 1 y 2. So, it is not possible that Alice is the one speaking here. So, suppose Bob is the one speaking here now. I can make the same argument with x 1 y 2 and x 2 y 2. Now Bob only knows that his input is y 2 and that he and that they are together at v. Whether Alice’s input is x 1 or x 2 cannot be distinguished by Bob so he cannot distinguish x 2 y 2 from x 2 y 1 sorry x 1 y 2 x 2 y 2 and x 1 y 2 he cannot distinguish the two.

So, it is not possible for him for Bob to be the one Bob to be speaking here because if Bob was speaking here. He cannot distinguish and he cannot give two different answers for the situation x 1 y 2 and x 2 y 2.

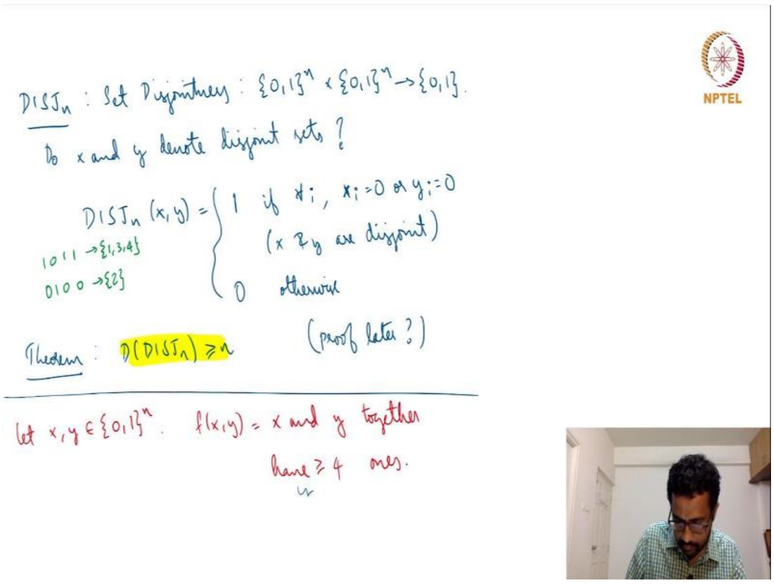
(Refer Slide Time: 11:36)



So, this means that there is no possibility that x 1 y 2 diverges from the path taken by x 1 y 1 and x 2 y 2. So, if x 1 y 1 and x 2 y 2 go to the same location x 1 y 2 must also go to the same location and same thing applies for x 2 y 1 it is a very symmetric argument. So, that completes the claim and that completes this proof that all the distinct pairs x 1, x 1 x 1, x 2 x 2, x 3 all the pairs that are yes inputs pairs must go to distinct leaves.

Because x 1 x 2 is not a yes input pair but if x 1 x 1 and x 2 x 2 put the same leaf then x 1 x 2 must also go to the same leaf. So, this means that the height of the tree has to have 2 power n leaves and that means the height has to be at least n. So, this shows that the deterministic communication complexity of equality is at least n.

(Refer Slide Time: 12:43)



So now let us see another function. This is called set disjoins we are actually measuring whether two sets are disjoint or not. So, we have already seen examples in the previous lecture where we have seen the correspondence between n bit vectors and subsets of 1 2 3 up to n. So, let us say Alice has the input maybe I have right here Alice has the input 1 0 1 1 which corresponds to the set 1 3 4.

