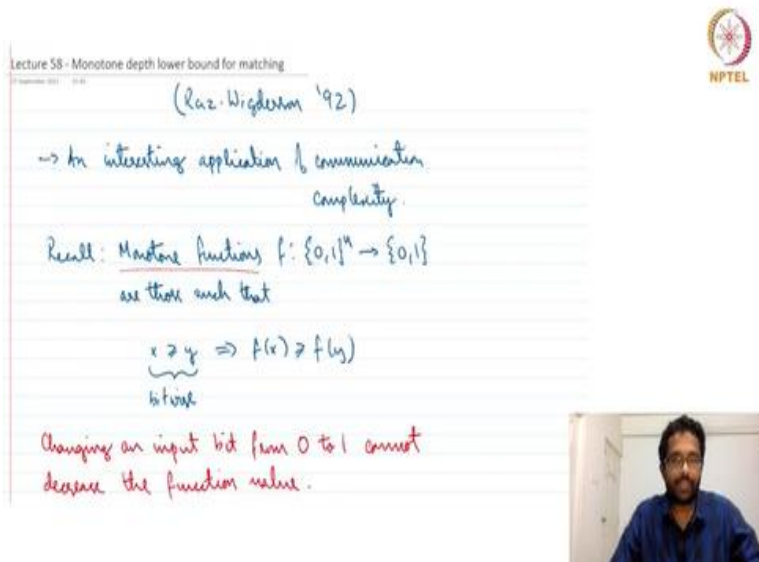


**Computational Complexity**  
**Prof. Subrahmanyam Kalyanasundaram**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Hyderabad**

**Lecture - 58**  
**Monotone Depth Lower Bound for Matching**

(Refer Slide Time: 00:15)





Lecture 58 - Monotone depth lower bound for matching  
(Raz-Wigderson '92)

→ An interesting application of communication complexity.

Recall: Monotone functions  $f: \{0,1\}^n \rightarrow \{0,1\}$   
are those such that

$$\underbrace{x \geq y}_{\text{bitwise}} \Rightarrow f(x) \geq f(y)$$

Changing an input bit from 0 to 1 cannot decrease the function value.



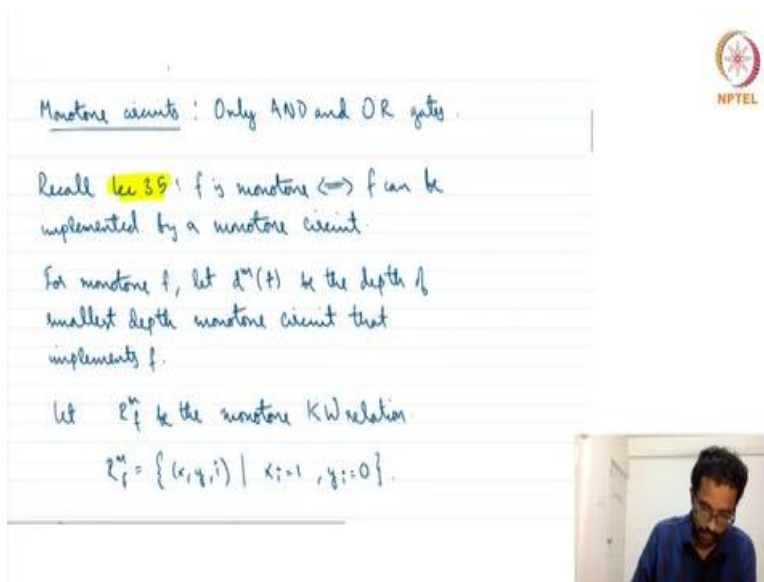
Hello and welcome to lecture 58 of the course computational complexity. In the previous lecture we saw Karchmer Wigderson relation, where we could relate the deterministic communication complexity of a certain relation with the depth complexity of a function, when it is represented as circuits. In today's lecture we are going to use the Karchmer Wigderson relation. In order to get a lower bound on the depth complexity of a certain function being given a graph determining whether the graph has a perfect matching or not.

And this result is by Raz and Wigderson from 1992, so it is a very interesting and somewhat of a surprising application of the communication complexity. Because, we are getting a depth lower bound on a problem that is on graphs. So, before we start let me just recall monotone functions and monotone circuits. So, monotone functions are those where, if there are two inputs  $x$  and  $y$ , if  $x$  is greater than or equal to  $y$ , then  $f$  of  $x$  is greater than or equal to  $y$ .

So, remember all of this is in the Boolean function, so  $n$  bits to 0, 1 output. And, when I say  $x$  is greater than or equal to  $y$ , I mean bit wise. So, the function the output of 01101 should always be greater than 001011. Because, here  $x$  is greater than  $y$ , so  $f$  of  $x$  should be greater than equal to  $f$  of  $y$ . Another way to view the same thing is that, if you change one input bit from, if you flip it from 0 to 1, then the output cannot go from 1 to 0, the output cannot decrease.

