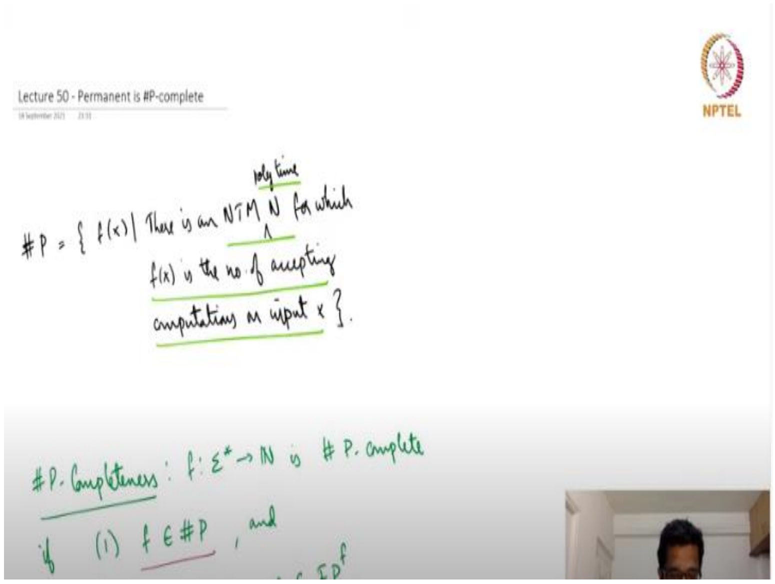
Computational Complexity

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

Lecture - 50
Permanent is #P- Complete: Part 1

(Refer Slide Time: 00:16)



Lecture 50 - Permanent is #P-complete

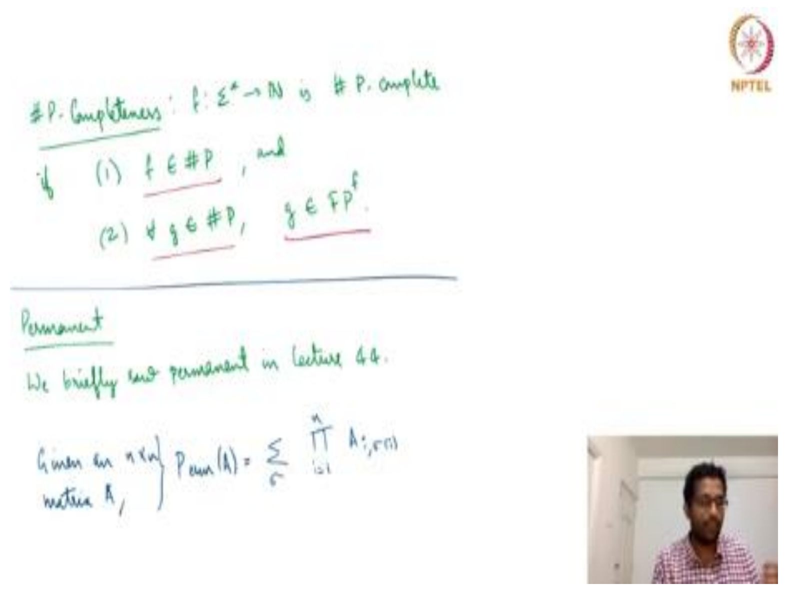
$\#P = \{ f(x) \mid$ There is an $\overset{\text{poly time}}{\text{NTM}}$ N for which f(x) is the no. of accepting computations on input x $\}$.

#P-Completeness: $f: \Sigma^* \to \mathbb{N}$ is #P-complete if (1) $f \in \#P$, and

Hello and welcome to lecture 50 of the course on computational complexity. This is also the beginning of week 10. In the preceding lectures we saw the complexity class sharp P which corresponds to the class of functions that correspond to the number of accepting computations of a non-deterministic polynomial time turing machine. So some of the examples that we saw were counting the number of satisfying assignments;

Counting the number of perfect matchings, counting the number of spanning trees, counting the number of cliques of a certain size and so on. These are all functions that are in sharp P.

(Refer Slide Time: 01:03)



#P-Completeness: $f: \Sigma^* \to \mathbb{N}$ is #P-complete

if (1) $f \in \#P$, and

(2) $\forall g \in \#P$, $g \in FP^f$.

Permanent

We briefly saw permanent in lecture 44.

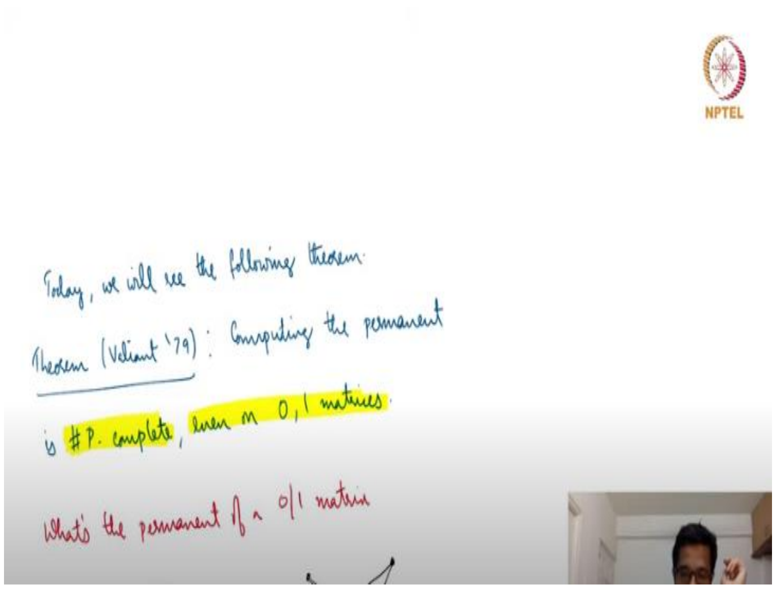
Given an $n \times n$ matrix A, $\Big\}$ $Perm(A) = \sum_{\sigma} \prod_{i=1}^{n} A_{i,\sigma(i)}$

We also saw sharp P completeness which was defined to be the functions that are in sharp P and such that all the functions in sharp P can reduce to this function. So if f is sharp P complete if f is in sharp P and all the g in sharp P reduces to f in the sense of an oracle reduction or a turing reduction. So now in this lecture we want to see an example of a sharp P complete language. So we already saw some examples.

We saw that a sharp SAT counting the number of satisfying assignments is sharp P complete. We also mentioned that many other reductions such as counting the number of cliques, counting the number of independent sets and so on are sharp P complete without proofs. I just mentioned that the proofs that show NP completeness if you look at it closely you will realize that they also preserve the number of each satisfying assignment that corresponds to a certain clique and so on.

So preserving the number was said to be called was defined as being parsimonious. In today's lecture we will see a language called permanent, rather a function called permanent which is a function on a matrix something like a determinant but slightly different. And you will see even though it is in some ways it is only a small difference. But in some ways it is a big difference. We will see the definition and then we will see that permanent is in sharp P and is also sharp P complete.

(Refer Slide Time: 02:50)

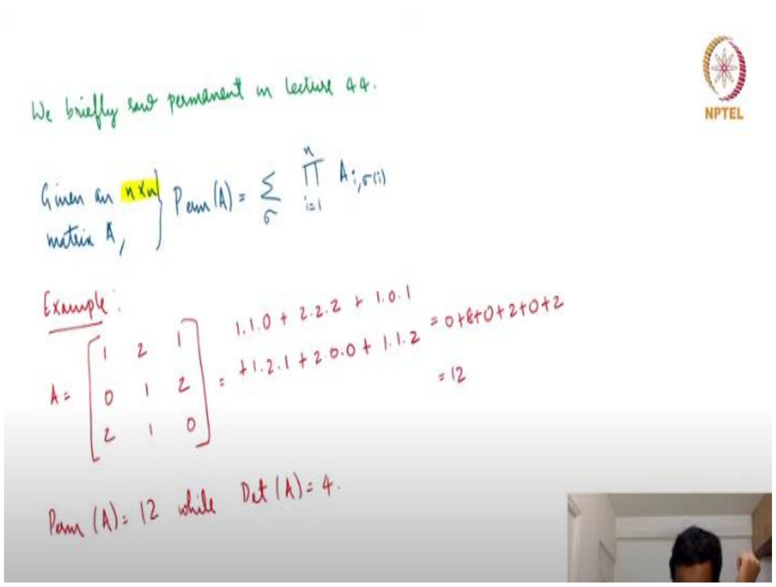

Today, we will see the following theorem.