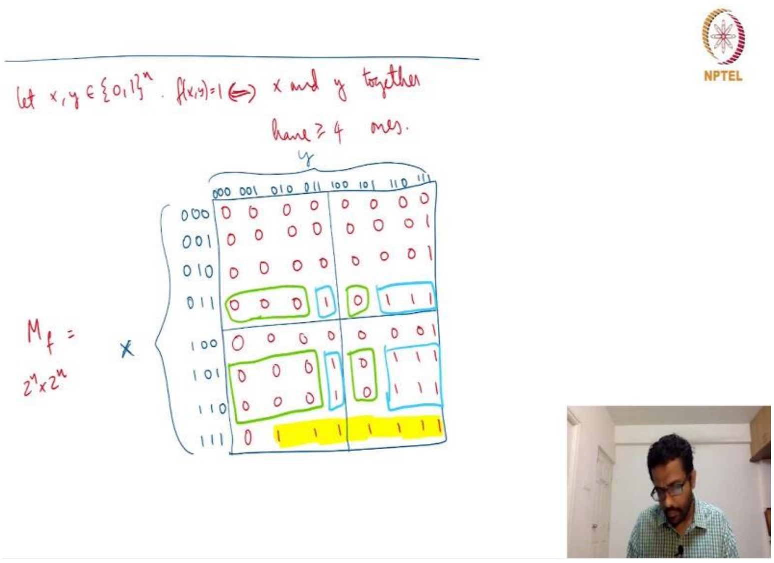
Let us say Bob has the input 0 0 1 0 which corresponds to the input which corresponds to the set containing just 3 and what we want to measure is whether these two sets are disjoint. Obviously, this is not disjoint because both of them contain the element 3. So, disjointness is one disjointness is one if the sets are disjoint or in other words in terms of bit positions for all i for all indices i either Alice's x i should be 0 or Bob's input y i should be 0.

And 0 otherwise, otherwise meaning there should be some index j for which x i x j and y j should both be 1. So, in this case the third index so suppose Bob's input was this which corresponds to the set two now this means that in all the indices one of the Alice's input or Bob's input is 0. So, this is disjoint. So, to put it very simply if the sets corresponding to Alice's input and Bob's input are disjoint then then output is 1.

If the sets corresponding are not disjoint the output is 0 and disjointness happens to be a fundamental or rather canonical kind of problem in the case of communication complexity. So, there are multiple settings like randomized deterministic and with some public coins private coins and so on, but in all of this disjointness plays a key role in trying to understand the complexity of the model.

So, we will see disjointness to some detail so that is why it is important to I decided to stated in a very formal manner. And just like equality the deterministic complexity of disjointness is also at least n. So, we will see the proof a bit later actually maybe it is not no proof maybe proof later.

(Refer Slide Time: 15:35)



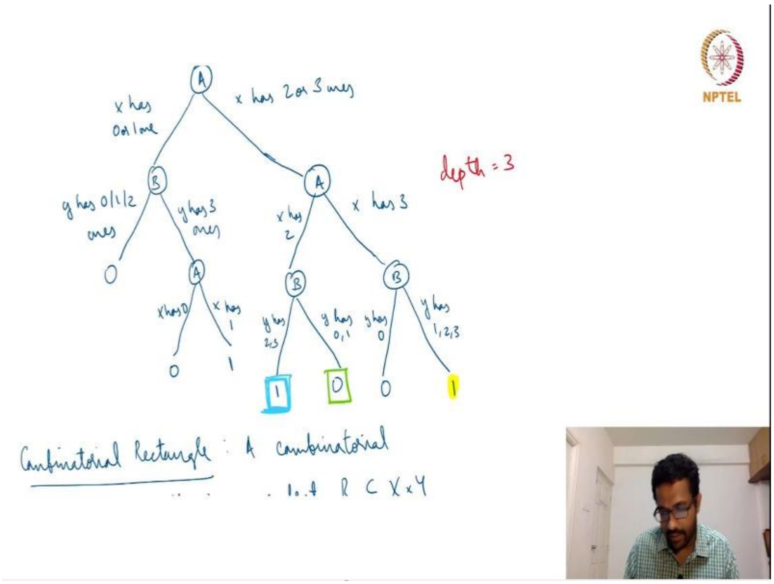
Before that I want to define an important notion called combinatorial rectangle. So, this is coming from the understanding of what is happening in a protocol tree. So, consider this function x and y are n bit vectors and f of x y is saying whether x and y together have at least 4 1s. So, maybe x f x y, I just state it bit differently f of x y = 1 if and only if x and y together have at least 4 ones.

So, in this situation or in any situation we can define what is called the matrix of the function. So, I will call it let us say M subscript f this is a 2 power n by 2 power n matrix as you can see what is happening here. So, where the rows are indexed by Alice's input x and the columns are indexed by Bob's input y and the let us take this the first entry here this entry that i am encircling with the blue thing.

So, this is when this is and this is a function value when Alice has 0 0 0 and Bob has 0 0 0 0. So, together they have 0, 1 so obviously the function value is 0. Now consider this this one over here that I am highlighting with yellow this is corresponds to the output when Alice input is 0 1 1 and Bob's input is also 0 1 1. So, together they have four 1s so the answer so f is also 1 that is why this entry has 1.