Output may remain same at 0 or 1, the output may increase from 0 to 1. But it cannot go from 1 to 0. This is another way to view monotone functions. And monotone circuits are just whatever we can implement with just AND and OR gates. In fact, we had seen these definitions in lecture 35 and in fact, there was also an exercise where I had asked you to verify that a function  $f$  is a monotone function.

**(Refer Slide Time: 02:51)**





Monotone circuits: Only AND and OR gates.

Recall lec 35:  $f$  is monotone  $\iff$   $f$  can be implemented by a monotone circuit.

For monotone  $f$ , let  $d^*(f)$  be the depth of smallest depth monotone circuit that implements  $f$ .

let  $R_f^*$  be the monotone KW relation.

$$R_f^* = \{(x, y, i) \mid x_i = 1, y_i = 0\}.$$


If and only if it can be implemented by a monotone circuit. So, hopefully if you have not convinced yourself of this simple fact already you may want to try that out and then listen to the rest of the lecture. So, the point here is that any function, if it is a monotone function, then it can be implemented by a monotone circuit and anything that can be implemented in a monotone circuit is a monotone function. So, it is an equivalent characterization.

So, given a function, given a monotone function, I can construct a given a monotone function, I can construct circuits that implement this circuit families, and I can look at the depths of these

circuits. Now, what I am further restricting it is, so this is the depth of the function. What I am further restricting it into is the monotone depth denoted by  $d^m$  of  $f$ . So, this is the monotone depth.

This means you restrict yourself to only the monotone circuit characterizations of  $f$ , or monotone circuit realizations of  $f$ . So, even if  $f$  is monotone perhaps you could have a very compact circuit using NOT gates, using negations and some things like that. And perhaps when you try to implement it using a monotone circuit alone without using NOT gates; perhaps the size increases, so the monotone depth could be more than the depth of the function.

Because I am only looking at functions or circuit implementations that do not have NOT gates. So,  $d^m$ , the superscript  $m$  of  $f$ , this stands for the depth of the monotone circuits that implement  $f$ , so this is the monotone depth of a certain function, which is the depth using monotone circuits. And last lecture we defined the Karchmer Wigderson relation, which was  $x \neq y$  such that  $x_i$  and  $y_i$  are different.

So, let me just go to the previous lecture and show you what it was. So, Karchmer Wigderson relation was set of all  $x \neq y$  such that  $x_i$  is not equal to  $y_i$ . And here we are defining the monotone Karchmer Wigderson relation. So, everything is same except that in the case of Karchmer Wigderson relation, we said that  $x_i$  not equal to  $y_i$ . In the monotone variant we say  $x_i = 1$  and  $y_i = 0$ .

So, I am telling  $x_i$  should be 1 and  $y_i$  should be 0, and we cannot have  $x_i$  not equal to  $y_i$  in general. So, we cannot have  $x_i = 0$  and  $y_i = 1$ . So, this is a monotone Karchmer Wigderson relation.

**(Refer Slide Time: 06:01)**



Let  $R_f^m$  be the monotone KW relation

$$R_f^m = \{(x, y, i) \mid x_i = 1, y_i = 0\}.$$

Theorem:  $d^m(f) = D(R_f^m)$  } Exercise  
(Monotone version of KW).

Given a graph  $G$  on  $n$  vertices, does it contain a perfect matching?

Let  $\text{MATCH}_G: \{0,1\}^n \rightarrow \{0,1\}$  be the above function.

And just like what we saw in the last lecture, we can also define the monotone variant of the Karchmer Wigderson theorem, which is that the monotone depth of a certain function is equal to the deterministic communication complexity of the monotone Karchmer Wigderson relation. So, it is exactly the same theorem, but for the subscripts  $m$  in the left hand as well as the right-hand side.

So, you can work out the proof, it is exactly along the same lines but then, you just have to take care of the fact that  $f$  that the depth is of the monotone circuit and so which means in the case of monotone circuits as I explained the previous lecture all the negations, all the inputs are in the positive form. So, I will not have inputs of the forms that I complement  $z_2$  complement and so on. It will all be  $z_1, z_2, z_3$  and so on.

So, the monotone depth is equal to the communication complexity of the monotone Karchmer Wigderson relation, so this you can try out.

**(Refer Slide Time: 07:09)**

(Menderson version of KW).

Given a graph  $G$  on  $n$  vertices, does it contain a perfect matching?

Let  $MATCH_n: \{0,1\}^{\binom{n}{2}} \rightarrow \{0,1\}$ , be the above function.