Theorem (Valiant '79): Computing the permanent is #P. complete, even on 0,1 matrices.

What's the permanent of a 0/1 matrix

And this was Valiant's theorem from 1979 and that computing the permanent is sharp P complete even on 0, 1 matrices. So even restricted to the 0, 1 matrices computing the permanent is sharp P complete. So we will see the proof of that today.

(Refer Slide Time: 03:02)


We briefly saw permanent in lecture 44.

Given an $n \times n$ matrix A, $\text{Perm}(A) = \sum_{\sigma} \prod_{i=1}^{n} A_{i,\sigma(i)}$

Example:

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 2 \\ 2 & 1 & 0 \end{bmatrix} = \begin{array}{l} 1.1.0 + 2.2.2 + 1.0.1 \\ + 1.2.1 + 2.0.0 + 1.1.2 \end{array} = 0+8+0+2+0+2 = 12$$

$\text{Perm}(A) = 12$ while $\text{Det}(A) = 4$.

Maybe some of it may spill over to the next lecture. But we will see the proof today and let us see what permanent is. So we briefly had seen the permanent in lecture 44 where we were trying to check whether a given bipartite graph had a perfect matching. So what is permanent? So permanent it is a summation. So permanent is defined on an n by n matrix. It has to be defined in a square matrix. So it is defined as the summation over all permutations and the product of the A i sigma i.

So all the permutations sigma the product of A i sigma i. So this may seem a bit abstract. So let me explain with an example. So consider this matrix here 3 by 3 matrix. So by a permutation I mean how many different ways can we permute the rows and columns. So there are 6 ways. So first is the identity permutation. This permutation where each row 1 is mapped to column 1, row 2 is mapped to column 2 and row 3 is mapped to column 3.

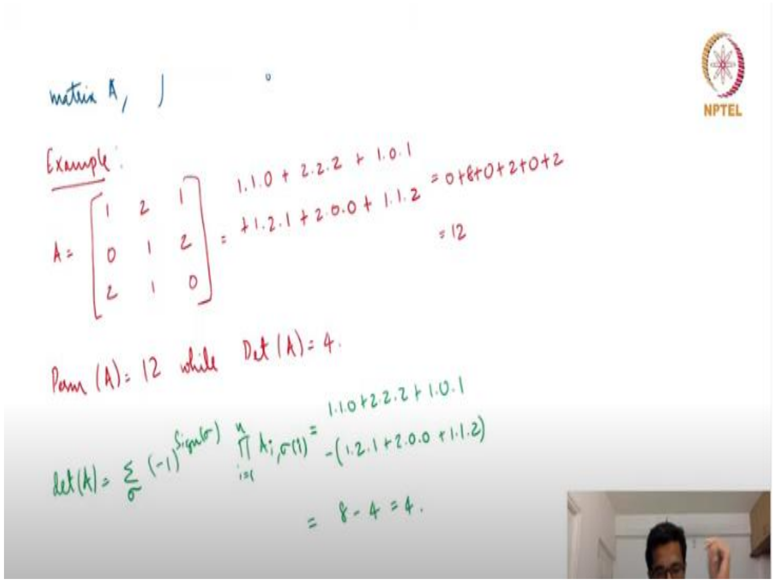
So one time and then take the product of this. 1 times 1 times 0. So that is the first term over here. And the second is the first row maps to the second column, second row maps to third column, third row maps to second column. So that gives you the second 2 2 2. Then the third permutation is first row maps to third column, second row maps to first column and third row maps to second column. So that gives you this 1 0 1.

And then we have first row maps to first column, second row maps to third column, third row maps to second column gives you this 1 2 1. Then the first row maps to the second column, second row maps to first column, third row maps to third column. That gives you this 2 times 0 times 0. And finally we have this. First row maps to third column, third row maps to second column and second row maps to second column, third row maps to first column which is the last term here 1 1 2.

So there are 6 terms, one corresponding to each of the six permutations. So you just add them up. So for each permutation you compute the product of A i sigma i. So the i, sigma ith entry. So 1 1 2 2 3 3 in the first entry and so on. So you get the sum. So if you do the summation you just get 0 plus 6. Not 6, two times two times two is 8 plus 0 times 2 plus 0 times plus 0 plus 2. So 0 plus 8 plus 0 plus 2 plus 0 plus 2 which is nothing but 12.

So this permanent of A is nothing but 12. So this is permanent. It is a rather simple thing once you see how to compute this.

(Refer Slide Time: 06:33)



matrix A,)

Example:

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 2 \\ 2 & 1 & 0 \end{bmatrix} = \begin{array}{l} 1.1.0 + 2.2.2 + 1.0.1 \\ +1.2.1 + 2.0.0 + 1.1.2 \end{array} = 0+8+0+2+0+2 = 12$$

Perm (A) = 12 while Det (A) = 4.

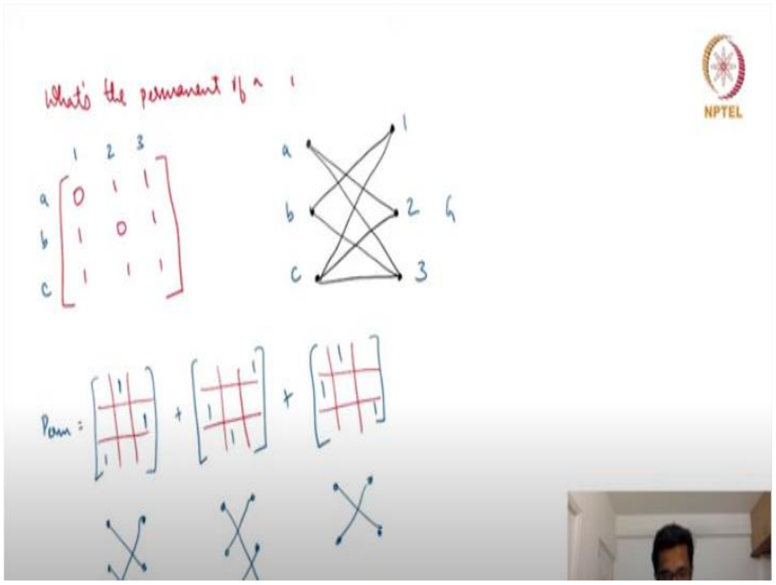
$$\det(A) = \sum_{\sigma} (-1)^{\text{sign}(\sigma)} \prod_{i=1}^{n} A_{i,\sigma(i)} = \begin{array}{l} 1.1.0+2.2.2+1.0.1 \\ -(1.2.1+2.0.0+1.1.2) \end{array} = 8-4=4.$$

Now how do you compute the determinant? So just to see what is the determinant maybe I will just use another colour. So the determinant of A is the same thing but with A is the same definition summation over all the permutations. But then there is a sine component minus 1 whole power sine of sigma. Then i equal to 1 to n, A i sigma i. So it is the only thing that we do differently is introduce a sign.

So if you include the sign here in the above summation it will be so you can check this. So there is something called odd permutations and even permutations. So odd permutations will get a negative symbol and even permutations the sign will be even. So it will be plus 1. So it will be just 1 times 1 times 0 plus 2 times 2 times 2. So in fact whatever I have written in the first row are the even permutation.

So they are the same. The second row, are the odd permutations. So they will get a negative sign. So I am just writing the negative sign outside. So this you can see that is nothing but 8 minus 4 is equal to 4. So this is the determinant computation. So the permanent of this matrix is 12 while the determinant is 4. So this hopefully is and the definition of permanent is clear. And as I said already today we will see that the result that computing a permanent of a matrix is a sharp P complete problem even when the matrix is a 0, 1 matrix.