And like that you can verify each and every entry of this matrix so it is just a function value for when x and y are given as a at the index.

(Refer Slide Time: 18:04)



So, now let us try to understand let us try to see a protocol and understand what is happening in this protocol. So, together Alice and Bob have to decide if they have four 1s at least four 1s. So, let us say the first at the top level Alice speaks, Alice says whether he has or she has her input has 0 or 1 1. If so, she goes to the left side or it has more than 1 1 so 2 or 3 1s. So, again this all of this is for the case when they have 3 bit inputs.

Even though I set n here let us say consider the case where they have 3 bit inputs even this matrix is for 3 bit inputs. So, that is why it is an 8 by 8 matrix. So, Alice says whether she has 0 or 1 1 or more than 1 1 2 or 3 1s now consider let us look at the left side. If Alice says 0 or 1 1 now if Bob has 0 1 or 2 1s that means that the maximum number of 1s they together have is 2 + 1 which is 3 and in this case the function value will be 0.

Because function value is 1 only when they have 4 1s. So, that is what I have indicated here at the leaf. The other case is when Bob has 3 1s that is the branch this branch over here. Why Bob has 3 1s? If Bob has 3 1s then Alice has 0 or 1 1 Bob has 3 1s. So, the function value really depends on whether Alice has 0 1s or 1 1. So, we asked that question and depending on that the function value is 0 or 1.

And similarly, you can trace the path on the right-hand side x has Alice has 2 or 3 1s and then we subdivide into further case when Alice has 2 and Alice has 3 1s. So, again notice that Alice herself is speaking even after saying whether she has 2 or 3 1s. And then depending on that then Bob has to classify his input whether together they have 4 or together they do not have 4 and basically one more response from Bob is enough.

So, together they compute the function and this tree has depth 3 so depth of this tree is 3. So, which means this function can be computed the deterministic communication complexity of this function is at most 3 less than or equal to 3. Well, there could be a better algorithm better protocol to compute this function but we do not know. So, let us try to understand what is happening maybe I will just remove the highlights.

Because it could be misleading to; what I am saying. So, now consider this 1 that is outlined here with blue this one. So, when do we reach here which input pairs reach this leaf? The input pairs that reach this leaf are those where Alice has 2 or 3 1s. And then further refined by Alice having 2 1s and then Bob has 2 or 3 1s. So, Alice has 2 1s and Bob has 2 or 3 1s. So, which; is actually corresponding to the cells that are marked here by the same kind of blue outline.

So, the 4th row 4th, 6th, and 7th rows and 4th 6th 7th and 8th columns 4th 6th and 7th row correspond to Alice having 2 1s 4th 6th 7th and 8th row columns correspond to Bob having 2 or 3 1s. So, exactly these entries that are outlined by the blue colour are the 1s that reach that leaf. Similarly, you can check this this particular leaf here that is outlined with the light green here Alice has 2 1s and Bob has 0 or 1 1.

And those values correspond to exactly the cells that are outlined with the green colour, where Alice has 2 1s and Bob has 0 or 1 1s. So, the 4th 6th and 7th row and first second and fourth columns sorry first second third and fifth columns of the matrix. First second third and fifth columns and fourth sixth and seventh rows. So, this is exactly marked with the green colour.

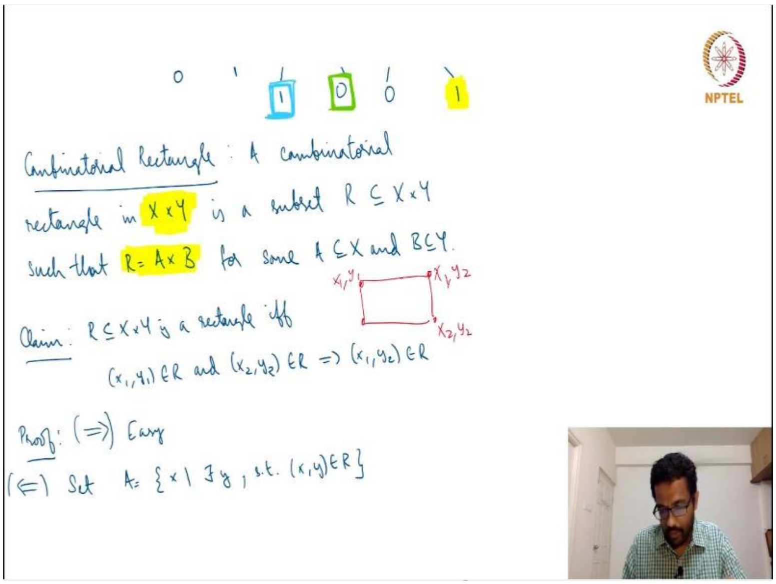
Similarly, this yellow highlighted the rightmost leaf happens when Alice has 3 1s and Bob has 1 2 or 3 1s. So, there is only one row corresponding to Alice having 3 1s which is the last row and all the columns except the first column correspond to 1 2 or 3 1s for Bob. So, that is exactly the set of highlighted entries in the last row. So, what is it that you can notice here? All the entries that are like the green outlined 1s reach a certain leaf.