Theorem (Ray-Wilson):  $d^M(MATCH_n) = S(n)$ .

So, now let me come to the main topic. So, given a graph, so you can think of giving a graph as an adjacency list or adjacency matrix. So, I can view a graph as a big vector, so for instance here I have a graph on four vertices, let me number these vertices let us say 1, 2, 3, 4. So, now I could represent this graph as a 6 bit vector, so I would say it is a simple graph, so let me represent it as a 6 bit vector.

So, the first bit indicates whether there is an edge from 1 to 2, then second bit indicates edge from 1 to 3, 1 to 4, 2 to 3, 2 to 4 and finally 3 to 4. So, in for this graph it will be 1 to 2 there is an edge, 1 to 3 and 1 to 4 there are no edges. So, we put 0 0, 2 to 3 there is an edge, 2 to 4 there is an edge, 3 to 4 there is an end. So, this is the vector that indicates this graph. So, now the question is given this  $n$  choose 2 bit vector, which is an entire description of the graph.

So, there are no self-loops, assume there are no self-loops an undirected graph, so  $h$  from  $i$  to  $j$  is same as  $h$  from  $j$  to  $i$  and no self-loops. So, given this vector, one has to determine whether there is a perfect matching. So, you can check this graph, does it have a perfect matching? Yes, indeed it does, so you could pick the edge 1 to 2 and maybe the edge 3 to 4. In fact, this is only perfect matching. But, let us say if I had a couple of more edges let us see 5 and 6.

Now does this graph have a perfect matching? Even this does, because you could consider something like this, this is a perfect matching, in fact I think this is the only perfect matching of

this graph. But, if it was let us say if we did not have this edge 2, 4 then, we do not have a perfect matching. Because 5 has to match with 1 and 6 has to match with 3. So, 2 and 4 do not have pairs. So, this for the vector of this particular graph there is no perfect matching.

So, the MATCH n function is just given this vector n choose 2 length long vector. It will map to 0, 1 depending on whether the graph has a perfect matching or no perfect matching.

**(Refer Slide Time: 10:03)**

So, now let us move to the theorem by Raz and Wigderson, which is a lower bound on the depth complexity of this function. This says that the depth complexity of this matching function is at least linear. The depth complexity of the matching function is at least linear you have to have a linear depth circuit to compute this. So, you may recall that we saw an RNC circuit, to determine whether the given bipartite graph has a perfect matching.

But, so in RNC circuit it is like polylog depth circuit. But here we are saying that, if we restrict the circuit to a monotone circuit, again that was RNC. But here it is monotone circuit and deterministic. Then, the circuit has to be of depth, linear depth. So, we will this is proved where the Karchmer Wigderson relation. So, given the matching function what is the corresponding Karchmer Wigderson relation?

So, one way to think of it is that Alice and Bob get vectors, Alice has a vector which outputs 1. So, which means he has a graph, so this vector is a representation of a graph. Alice, she has a graph that has a perfect matching, because the matching function evaluates to 1. And, Bob has a graph that does not have a perfect matching. So, the matching function evaluates to 0. So, this means that Alice must have an edge that is not there with Bob.

Because, matching is a monotone function, if you put an additional edge the perfect matching that was already there does not vanish. So, Alice has a perfect matching, Bob does not have a perfect matching. So, they have to determine one specific edge that Alice has and Bob does not, this is the requirement. Alice and Bob needs to determine that one edge that is in Alice's graph but not in Bob's graph. So, this is the requirement of the function.

We will show that the disjointness function, so the disjointness function we already said that, it is you require at least  $n$  bits. We will show that this there is a randomized reduction from the disjointness function to this this relation, this Karchmer Wigderson relation on matching. So, you can reduce the disjointness function in a randomized fashion to this Karchmer Wigderson relation.

So, if there is a Karchmer Wigderson, if the relation a protocol for the Karchmer Wigderson relation on matching. We will correspond to a randomized protocol for disjointness, because given a disjointness instance we transform it to a matching instance or an instance of computing the Karchmer Wigderson relation. So, this transformation is randomized and then run the protocol for the Karchmer Wigderson relation on matching, we run the protocol in R MATCH.

And then using that, we will be able to determine whether the inputs were disjoint or not. So, it is a randomized process to decide disjointness, randomize protocol to decide disjointness. And, the interesting thing is that, this will involve no extra communication. Whatever this reduction entails, this will be done locally at Alice's end and Bob's end. They will not need to communicate with each other to perform this reduction.

So, the communication complexity of the resulting randomized protocol for disjointness will be exactly the same as the communication complexity of the complex of the protocol that decides the Karchmer Wigderson relation or the computes the Karchmer Wigderson relation. So, this gives us a randomized protocol for disjointness.