(Refer Slide Time: 08:30)



So there are a couple of interpretations that will help us. So one is again this interpretation we had already used in the lecture 44, 45 where we saw that bipartite perfect matching is in R and C. So one is that we could view this matrix of 0 1 matrix as an adjacency matrix of a bipartite graph. So let us say a, b, c be three vertices that correspond to the rows. We are viewing it as an adjacency matrix of a bipartite graph. So a, b, c are let us say are these vertices and 1, 2, 3 be the vertices corresponding to the columns.

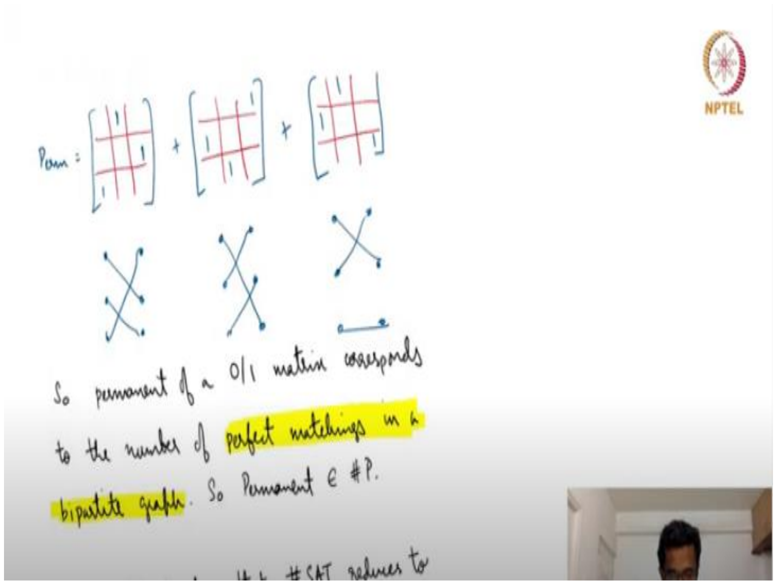
So a does not have an edge to 1, a has an edge to 2 and 3, b has an edge to 1 and 3, c has an edge to all of 1, 2,and 3 which gives us this graph that is on the right hand side. Now what are the terms here? So you see many terms are 0 but what are the terms that remain? The terms that remain are maybe I will just write this down. The terms that remain are the terms that contribute to the permanent rather than I will just write the 1 1 1 plus 1 1 1.

So I am just writing the terms that contribute and so all the star entries are the ones that contribute. So strictly speaking this is not the sum of the matrices. So maybe I will just erase this. So permanent is equal to this 1 1 1. So I hope it is clear when I write it like this the term corresponding to this. Then there is another term 1 1 1 and finally one more term. It will be I think 1 1 1 1 1 and this one here. And you can check that these are indeed the only three non-zero terms.

And what happens is that if you look at it these are the three non-zero terms and these three correspond to we will just make a bit more space. These three correspond to perfect matchings of the above graph. So the first one corresponds to a matches 2, b matches 3, c matches to 1. The second one corresponds to a matches to 3, b matches to 1, c matches to 2 and the third one corresponds to this a matches to 2, b matches to 1 and c matches to 3.

So these are the three perfect matching's that these correspond to and you can see that these are the only three perfect matching's of this graph.

(Refer Slide Time: 12:34)



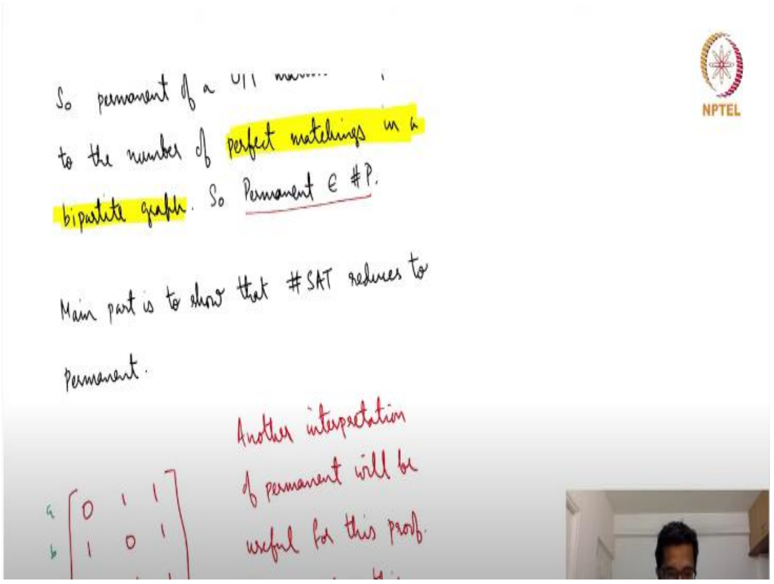
$$\text{Perm} = \begin{bmatrix} & 1 & \\ & & 1 \\ 1 & & \end{bmatrix} + \begin{bmatrix} & & 1 \\ 1 & & \\ & 1 & \end{bmatrix} + \begin{bmatrix} & 1 & \\ 1 & & \\ & & 1 \end{bmatrix}$$

So permanent of a 0/1 matrix corresponds to the number of perfect matchings in a bipartite graph. So Permanent $\in$ #P.

#SAT reduces to

So the permanent of a 0 1 matrix corresponds to the number of perfect matching's in a bipartite graph. So if you view the 0 1 matrix as an adjacency matrix of the bipartite graph. So this is one way to view that the permanent of a 0 1 matrix is actually in sharp P. Because in fact I think I had already mentioned this as an example in the previous lecture. So permanent is in sharp P.

(Refer Slide Time: 12:58)


So permanent of a 0/1 matrix

to the number of perfect matchings in a bipartite graph. So Permanent $\in$ #P.

Main part is to show that #SAT reduces to permanent.

Another interpretation of permanent will be useful for this proof.

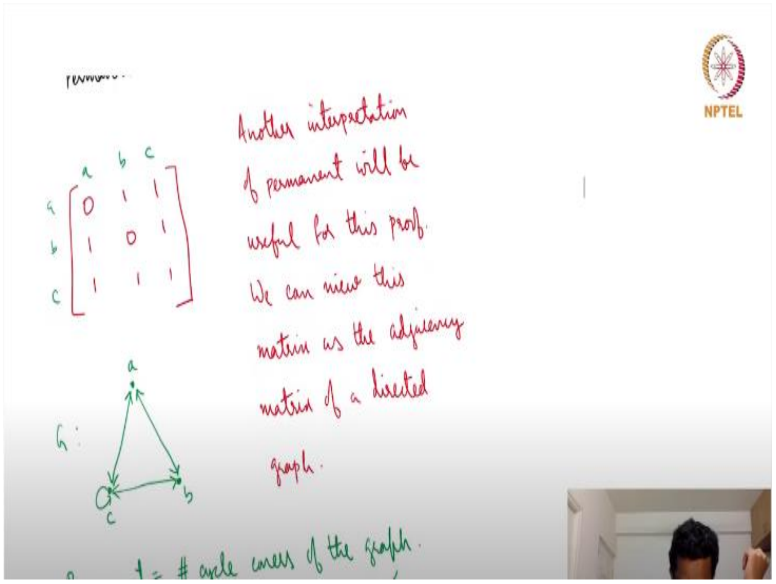
$$\begin{array}{c|ccc} a & 0 & 1 & 1 \\ b & 1 & 0 & 1 \end{array}$$

And as with NP completeness proofs to show that it is in sharp P in this case is not that difficult. The main part is to show that any language or any function in sharp P reduces to the computational permanent and that too even for a 0 1 matrix. And instead since we have already established that sharp SAT is a sharp P complete function we will show that sharp SAT reduces to permanent.

And in fact what we would actually be showing is a sharp 3 SAT reduces to permanent. In fact you can check that even a sharp 3 SAT is maybe I will just write this here. Sharp 3 SAT is also in sharp P complete. So I already mentioned that sharp SAT is in sharp P complete. The same proof applies for sharp 3 SAT. Basically it is about any Boolean formula you can reduce it to or the Boolean formula that you get as a result of Cook-Levin theorem you can convert it into a CNF formula and then a 3 CNF in the formula.