And all the entries that reach that leaf or give the give the output 0. Blue outline once all the entries that reach that leaf give the output 1 all the entries that reach the yellow highlight give the output 1. So, everything that reaches a certain leaf gives us fixed output. Another thing is that all these are kind of very like there is a pattern here. So, now suppose I swap the 4th and 5th columns bringing the green parts together.

And then maybe there is one more row here if I swap the 4th and 5th rows as well. Then, all the green square green outline cells will be together as a rectangle and the same thing will happen for the blue entries. So, there is this notion of a combinatorial rectangle, combinatorial rectangle is nothing but you are picking a certain number of rows and some columns.

And you take the cross product of that and that is what is happening with the blue outline green outline and the yellow highlight here. And in fact, you can see that at every stage of the protocol and at every node in this protocol tree not just the leaf any node in this protocol tree you are always at a combinatorial rectangle.

(Refer Slide Time: 25:22)



So, I will define it formally and then we will go there. So, a combinatorial rectangle in x cross y x is Alice’s input domain and y is Bob's input domain is a subset that is equal to A cross B. So, combinatorial rectangle R is just a subset A cross B, so what I mean by that is so this is not a combinatorial rectangle. Because it is not a of the form A cross B if, so A has to be this this range and B will have to be this range.

And A cross B will include this part also this top corner also. So, this is how a combinatorial rectangle should look. But then it need not be contiguous meaning I could allow let us say B was this itself but A was in two pieces this one and this one. So, then the combinatorial rectangle would look like this and this or even B could be split. So, let us say there is some part of B that is over here.

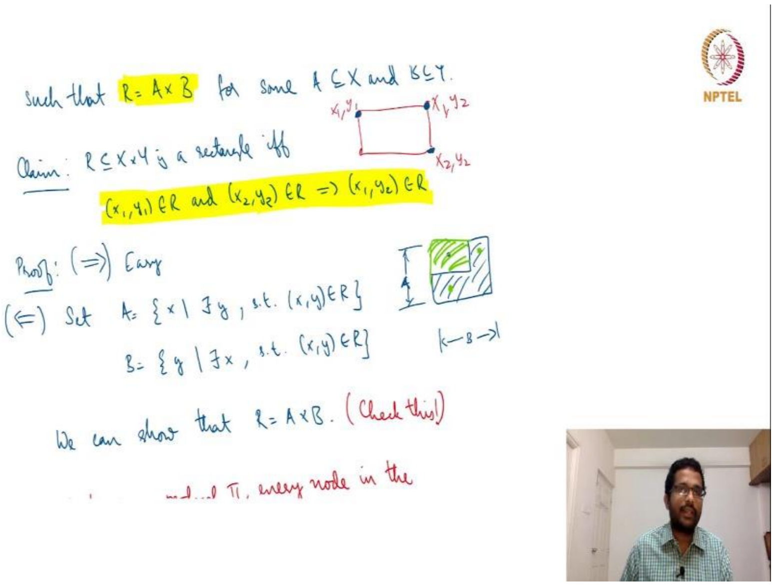
So, then these 4 parts together will form the combinatorial rectangle, there could be even more parts. But so, all the green in the coming back to the original matrix all the green outlined 1s are form a combinatorial rectangle or the blue outline 1s also form a combinatory rectangle or the yellow highlighted 1s also form a combinatorial rectangle. So, it is just a subset of the form A cross B.