**(Refer Slide Time: 14:09)**

NPTEL

But at here

Theorem: Rand. Complexity of  $DISJ_n \geq \Omega(n)$   
(Kalyanasundaram & Schnitger '92).

$\Rightarrow D(RMATCH) \geq \Omega(n)$ .

$\Rightarrow d'(MATCH) \geq \Omega(n)$ .

We will present a reduction to a simpler problem:  $MATCH_{SAT, NFI}$ : Given a graph

But then there is a result, which I will not prove, I will just state that the randomized complexity of disjointness is linear, is at least linear. So, this was a result in 1992 by Kalyan Sundaram and Schnitger, who showed that the randomized complexity of disjointness is at least linear. And, here we have a randomized where this reduction to the Karchmer Wigderson relation, we have a randomized protocol for disjointness, without any extra communication complexity.

So, this means that, since we know that randomized complexity of disjointness is at least order  $n$ , at least  $\omega n$ , this means that the complexity of  $R MATCH$  is also at least order  $n$ . And by the Karchmer Wigderson theorem in the monotone setting, this means that, because the determinacy complexity of  $R MATCH$  is equal to the monotone depth of matching as well. So, this implies that the depth complexity of the matching, circuit that computes matching in the monotone setting is also linear.

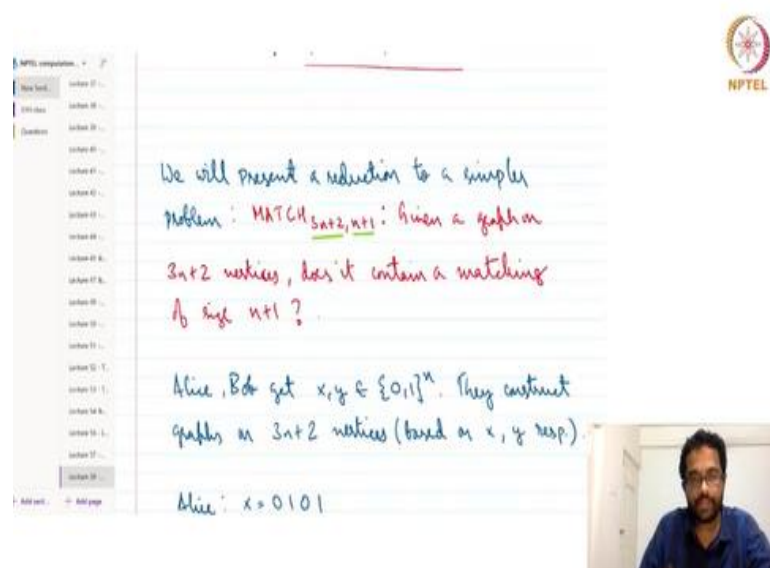
So, that is the overview of the proof. Once again, I will just quickly go through, we reduce disjointness in a randomized fashion to the  $R MATCH$ .  $R MATCH$  being the Karchmer



Wigderson relation on matching. So, a protocol for R MATCH gives a randomized protocol for disjointness with the same complexity, because it involves no extra communication. But we know that randomized complexity of disjointness is at least linear.

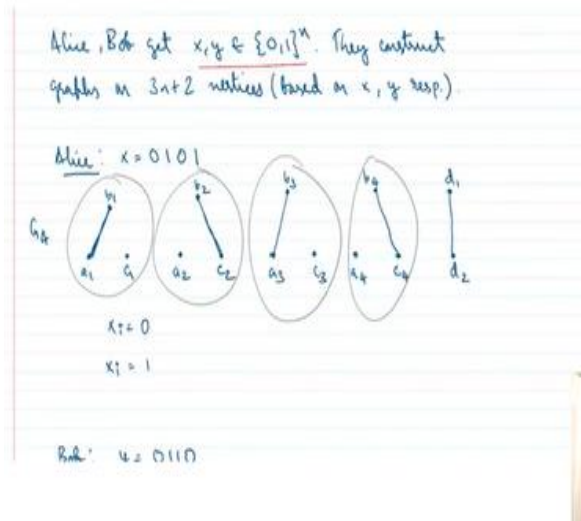
So, the protocol for R MATCH should also be at least linear. But, by the Karchmer Wigderson theorem we know that, the complexity of the protocol R MATCH is equal to the depth complexity of the function, so the depth complexity, the monotone depth complexity of the matching function is also at least linear.

**(Refer Slide Time: 16:10)**



So, now what remains is to see the reduction, this this particular reduction is what is left. So, in fact it is we will present a slightly simpler reduction, but it is not very difficult to see the idea from this, so the core idea I will just present in reduction. So, instead of asking whether  $n$  vertex graph as a perfect matching, we will ask whether a graph with  $3n + 2$  vertices has a matching of size  $n + 1$ , so a graph with  $3n + 2$  vertices has a matching of size  $n + 1$ .