So that is about this. It is not really we had done this at the end of the proof of Cook-Levin theorem to show that 3 SAT is also NP complete. Same proof works here as well and what we will do today is to show that sharp 3 SAT reduces to the permanent. So meaning if you can compute the permanent of a 0 1 matrix you can also compute the number of satisfying assignments of a 3 SAT instance, 3 CNF formula.

(Refer Slide Time: 15:01)



$$\begin{array}{c|ccc} & a & b & c \\ \hline a & 0 & 1 & 1 \\ b & 1 & 0 & 1 \\ c & 1 & 1 & 1 \end{array}$$

Another interpretation of permanent will be useful for this proof. We can view this matrix as the adjacency matrix of a directed graph.

= # cycle covers of the graph.

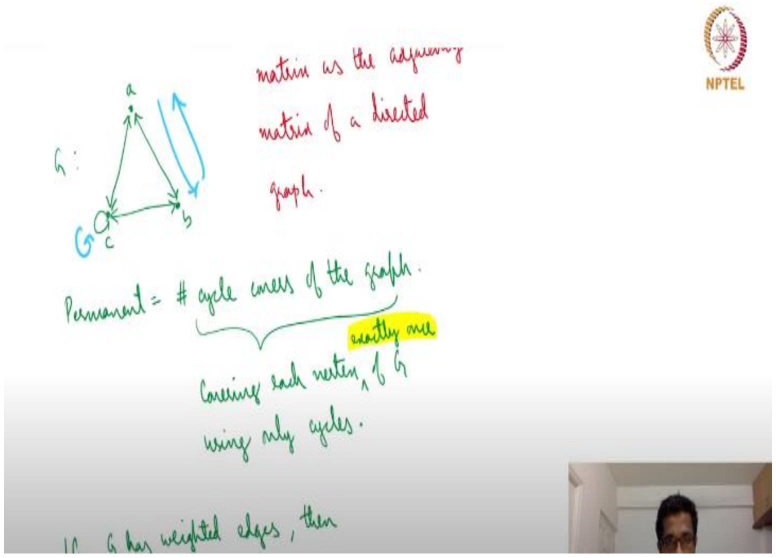
And for this another interpretation of the permanent will be useful. What is that interpretation? The interpretation is that we can view this graph sorry this matrix as an adjacency matrix of a general graph, not necessarily bipartite but a directed graph. So if you view this as an adjacency matrix of a directed graph let us see what we get. So the rows are labeled a, b, c and so are the columns because it is a three vertex graph.

So a has an outgoing edge to b and c which I have drawn here right below. b has an outgoing edge to a and c. c has an outgoing edge to a, b and c. So all a, b, c have edges going to each other and then in addition c has a self loop which is what I have drawn here. You can see that. So this is the graph that corresponds to this adjacency matrix and in this case of a general directed graph you can see that each non-zero term of the permanent corresponds to a cycle cover.

So for instance this corresponds to a non-zero term and actually this is a cycle cover. So a going to b. So maybe I will just draw it slightly here. a going to b, b going to c and c go into a. So this corresponds to one cycle cover. This is another term that corresponds to the permanent and this corresponds to the cycle cover a going to b, b going to a back. So it is a cycle of length 2 and then c part of a self loop. So this corresponds to another cycle cover.

So each non-zero term of the permanent here corresponds to a cycle cover. So what is the cycle cover? Cycle cover is a covering of each vertex exactly once. It is a covering of the entire graph with cycles where each vertex is covered exactly once.

(Refer Slide Time: 17:40)



matrix as the adjacency matrix of a directed graph.

Permanent = # cycle covers of the graph.

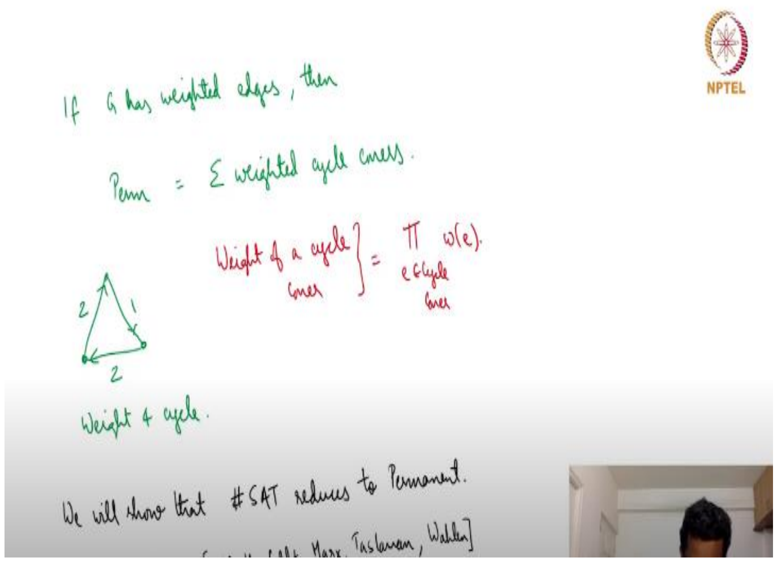
Covering each vertex of G exactly once using only cycles.

If G has weighted edges, then

So you saw that in this case it was like sorry maybe I will just erase this. In this case it was like a going to b. This is one cycle cover, c self loop. So c gets covered by the self loop and a and b as a two vertex cycle or we also saw another cycle where a, b, c are covered by a single cycle. The key thing to note is that each vertex is covered exactly once. Every vertex should be covered and every vertex should be covered once and once only.

Not more and the entire cover should be using cycles. You cannot have a path or a standalone edge or anything like that. And in fact this is 0 1. This interpretation is going to be key to our proof that we are going to see. In fact so this was for a 0 1 matrix it is just the number of cycle covers of this graph.

(Refer Slide Time: 18:37)



If G has weighted edges, then

Perm = Σ weighted cycle covers.

Weight of a cycle cover $\Big\} = \prod_{e \in \text{cycle cover}} w(e)$.

Weight 4 cycle.

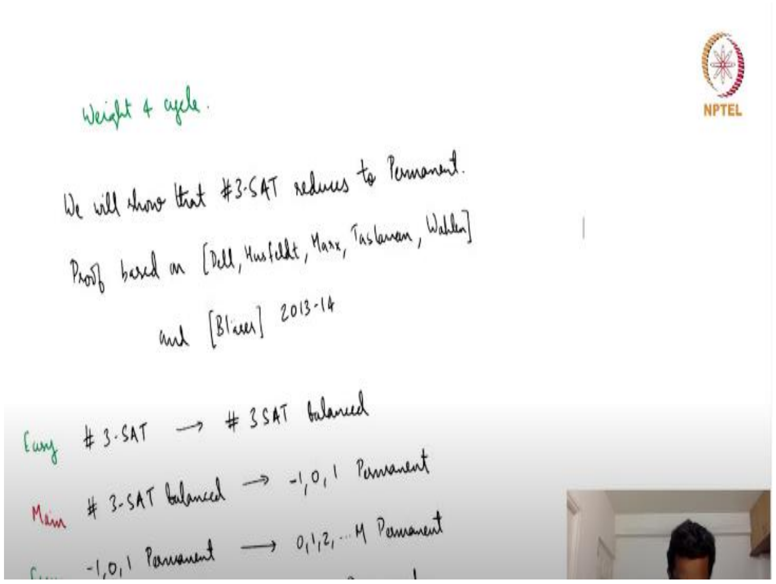
We will show that #SAT reduces to Permanent.

Marx, Taslaman, Wahlen]

If G has weighted edges let us say the c, b, h or or let us say the self loop had weight 2 that would have corresponded to the entry here being replaced by 2 if the self loop had weight 2. The c self loop will be the corresponding entry in the adjacency matrix and in that case the permanent will be the sum of the weighted cycle covers. So the weight of a cycle cover is nothing but the product of e in cycle cover weight of e. In fact this number of weighted cycle covers does not make sense. So I will just say the sum of weighted cycle cover.