And the key point here is that why did we define this is that we observe that all the leaves all the entries that correspond to all the leaves are combinatorial rectangles. All the entries that correspond to a certain leaf are combinatorial rectangles and not just that all the entries correspond to leaves are all output the same value. So, the green outline entries all correspond to the 0 output blue outlined all correspond to the 1 output.

And you can even go back and verify the earlier tree that we drew for equality. You can see that each the outputs corresponding to each leaf is a combinatorial rectangle. So, let us try to understand why this happens? There is another definition of combinatorial rectangle. So, A set R is a rectangle, so the before getting into that one more point, I want to say is that combinator rectangle need not be a contiguous one block rectangle it could be something like multiple blocks also.

So, it could be something like this as long as it is some cross product of A cross B this also could be a combinatorial rectangle. So, where A will be this domain this domain and B will be this domain.

(Refer Slide Time: 28:51)



So, coming to the other definition. The other definition is that R is a rectangle if and only if whenever you take x 1 y 1 in R which is this corner. And x 2 y 2 in R the points that I am marking the blue colour that necessarily means that x 1 y 2 also must be in R. So, this may seem very familiar to what we said about. Why is this true? So, suppose it is a rectangle in the sense that suppose it is of the form A cross B.

Then we know that if x 1 y 1 is in R, then x 1 is in A and x 2 y 2 is in R means y 2 is in B. So, A cross B means x 1 y 2 is also in R. To see the other direction, suppose x 1 y 1 in R and x 2 y 2 in R implies x 1 y 2 in R. Now consider the sets let us say so then we have to show that R cannot be of something like this it has to be a proper rectangle. So, the answer is consider the set of all is a set of all x s for which x y is in R.

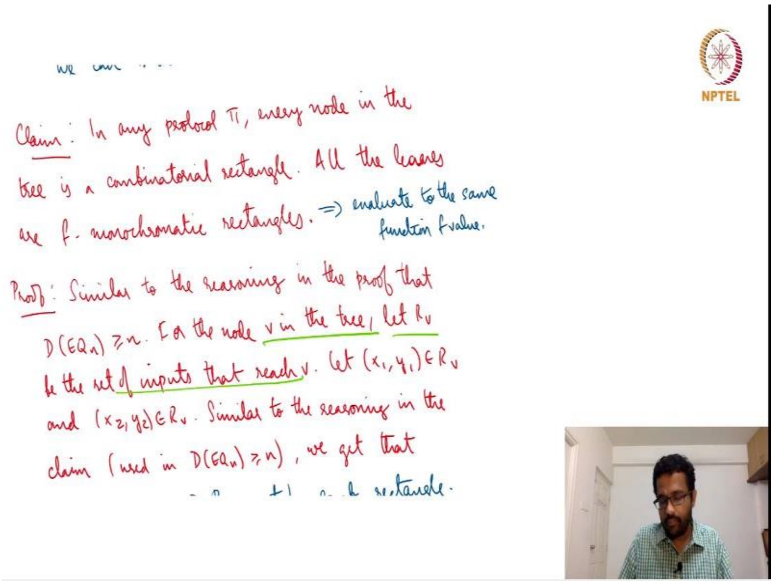
So, maybe set of all x's for which x y is in R and you call this A. So, the projection onto the x domain and similarly consider the projection onto the y domain and call it B. All the pairs y is such that x y there is some x for which x y is in R. And the claim is that A cross B, R will be equal to A cross B. Why is that? That is because, consider a point let us say we want to show that this l shape thing R cannot be an l shaped theme, so consider this point outside this.

Now we know that there is maybe I will take it different colour or maybe green graph there is this green point here. That is in R and there is another green point here that is also in R, because otherwise that is why this blue point was an A cross B. The x axis x coordinate of blue point was in a because of some point and similarly for the y coordinate now. Because these green points are in the rectangle.

And since we have the property that if x 1 y 1 and x 2 y 2 are in R then x 1 y 2 is in R this blue point also must be in R. So, like that we can fill this entire this range as well, so that means R also will be a rectangle. So, this means that this is another way to define a rectangle. If I have this property then what I have the shape that I have is a combinatorial rectangle. So, this is another different way to define.