**(Refer Slide Time: 16:55)**



So, how will the reduction happen? So, Alice and Bob get 2 strings of length  $n$  and they will construct matching instances locally. So, but they will construct different graphs and that will help us do the; reduced disjointness to this matching function. Both of them will construct graphs on  $3n + 2$  vertices and their construction will be dependent on the strings that they have. So, the  $3n + 2$  vertex graph looks like this, so  $3n + 2$  vertices are in this form for each.

So, let us say  $n = 4$ , then we have 4 such  $n$  tuples or 4 such 3 tuples sorry this is 1, this is 1, this is 1 and this is 1. Four such 3 tuples  $a_1 b_1 c_1$ ,  $a_2 b_2 c_2$ ,  $a_3 b_3 c_3$  and  $a_4 b_4 c_4$ , so that corresponds to  $3n$  and then there is a single edge or there are two vertices called  $d_1$  and  $d_2$ . That is there by the right side two words is called  $d_1$  and  $d_2$ , so this is the  $3n + 2$  vertices. Now what are the edges? So, let me first describe.

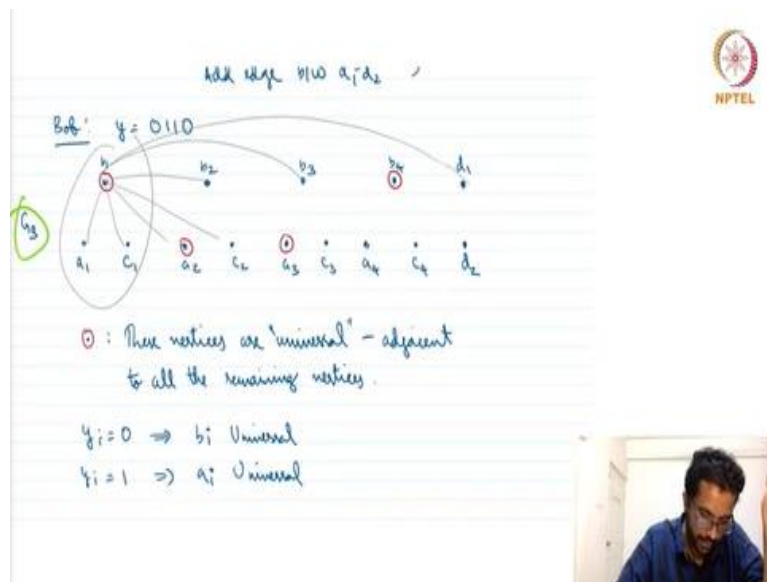
So, the vertices are common to the graphs of both Alice and Bob, the vertices are common only the edges will differ. So, let us see what is the graph, Alice's graph, let us call it  $G_A$ . So, let us look at this Alice's string, let us say the string is  $0101$ . So, the first bit is 0, so  $x_i = 0$  means, add edge between  $a_i$  and  $b_i$ . So, you adjust you make  $a_i$  and  $b_i$  adjacent  $x_i = 1$  mean you add an edge between  $b_i$  and  $c_i$  and that is it.

So, Alice's graph and finally also add edge, this you always add for Alice  $d_1$  and  $d_2$ . So, Alice's edge, Alice's graph has exactly  $n + 1$  edges, so there are  $n + 1$  edges for each bit whether

it is 0 or 1, she adds an edge between either  $a_i$  and  $b_i$  or  $b_i$  and  $c_i$  and finally there is an edge between  $d_1$  and  $d_2$ . So, this is the graph of Alice, it is just so in fact the graph itself is just a perfect matching, there is no other things going on here a lot of vertices without any edges.

This is Alice's graph. So,  $x$  is 0 1 0 1 so you connect  $a_1$  and  $b_1$ , but second bit is 1 so you can have  $b_2$  and  $c_2$ , third bit is 0 so you connect  $a_3$  and  $b_3$ , fourth bit is 1 you connect  $b_4$  and  $c_4$  and finally you connect  $d_1$  and  $d_2$ .

**(Refer Slide Time: 20:17)**



Now let us see Bob's graph, so again it is on the same set of vertices,  $a_1 b_1 c_1$  and  $a_2 b_2 c_2$  and so on. So, same set of vertices we call this graph  $G_B$ , and what are the edges, so what we do is, we have these universal vertices. So, what do universal vertices mean? So, we mark  $b_1$  as universal, so what we do is you have an edge from  $b_1$  to  $a_1 c_1 b_2 a_2$  everything. So, to all the vertices, so you make  $b_1$ , so you make so this notation where I have a red circle.