So permanent is just the sum of the weight of a cycle cover over all the cycle cover. So here this is a weight 4 cycle that I have written over here because it is 2 times 2 times 1. So if the graph has weighted edges it will not be a 0 1 matrix and the permanent will correspond to the sum of the weighted cycle covers. These interpretations of this extension are also going to be useful in the proof.

(Refer Slide Time: 20:07)


Weight 4 cycle.

We will show that #3-SAT reduces to Permanent.

Proof based on [Dell, Husfeldt, Marx, Taslaman, Wahlen]

and [Bläser] 2013-14

Easy # 3-SAT $\longrightarrow$ # 3SAT balanced

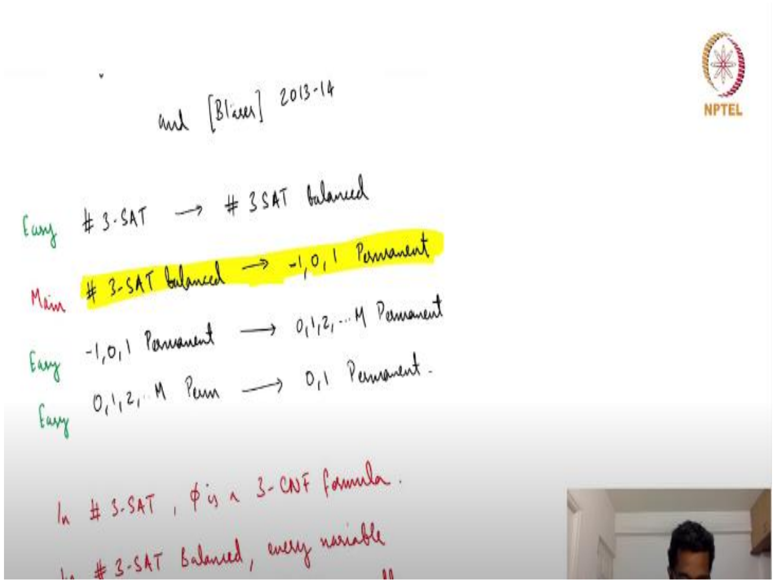
Main # 3-SAT balanced $\longrightarrow$ -1, 0, 1 Permanent

Easy -1, 0, 1 Permanent $\longrightarrow$ 0, 1, 2, ... M Permanent

So what we are going to show is that sharp SAT in fact I am making this correction sharp 3 SAT reduces to the permanent. And this is not the exact same thing as Valiant's proof. Even though in spirit it is the same there are some small changes in the gadgets. So this is based on two papers. So one by Dell, Husfeldt, Marx, Taslaman and Wahlen and one by Blaser. I think 2013 and 2014. Both the papers came roughly at the same time.

Perhaps they may even have consulted each other while arriving at these gadgets. So in fact those papers talk much more but in fact these gadgets will I feel helps to simplify the presentation. But it is essentially the same idea. It is just some minor tweaks in the construction.

(Refer Slide Time: 21:06)


and [Bläser] 2013-14

Easy # 3-SAT $\longrightarrow$ # 3SAT balanced

Main # 3-SAT balanced $\longrightarrow$ -1, 0, 1 Permanent

Easy -1, 0, 1 Permanent $\longrightarrow$ 0, 1, 2, ... M Permanent

Easy 0, 1, 2, .. M Perm $\longrightarrow$ 0, 1 Permanent.

In # 3-SAT, ϕ is a 3-CNF formula.

In # 3-SAT balanced, every variable

So the proof will be using gadgets like we saw in the case of the Hamiltonian cycle. For instance, we had some reasonably complex graphs which had some things correspond to clauses, something correspond to variables. And we drew a correspondence between the formula being satisfiable and there exists a satisfying assignment. In fact if you recall the proof of the Hamiltonian cycle.

You may even recall that the number of Hamiltonian cycles of that particular graph that we constructed actually corresponds to the number of satisfying assignments. So it was not just a proof that the formula is satisfiable if and only if there is a Hamiltonian cycle. In fact the number of satisfying assignments would actually each satisfying assignment correspond to a Hamiltonian cycle.

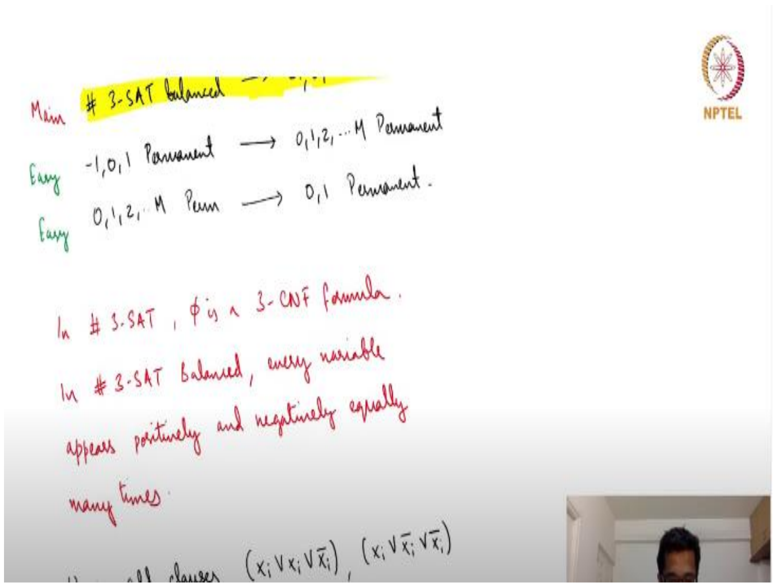
So it was a parsimonious reduction. In any case we will use similar gadgets in a similar spirit. So we will draw a graph which has these gadgets and we will show that the permanent of this graph corresponds to some number from which we can get the number of satisfying assignments. So we can roughly divide the proof into four parts. One is that we reduce sharp 3 SAT into sharp 3 SAT balanced.

Sharp 3 SAT balanced means it is not just a 3 SAT instance it satisfies some other additional constraint. Then this is going to be the main part. The reduction from sharp 3 SAT balanced into minus 1 0 1 permanent meaning we are reducing the computation of the number of satisfying assignments of a sharp 3 SAT balanced instance into the computation of a permanent of a matrix with 0 minus 1 and plus 1 entries. Remember we promise that we will reduce it to 0 1 permanent.

But we first go to minus 1 0 1. This is going to be the main part of the proof. Once we have reduced it to minus 1 0 1 permanent then we show how to move to 0 1 permanent by also going through an intermediate step which is 0 1 2 up to m permanent where m is some kind of positive number. So we move to a positive range. The weights are from a positive range and then we move to 0 1.

So the first third and fourth steps are rather easy and straightforward and the main part of the proof will be the second step. So let us go and do it in the order.

(Refer Slide Time: 23:57)



Main $\#$3-SAT balanced $\rightarrow$

Easy $-1,0,1$ Permanent $\longrightarrow$ $0,1,2,\ldots M$ Permanent

Easy $0,1,2,\ldots M$ Perm $\longrightarrow$ $0,1$ Permanent.

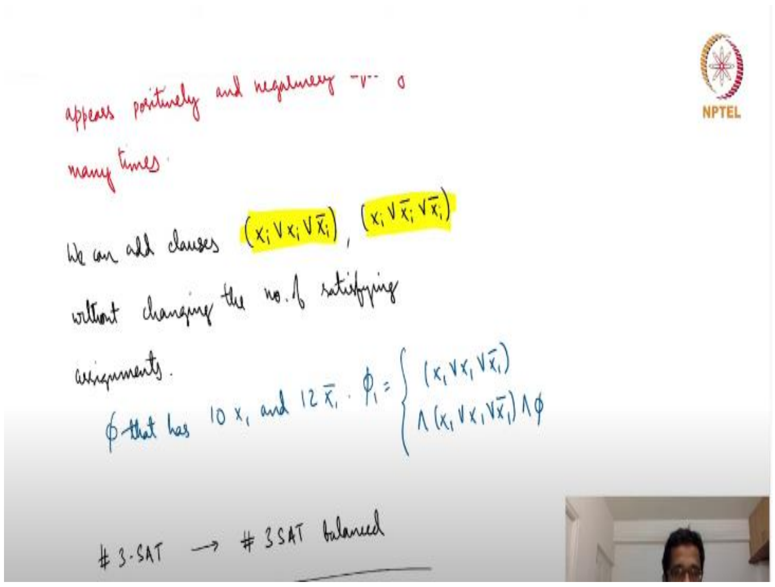
In $\#$3-SAT, ϕ is a 3-CNF formula.