So, it rectangular is either two ways to define one is the it is the it is A cross B where A is a subset of x B is a subset of y or it is any set that has a property that if x 1 y 1 and x 2 y 2 are in R that implies that x 1 y 2 is in R.

(Refer Slide Time: 32:14)



And one point that I want to say is that consider any protocol tree like this, any node every node here every vertex in this tree corresponds to a combinatorial rectangle. So, in the beginning if you notice the root here corresponds to the entire matrix and then we are dividing it based on whether x is 0 or 1 1 or x is 2 or 3 1s. So, then we divide it so 0 or 1 1 means it is either the first 3 and the 5th row will correspond to.

It is just the first 3 and the 5th row; go to the left and the 4th 6th 7th and 8th row goes to the right. And but then this time the only the rows are divided the columns all the columns so it is something like this. This is what goes to the green part goes to the left-hand side and the red part goes to the right-hand side. So, as you can see even both the green and red are combinatorial rectangles.

So, the claim is that in any protocol every node is a combinatorial rectangle and here you can see now we came to the green part and y has 3 1s will correspond to let us say this the sub path here that is a further refinement so that is also a combinatorial rectangle, that is the left path the right path from here. So, in any protocol every node is a combinatorial rectangle and further all the leaves are all f monochromatic.

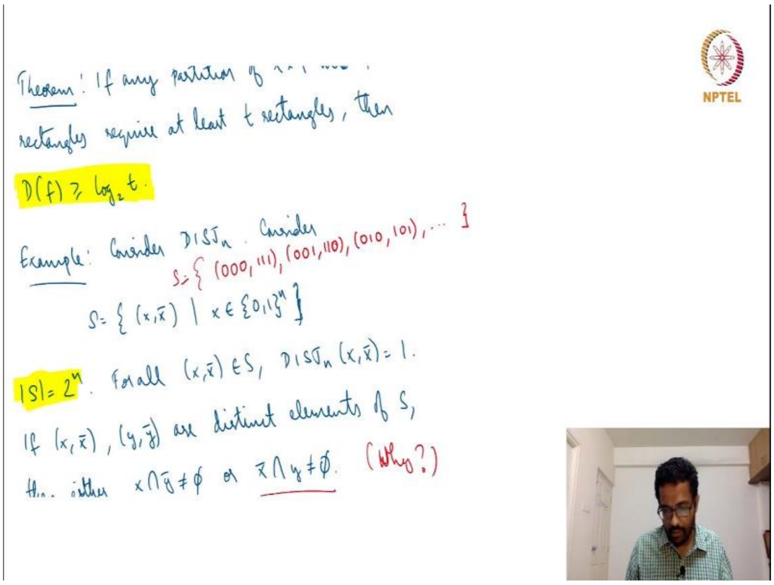
Meaning all the leaves have correspond are places where the function value is the same. So, in a leaf either the function value is all 0 or the function value is all one you cannot have 1 point of the of the leaf with f x equal f x y = 0 and another point of the leaf where f x y = 1 of the same leaf and we have actually in fact actually seen the idea of the proof. So, we saw this proof earlier in lecture the previous lecture.

If x 1 y 1 and x 2 y 2 lead to any leaf then x 1 y 2 and x 2 y 1 also must reach the same leaf and in fact this is true for any not necessarily leaf any path any node in the tree. So, this means that any node in the tree corresponds to a combinatorial rectangle. So, I have written it here anyway consider any node in the tree v and let R v be the set of inputs that reach v. Now, the claim is that if x 1 y 1 and x 2 y 2 reach R v then x 1 y 2 also must reach R v.

Which means R v must be a rectangle combinatorial rectangle. And since if v is a leaf, then we output the function value so either the function value output at that leaf is 0 or 1. So, which; means all the entries at that reach there which is R v has to be 0 or 1. So, which means which is what we call f monochromatic otherwise this protocol does not compute that f. So, if there is a leaf at which some points correspond to 0 output some points correspond to one output.

That means it you have to further go down so that point is not a leaf. So, all the leaves are f monochromatic meaning evaluate to the same function f value.

(Refer Slide Time: 36:51)



So, any node in the tree is a combinatorial rectangle. So, starting from the so basically what happens is when you start the root node is the entire matrix and then every time Alice speaks there is a subdivision of the rows. So, it need not be just one cut it could be like it could be a combinatorial fact like over here the green was not just like one split but then it is actually one split but some rows go into one side some rows going to the other side.