These vertices are made universal meaning you make them adjacent to all the remaining vertices of the graph, so even including like  $d_1 d_2$  everything. And which vertices are made universal?  $y_i = 0$  means you make  $b_i$  universal, and  $y_i = 1$  you make  $a_i$  universal. So, in every group, every 3 tuple or  $a_1 b_1 c_1$ , a  $a_i b_i c_i$  group you make one vertex universal depending on what  $i$ th bit is. So, you can see the string is 0 1 1 0, so since the first bit is 0  $b_1$  is made universal.

Second bit is 0 so a 2 is made universal, so second bit is 1 so a 2 is mainly universal, third bit is 1 a 3 is made universal, fourth bit is 0 so b 4 is made universal. So, because it will get messy, I am not drawing all the edges because, a universal vertex means it is connecting to all the remaining edges, the  $3n + 2$  vertices. So, a universal vertex is just is result in  $3n + 1$  edges, so many edges it will just get messy. So, there are four such universal vertices.

And this is the graph  $G_B$ , Bob's graph. What I want to mention here is, look at Alice's string and Bob's string. The claim is this so notice that Alice's graph has a matching of size  $n + 1$  in fact it is just a matching of size  $n + 1$ . What can we say about Bob's graph? We know that any edge in Bob's graph, one of the endpoints is the universal vertices. So, for instance there will not be an edge between a 4 and c 4 because, neither vertex is universal.

So, this edge will not be there, a 4 and c 4 will not be there. So, and how many universal vertices are there? There are overall  $n$  universal vertices. So, any edge has one of these  $n$  universal vertices as its endpoint, there could be edges that just that have both of them as universal vertices and  $n$  points, but that is okay. Every edge has at least one of its  $n$  points as a universal vertex.

**(Refer Slide Time: 23:53)**

→  $G_A$  is a matching!

Note:  $G_A$  has a matching of size  $n + 1$ .

Max matching in  $G_B$  has size  $\leq n$ .

If  $x$  and  $y$  are disjoint, the universal vertices in  $G_B$  "cover" Alice's edges. If  $x$  and  $y$  are not disjoint, then there is an edge in  $G_A$ , not covered by a universal vertex in  $G_B$ .

Proof for DISJ:



This means that, you cannot have a matching of size bigger than  $n$  because, if you have a matching of size bigger than  $n$ , that means you have  $n + 1$  edges, that means you have  $n + 1$  edges and each one of them can should have one of the universal vertices as its  $n$

points, and you cannot share an  $n$  point. But then you have only  $n$  universal vertices. So, any matching in  $b$  in graph  $b$ , Bob's graph  $G_B$  has size at most  $n$ .

And  $G_A$  has a matching of size  $n$  plus, in fact it is a matching,  $G_A$  itself is just a matching, there is nothing else. But any matching in size and  $G_B$  is of size at most  $n$ . So, now Alice's graph has now by construction it is a yes instance, and Bob's graph is a no instance. So, the function matching function applied to Alice's graph outputs 1 and applied to Bob's graph output 0. Because, now we are trying to determine whether, is there a matching of size  $n + 1$ .

That is what we said, we will do a simpler reduction. So, now I want to point out something about the way these things were constructed. So, the point is that if  $x$  and  $y$  are disjoint meaning, there is no index where both  $x_i$  and  $y_i$  are 1. Then what happens is that? There are three possibilities,  $x_i = 0$  and  $y_i = 0$ ,  $x_i = 1$  and  $y_i = 0$ ,  $x_i = 0$  and  $y_i = 1$ . In each of these case the universal vertex of that group of the group  $i$  is an  $n$  point of the edge added by Alice.

So, let us see so in fact in this figure all possibilities are covered. So, for the first group  $x_1 = 0$  and  $y_1 = 0$ , so the universal vertex is  $b_1$  and that is an  $n$  point of Alice's edge  $a_1 b_1$ . For the second bit  $x_2 = 1$  and  $y_2 = 1$ , so it is not disjoint. But let us see what happens. The universal vertex is  $a_2$ , but  $a_2$  is not an end point of  $b_2 c_2$ , which is edge added by Alice. In both the third group and fourth group, Alice and Bob are disjoint.

Alice has 0 and Bob has 1 for the third index, Alice has 1 and Bob has 0 for the fourth index, and you can see that in the third group  $a_3$  covers  $a_3 b_3$  and in the fourth group  $b_4$  covers  $b_4 c_4$ . So, any possibility if  $x$  and  $y$  are disjoint then all the end points of Alice's graph are covered by Bob's graph. So, what I mean is that? When if Alice and Bob are disjoint, then all the edges added in  $a_1 b_1 a_2, b_2 c_2 a_2, a_1 a_2 b_2$  whatever all these groups.