In $\#$3-SAT balanced, every variable appears positively and negatively equally many times.

$(x_i \vee x_i \vee \bar{x}_i)$, $(x_i \vee \bar{x}_i \vee \bar{x}_i)$

So in sharp 3 SAT phi is a 3 CNF formula. What is sharp 3 SAT balanced? Sharp 3 SAT balanced is a 3 CNF formula where every variable appears the same number of times positive and negative.

(Refer Slide Time: 24:23)



appears positively and negatively equally many times.

We can add clauses $(x_i \vee x_i \vee \bar{x}_i)$, $(x_i \vee \bar{x}_i \vee \bar{x}_i)$ without changing the no. of satisfying assignments.

ϕ that has $10\ x_1$ and $12\ \bar{x}_1$. $\phi_1 = \begin{cases} (x_1 \vee x_1 \vee \bar{x}_1) \\ \wedge (x_1 \vee x_1 \vee \bar{x}_1) \wedge \phi \end{cases}$

$\#$3-SAT $\rightarrow$ $\#$3SAT balanced

So this is easy. The reduction is rather easy because if you look at these two clauses x i or x i or x i negation, if you add it to a formula the formula's truth value does not change. If the formula was satisfiable let us say it was satisfiable by setting x i to false. You can add it. It still continues to satisfy. You can add this as a clause. Another clause is x i or x i complement or x i complement. You can add this clause as well. The truth value does not change.

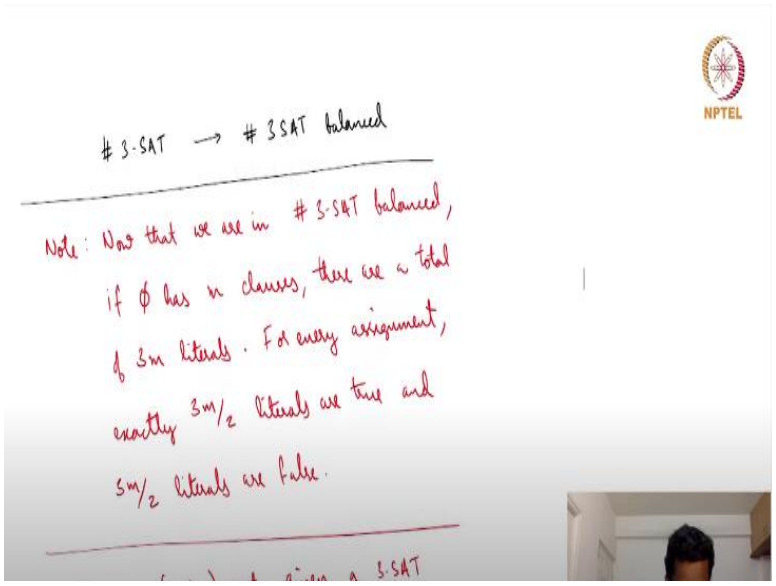
But what happens is the number of times x i appears is now and the number of times x i complement appears changes when you add these clauses. So if it is not balanced you can add the appropriate one appropriate number of times to make it balanced. So for instance if you have let us say phi that has 10 x 1 and let us say 12 x 1 complement. So x 2, x 3 etcetera are there. So then what do you do? You can add. So then you add x 1 or x 1 or x 1 complement and x 1 or x 1 or x 1 complement and phi.

So let phi 1 equal to this. So now in phi 1 the number of times x 1 appears is 14 because phi has 10 x 1 and phi 1 has 14 x 1. And the number of times x 1 bar appears is 14 because phi has 12 and we added 2. So this is it. So as far as x 1 is concerned this is balanced and like that we can balance out each variable. Notice that we do not need to make everything 14. So x 1 may be the balancing value may be 14, for x 2 it may be 5, for x 3 it may be 2, for x 4 it may be 25 whatever.

So what I am saying is for each x i the number of positive x i appearing and x i complement appearing should be equal. This is important. This is what I mean by 3 SAT balanced and the reduction. This is just the reduction. So now you have seen how to reduce sharp 3 SAT to sharp 3 SAT balanced. And you can also see that any satisfying assignment remains a satisfying assignment any falsifying or unsatisfying assignment remains an unsatisfying assignment.

So as far as the number of satisfying assignments is concerned there is absolutely no change. So from phi we have gone to a balanced formula without changing the number of satisfying assignments. So that is the first part. Now let us come to the main part. How do we reduce the computation of sharp 3 SAT balanced to a permanent of minus 1 0 and 1 values.

(Refer Slide Time: 27:49)



$\# 3\text{-SAT} \longrightarrow \# 3\text{SAT balanced}$

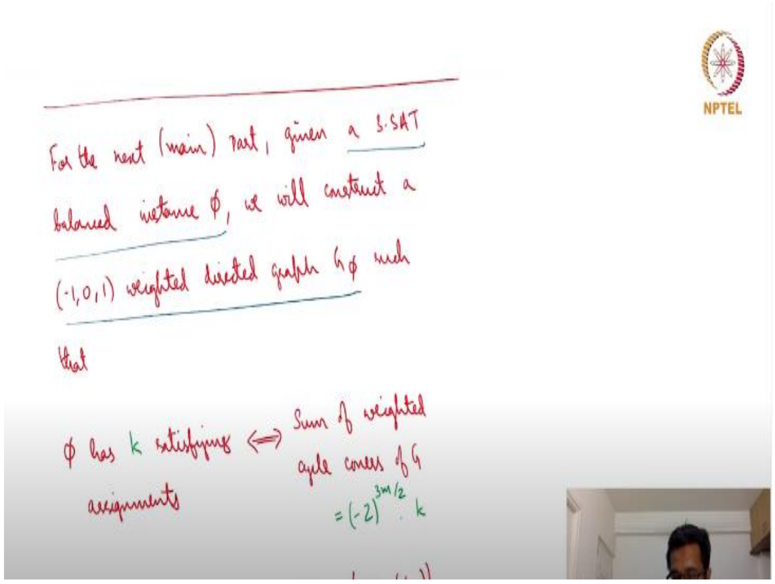
Note: Now that we are in # 3-SAT balanced, if ϕ has m clauses, there are a total of 3m literals. For every assignment, exactly $3m/2$ literals are true and $3m/2$ literals are false.

So let us say phi has m clauses. Then there are a total of 3 m literals because each clause contains exactly 3 laterals. And now we know that if you take any variable the number of times it appears positive as well as negative is the same. So you take any satisfying assignment what will be the number of literals or you take any assignment to get satisfying. What will be the number of literals that this assignment sets to true?

So if x 5 is a literal and if x 5 is set to true then x 5 is a positive. It is a true literal. If x 5 is set to false then x 5 is a false literal. Suppose the literal is x 5 complement then setting x 5 to false makes it a true literal and x 5 to true makes it x 5 complement a false literal. So whatever it is the point is that the number of true literals and the number of false literal is equal for each variable and hence it is also equal across all the variables and across the entire formula.

So since there are 3 m literals for any assignment exactly 3 m literals will be true and 3 m literals will be false. So if you change the assignment some true literals will become false, some false literals will become true. But still the number of true literals and number of false literals will be exactly equal to 3 m by 2 and 3 m by 2. The assignments may be satisfying or not satisfying. But this will always be the case. So exactly 3 m by 2 literals are true and 3 m by 2 literals are false.