And whenever Bob speaks it is a vertical cut so like that every time it split horizontally or vertically but it a rectangle when cut entirely it remains a rectangle that is a proof. So, this so why are we doing all this? We are doing all this to get a technique to bound the communication complexity of any function. So, suppose we know that a certain function has a certain matrix. And however which way you try to divide it into combinatorial rectangles.

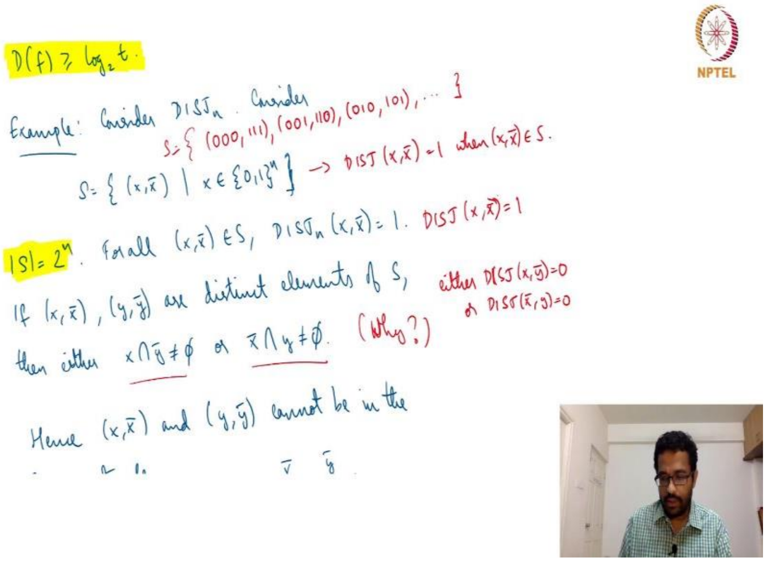
Suppose that matrix requires in it to divide into 1000 rectangles that means that you cannot get divided into smaller numbers you cannot divide into 999 rectangles. Any way you try to divide you need at least 1000 rectangles meaning any protocol that you have for the function must have at least 1000 leaves. So, if it has 1000 leaves the tree must be sufficiently deep you should go up to log 1000 depth, which is log 1000 mod to the base 2 is at least 10 depth.

So, that is the statement here if any partition of x cross y with the matrix f cross y into f monochromatic rectangle requires at least t rectangles then D f is at least log t. So, in fact I can say it is ceiling of log t, because you cannot have half bits of communication. So, just to give an example; consider disjointness that we said earlier. Disjointness that is a function that indicates whether the two sets are disjoint or not consider disjoin us.

Now consider this set maybe just write it clearly so we will show that disjointness must have some number of leaves. So, now let us consider disjoin us over 3 and consider this following 0 0 0 1 1 1 0 0 1 1 1 0 0 1 0. So, what am I doing here? I am just trying to 1 0 1 I am writing all the strings x and the complement string. So, these are sets a certain set x and or a certain set a and their complement.

Now the claim is so how many elements are there in s this is called s how many elements are there in s, s has 2 power n elements because I take 2 power in possible strings x and for each possible string x i it is just one possible complements. I have written here already s is of size 2 power n and the claim is that each of these pairs must go to distinct leaves you cannot have two of these pairs come to the same leaf.

(Refer Slide Time: 40:51)