All the edges added in these groups  $a_1 b_1 c_1, a_2 b_2 c_2$  are covered by the universal vertices added in Bob's graph. The only edge in Alice's graph that will not be covered is the edge  $d_1 d_2$ , because Bob is not going to add a universal vertex in  $d_1$  or  $d_2$ . So, this is the key observation I


just written down that here, if  $x$  and  $y$  are disjoint the universal vertices in  $G \oplus B$  cover Alice's edges, so this is the key point it covers Alice's edges.


If  $x$  and  $y$  are not disjoint, then there is at least one edge in Alice's graph, at least one edge that is not covered by the universal by a universal vertex in  $G \oplus B$ . So, if  $x$  and  $y$  are disjoint, then the universal vertices in  $G \oplus B$  cover Alice's edges, in fact this one more point except  $d_1 \oplus d_2$ , it covers all the edges except  $d_1 \oplus d_2$ . But if they are not disjoint then there is an edge apart from  $d_1 \oplus d_2$ , that is not covered by the universal vertex in  $G \oplus B$ .

**(Refer Slide Time: 28:46)**

Protocol for  $DISJ_n$ :

- \* Alice & Bob build  $G_A$  and  $G_B$ .
- \* Using shared random bits, randomly permute vertices of  $G_A$  and  $G_B$  (with the same permutation).
- \* If  $DIST(x, y) = 1$ , then protocol must output edge  $d_1 \oplus d_2$ .
- \* If  $DISJ(x, y) = 0$ , then there are other candidate edges. So  $P_x(d_1 \oplus d_2 \text{ is output}) \leq 1/2$ .





And if this point is clear the entire the rest of the proof is very straight forward. So, all that now it boils down to is that, if it is disjoint the only edge that Alice has, but that is not there in Bob's graph is  $d_1 \oplus d_2$ . So, if Alice and Bob are disjoint the only edge that Alice's graph has, but Bob's graph does not have is  $d_1 \oplus d_2$ . If they are not disjoint, then there are multiple edges that Alice has that Bob does not have including  $d_1 \oplus d_2$ .

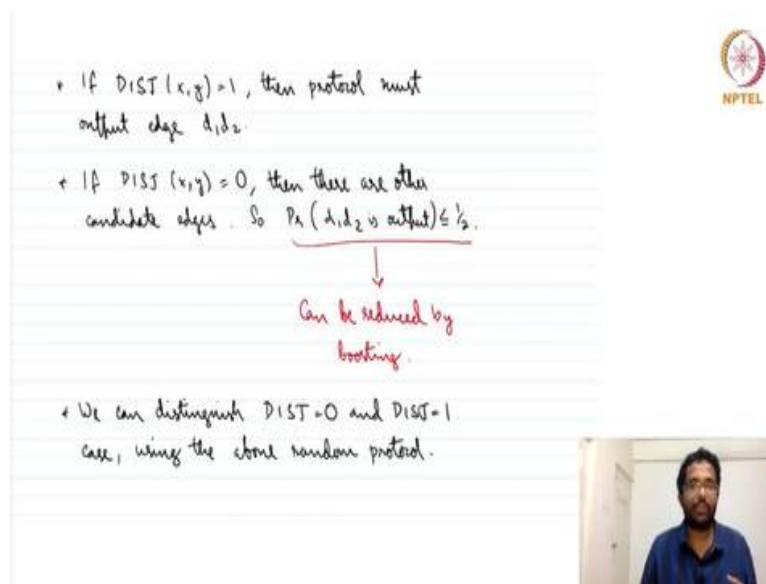
At least one more edge apart from  $d_1 \oplus d_2$  is there with Alice. So, the correct execution or correct computation of the Karchmer Wigderson and protocol of the Karchmer Wigderson relation on matching, should output an edge from Alice's graph. So, if Alice's and Bob's input are disjoint, then it will always output  $d_1 \oplus d_2$ . Because, everything else is already covered by Bob's graph. If they are not disjoint, then it could be  $d_1 \oplus d_2$  or it could be some other edge.

And that is a key thing. So, what is the protocol? Alice and Bob build their graphs  $G_A$  and  $G_B$  and there is one small step that they do, this is to remove any bias of the algorithm. So, you can come think about it later that, what they do is using shared randomness they permute their vertices and edges, with the same permutations. So, this is just think of it as this relabelling, this is to make sure that some the protocol does not have any bias towards looking at some set of edges or some other set of edges.

But, both of them do the same permutation. So, if; Alice re-labels  $a_1$  to  $x$  and  $b_1$  to  $y$  then Bob also will make the same re-labelling. So, if their inputs are disjoint, if Alice's and Bob's inputs were disjoint, then the protocol must output  $d_1 d_2$ . If they are not disjoint, then there are other edges at least one other edge that Alice can output. So, the; probability that  $d_1 d_2$  but even  $d_1 d_2$  can be output, so what is the probability that  $d_1 d_2$  can be output?