(Refer Slide Time: 29:51)

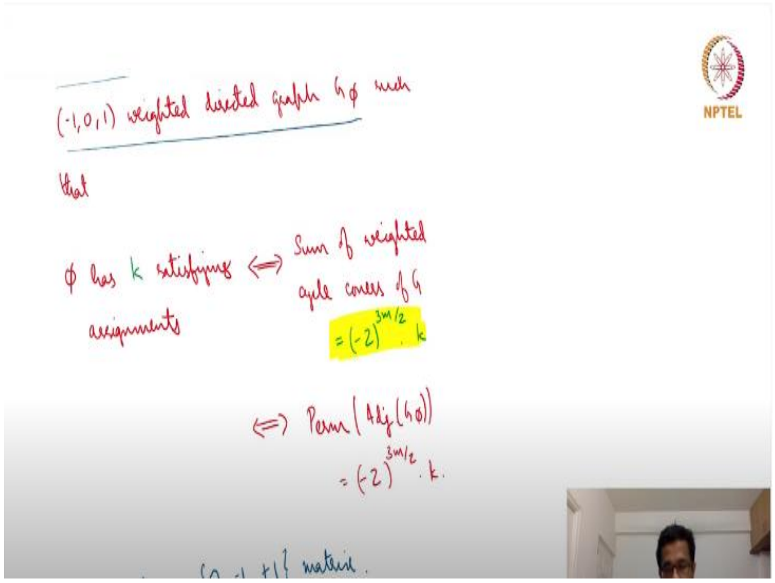


For the next (main) part, given a 3-SAT balanced instance ϕ, we will construct a $(-1,0,1)$ weighted directed graph G_ϕ such that

ϕ has k satisfying assignments $\iff$ Sum of weighted cycle covers of G $= (-2)^{3m/2} \cdot k$

Now what we will do is given the 3 SAT balanced instance phi we will construct a weighted directed graph G phi and the weights will be 0 plus 1 and minus 1. In fact it will be mostly zeros and ones and there will be some minus 1 for the gadget such that if phi has k number of satisfying assignments then the sum of the weighted cycle covers will be equal to minus 2 whole power 3m by 2 multiplied by k.

(Refer Slide Time: 30:24)



$(-1,0,1)$ weighted directed graph G_ϕ such that

ϕ has k satisfying assignments $\iff$ Sum of weighted cycle covers of G $= (-2)^{3m/2} \cdot k$

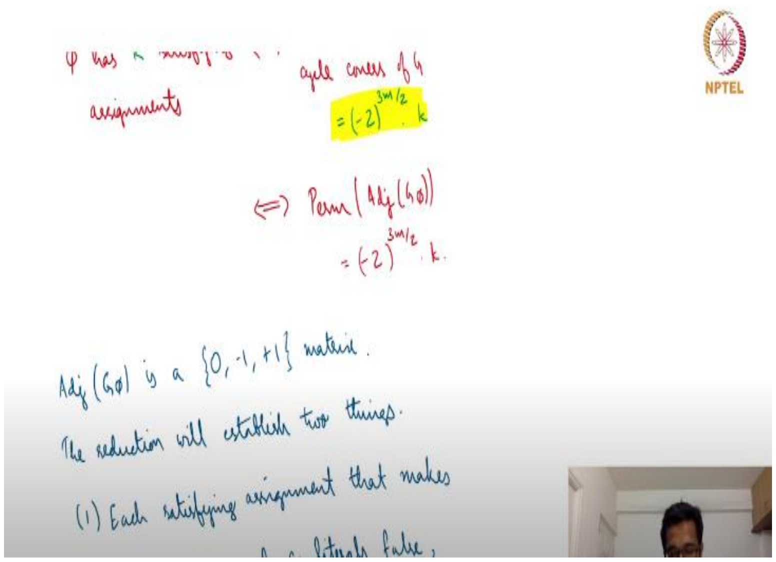
$\iff \text{Perm}(\text{Adj}(G_\phi)) = (-2)^{3m/2} \cdot k.$

$\{0,-1,+1\}$ matrix.

This will be the number of sum of weighted cycle covers which is actually equal to the permanent also. This is equal to permanent. In fact I have written below which is equal to the permanent of the adjacency matrix of G phi. So if you compute the permanent all you have and you know that based on the number of clauses you can easily compute minus 2 power 3 m divided by 2 and you can divide by that to compute the value k and you can compute the number of satisfying assignments.

So if you can compute the permanent you just divided by this minus 2 power this quantity and get k which is the number of satisfying assignments. So that completes the reduction. So now the main part is going to be how do we construct this graph?

(Refer Slide Time: 31:27)



φ has k satisfying assignments

cycle covers of G $= (-2)^{3m/2} \cdot k$

$\Leftrightarrow$ Perm(Adj(G_φ)) $= (-2)^{3m/2} \cdot k.$

Adj(G_φ) is a $\{0, -1, +1\}$ matrix.

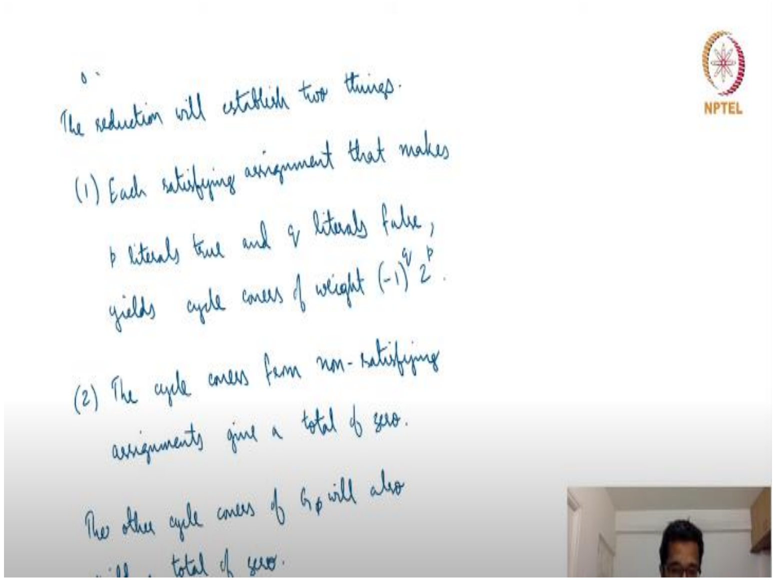
The reduction will establish two things.

(1) Each satisfying assignment that makes

literals false,

And the adjacency matrix of G phi will be a 0 minus 1 and plus 1 matrix, a matrix containing exactly these things as entries.

(Refer Slide Time: 31:37)



The reduction will establish two things.

(1) Each satisfying assignment that makes p literals true and q literals false, yields cycle covers of weight $(-1)^q 2^p$.

(2) The cycle covers from non-satisfying assignments give a total of zero.

The other cycle covers of G_φ will also

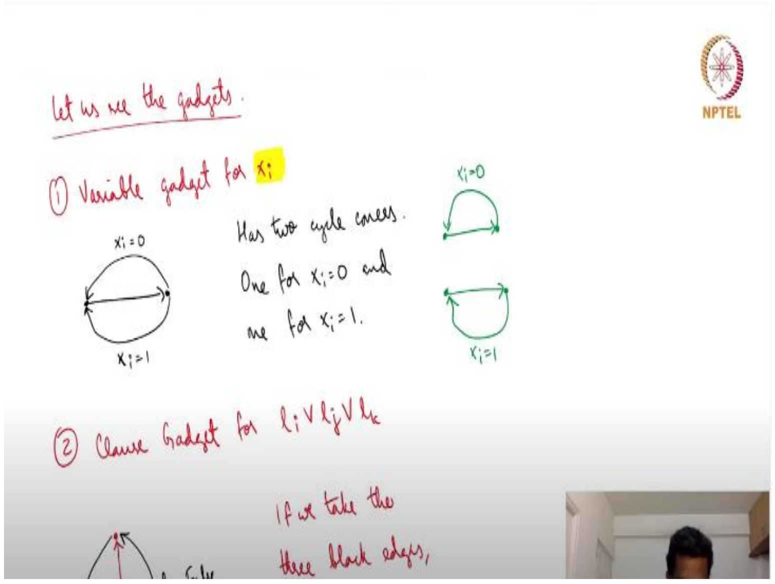
total of zero.