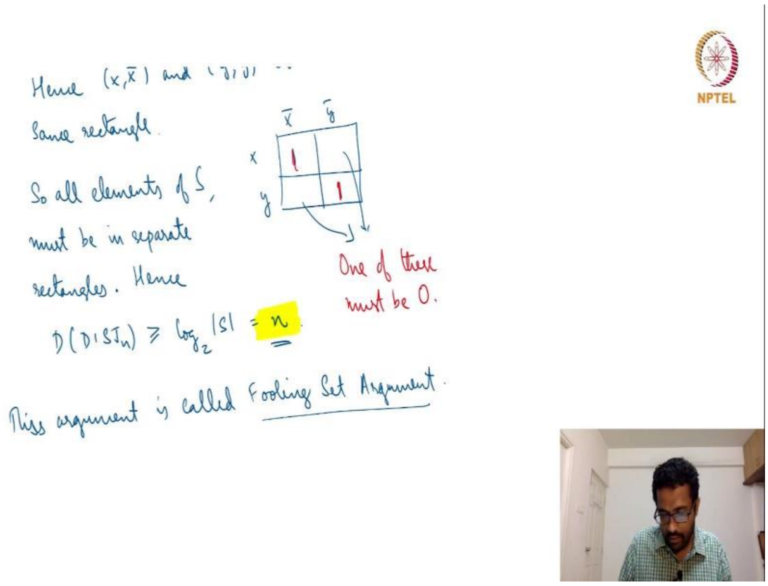


So, basically if x and y are not equal then x, x complement and y, y complement cannot go to the same leaf. Why? Because either x intersection y complement or y intersection x complement is non empty it is disjointness function. So, x intersection x complement is certainly empty and y intersection y complement is empty. So, disjoint sorry so we know that disjointness of all the entries in x, x all the entries in x is 1.

So, disjointness that all the entries in s are 1, but if you take two distinct entries of x, x s x, x complement and y, y complement either x and y complement intersect or y and x complement intersect. Which means that either disjointness of x and y complement equal to 0 or disjointness of x complement and y = 0. So, now you can see why they cannot go to the same leaf if x, x complement and y, y complement go to the same leaf.

Then x, y complement and x complement y also must go to the same leaf. But then we know that x, x complement and y, y complement are 1 entries.

(Refer Slide Time: 42:32)



As I have drawn here x, x complement is 1 y, y complement is 1 but we know that at least one of the other entries x, y complement or y, y x, x complement y one of them must be 0. So, they cannot be part of the same monochromatic rectangle so which means that all the elements of x s must be in separate rectangles. So, that means we have 2 power n different entries all of which must all of which must be in separate rectangles.

Which means the tree must have at least 2 power and leaves which means the height of the tree must be at least n. So, the disjointness of the deterministic complexity of disjointness must be at least n. So, what we did here is to demonstrate a set, demonstrate a set such that all the entries of that set must be in distinct leaves. And this kind of a set is called fooling set and this kind of an argument is called a fooling set argument.

So, we will see some more examples in the next lecture. So, s has some certain size and all of s must go to different leaves. So, that says that there has to be at least this many leaves for the protocol tree so, the protocol tree has to have at least this much height which means the deterministic communication complexity is at least as much. So, this particular claim; just try to think of it if x, x complement and y, y complement are distinct elements of x.

Why must it be the case that? That x complement x and y complement intersect or x complement and y intersect think about this, so, I should write y and x complement here so at least one of them must intersect just think about it. If x and y are of different sizes then you can see that at least one of them must intersect because of the sizes. But if they are of the same size also you can make a different argument just think over it.

So, what we have seen in this lecture just to summarize we saw the protocol tree in the previous lecture. We saw this notion that if x 1 x 1 and x 2 x 2 go to the same leaf then or same node then x 1 x 2 also must go to the same node. And that led to the definition of what we call combinatorial rectangles. Combinatorial rectangles are just A cross B where A subset of x and B is subset of y.

Or we had the other definition where it says at any set where x 1 y 1 and x 2 y 2 being in the set implies x 1 y 2 being in the set. So, both of them are equivalent definitions if you have one you have the other. And we use that to show that show the following theorem that if any partition of and we saw that in any protocol tree every node is a combinatorial rectangle. And further all the leaves are f monochromatic combinatorial rectangles.

F monochromatic means in those rectangles the f values are all 1 are all 1s. So, consequently given a function look at the matrix what are the smallest number of ways in which you can divide the matrix into monochromatic rectangles or the smallest number of ways you can divide the matrix into monochromatic rectangles. Suppose, it is t then any protocol must take at least log t must send at least log t bits of communication.

Again, it is exactly the same argument as what we saw. And disjoin us we saw that example of disjointness we saw that we showed that it showed a set of size 2 power n and we said that any partitioning into monochromatic rectangles must have at least 2 power n rectangles this implies that disjoiners has complexity at least log of 2 to power n which is n. So, that is all that we saw in this lecture we saw the main thing being the definition of combinatorial rectangle.

And with that I think I will conclude this lecture and in the next lecture we will see maybe few more techniques for lower bounding the communication complexity. After which we will in subsequent lectures, we will see one application on circuit lower bounds. Thank you.