It is at most half, because there is at least one more edge and this probability that of outputting  $d_1 d_2$ , this can be further reduced by boosting you could repeat that.

**(Refer Slide Time: 31:34)**



The slide contains handwritten notes on a lined background. In the top right corner, there is a circular logo with a star and the text 'NPTEL'. The notes are as follows:

- + If  $DIST(x,y)=1$ , then protocol must output edge  $d_1 d_2$ .
- + If  $DIST(x,y)=0$ , then there are other candidate edges. So  $P_x(d_1 d_2 \text{ is output}) \leq \frac{1}{2}$ .

An arrow points from the second line to the text: "Can be reduced by boosting."

At the bottom, it says: "+ We can distinguish  $DIST=0$  and  $DIST=1$  case, using the above random protocol."

In the bottom right corner, there is a small video inset showing a man with a beard and glasses, wearing a blue shirt, speaking.

So now all that they need to do to distinguish whether  $x$  and  $y$  are disjoint. So, again recollect that we are trying to reduce disjointness to a computation of the Karchmer Wigderson relation. So, they will compute the Karchmer Wigderson relation, which means they will arrive at an edge

that Alice has that Bob does not have. They will look at the edge. If this edge is  $d_1 d_2$  they will say that the strings are disjoint.

If this is not  $d_1 d_2$ , they will say that the string is are not disjoint,  $x$  and  $y$  are not disjoint. And, we can improve this probability by boosting, by repeating and boosting. So, the bottom line is that we can distinguish the case when disjointness is 0 and disjointness is 1 by using this protocol. It is a random protocol, because we do this random permutation. So, we transform  $x$  and  $y$  into graphs and if they are disjoint the only output possible by Alice and Bob is  $d_1 d_2$ .

Otherwise, there are multiple outputs possible. So, if they if Alice output some other edge, like in this case she may output  $b_2 c_2$ , because they are not disjoint. Alice and Bob know that the strings are not disjoint. So, what is the conclusion? The conclusion is that we are computing disjointness using the Karchmer Wigderson relation on matching. We are computing disjointness using the Karchmer Wigderson relation on matching.

This means that the complexity of computing the Karchmer Wigderson relation in matching is at least that of the randomized complexity of disjointness which we know to be at least linear. So, that gives us that complexity of  $R$  MARCH is at least linear and by the relation between the complexity of  $R$  MARCH and the depth of the matching. We know that the depth of matching is also linear, the monotone depth of matching.

So, that completes the proof. I will just do a quick recap. So, we first saw monotone functions and monotone circuits again this was a recap and then we said that the Karchmer Wigderson theorem that we saw in the previous lecture has a monotone analogue. The monotone depth complexity is equal to the deterministic complexity of the monotone Karchmer Wigderson relation.

And then we defined the matching function and then we saw that  $R$  MARCH, which is the Karchmer Wigderson relation applied to the matching function, from disjointness there is a randomized reduction to  $R$  MARCH. So, a protocol for  $R$  MARCH will give a randomized protocol for disjointness without extra communication. And since the randomized complexity of



disjointness is at least linear, this implies that the deterministic complexity of R MARCH is at least linear.

And hence the monotone depth complexity of matching is also linear. And the reduction was just a simple graph, Alice's graph was just a matching of  $n + 1$  edges and Bob's graph is just a bunch of universal vertices. So, we by construction we ensure that Bob's graph does not have a matching of size  $n + 1$  and Alice's graph has a size of matching of size  $n + 1$ , and the key thing was that, if the strings were disjoint then Alice will always output  $d_1 d_2$ .

If the strings were not disjoint then Alice could potentially output other things or Alice is sure to be able to output at least one more string or one more edge. Alice has other option, so the probability of outputting  $d_1 d_2$  is at most half and which could be further improved by boosting. The protocol is if Alice and Bob output  $d_1 d_2$ , then say it is disjoint. If they do not output  $d_1 d_2$  and output something else it is not disjoint.

And that completes the summary of this proof that monotone depth complexity of matching is at least linear. Just to summarize what we saw this week, you saw communication complexity, what is communication complexity, the communication complexity model. Then we saw a bunch of techniques we saw the protocol tree, we saw the matrix, we saw lower bound techniques such as the tree in the bound on the number of monochromatic rectangles.

Then we saw fooling set, then we saw size base bound, and we saw a fooling set size based bound, and then we saw rank bound and then we stated the log rank conjecture. We did not prove the rank triangle bound. And, then we saw this connection between circuit complexity, depth complexity and communication complexity by the Karchmer Wigderson theorem and then we saw this application into for matching that computes the lower bound for matching in this lecture.

And that completes the part about communication complexity and also completes week 11. So, that is all, thank you.