So what we will show is this. Every satisfying assignment that makes p literals true and q literals false it will yield a cycle cover of weight minus 1 whole power q multiplied by 2 whole power p. So where p is the number of true literals and q is the number of false literals and we have already said that in any assignment the number of true literals and number of false literals are actually 3 m by 2.

So this will just boil down to in fact we have seen that in any assignment not just satisfying assignment p equal to q equal to 3 m by 2. So you can see that this is just minus 2 whole power 3 m by 2. And there will be non-satisfying assignments and the cycle covers that correspond to non-satisfying assignments actually give a total of 0 and there will also be other cycle covers that do not correspond to any assignments.

But even they will give a total of 0. So individually they may not be of weight zero but the construction will be such that they will cancel out and the total will be 0. So far so good. Let me explain again. The goal is to give a balanced 3 SAT instance phi where each variable appears in the positive and negative form the equal number of times. To construct a directed graph with weights plus 1 minus 1 and 0 such that the sum of the weighted cycle covers of this graph will be directly related to the number of satisfying assignments of the Boolean formula.

(Refer Slide Time: 33:58)



Let us see the gadgets.

1. Variable gadget for x_i

Has two cycle covers. One for $x_i = 0$ and one for $x_i = 1$.

2. Clause Gadget for $l_i \vee l_j \vee l_k$

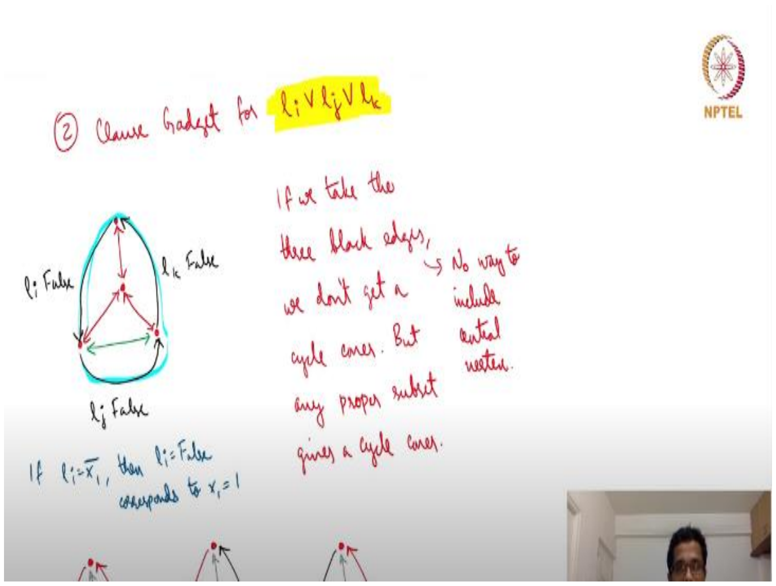
If we take the three black edges,

So let us see the gadgets. So one point I want to mention here is that the edges with weight 0 we will just consider them as edges that are not there. So edges that are not there are all edges of weight 0. So the edges that are there will be edges of weight 1 and of weight minus 1. In fact most of the edges if I do not write the weight by default it is 1. So if it is minus 1 I will explicitly write minus 1. So there will be a variable clause gadget and a consistency gadget.

So we will first explain what the variable and clause gadgets are and then go to the consistency gadget. So the variable gadget is very simple. It contains only two vertices and there are 3 edges. So let us say this is for the variable x i. So there are two edges, one going on the top and one going on the bottom. And all of them just have weight 1. So what are the cycle covers of this particular gadget alone? There are two cycle covers. One is this.

So this goes through the edge marked x i equal to 0 the other cycle cover. So basically there are only two cycles or one cycle is enough. So both vertices are covered in both the cycle covers. So one that includes the bottom edge one that includes the top edge. So the way to understand this or interpret this is that when x i is set to 0 or false we will consider that top cycle cover. And when x i is equal to true or 1 we will consider the bottom cycle cover.

(Refer Slide Time: 36:02)



② Clause Gadget for $l_i \vee l_j \vee l_k$

If we take the three black edges, we don't get a cycle cover. But any proper subset gives a cycle cover.

→ No way to include central vertex.

If $l_i = \bar{x}_1$, then l_i = False corresponds to $x_1 = 1$

And two is the clause gadget. Clause gadget is a bit more involved. So let the clause gadget be l i sorry l i or l j or l k. This is the clause where l i, l j, l k are all literals. So each of l i, l j, l k could be some variable. Let us say x 1 or x 2 or something or they are negations. So l i could be x 1. It could be x 1 complement and so on for l j also. And this graph that I have drawn here is the gadget. So there are 3 black edges like the outside.

They correspond to l i being false l j being false and l k being false. So maybe I will just write here if let us say if l i equal to x 1 then l i equal to false corresponds to x 1 equal to false. That is 0. If l i is x 1 complement then l i is false corresponds to x 1 to be 1. So I just write one of them because then if x 1 equal to 1 then l i false means x 1 complement is false. So x 1 is true. So notice that in this particular gadget so why does this correspond to let us try to see why this gadget makes sense.

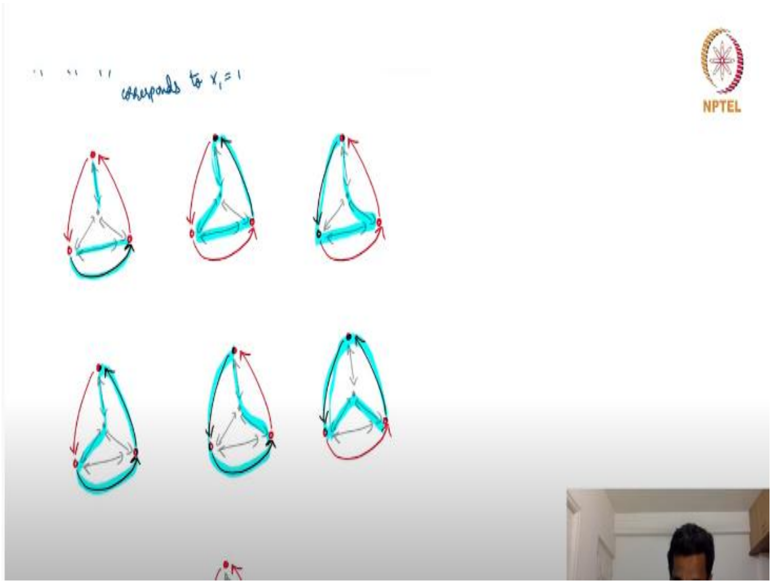
So the variable gadgets make sense because there are two settings for each variable. The true and false. And we had two cycle covers corresponding to that. The clause gadgets make sense because of the following reasons. There are three literals. So if you consider l i, l j, l k assuming they can be controlled independently there is a total of there is only one way in which you can falsify this clause. You must set l i to be false, l j to be false and l k to be false and all the remaining.

So there are total 8 ways to assign l i, l j, l k out of which the remaining seven ways the clause will be true. The one way where all three of them are false the clause will be false. So this will have whatever so corresponding to that whichever way you set l i, l j, l k as long as they are not all false you will get a cycle cover. And all the edges in this graph are of weight 1. So the cycle covers will all be of weight 1. But when l i, l j, l k are all false there will not be any cycle cover.

So let us see this. So here let us say in this particular figure on the left side l i, l j, l k are all false. So which means I consider these 3 black edges. So I consider these 3 black edges and now you can see that there is no way for the cycle cover to include the center vertex. Sorry this has changed the vertex colour so there is no way to include the central vertex. So there is no cycle cover for this gadget if all the 3 outer edges are considered.

Suppose not all the 3 outer edges are considered which means which corresponds to not l i, l j, l k are not all of them are false.

(Refer Slide Time: 40:08)

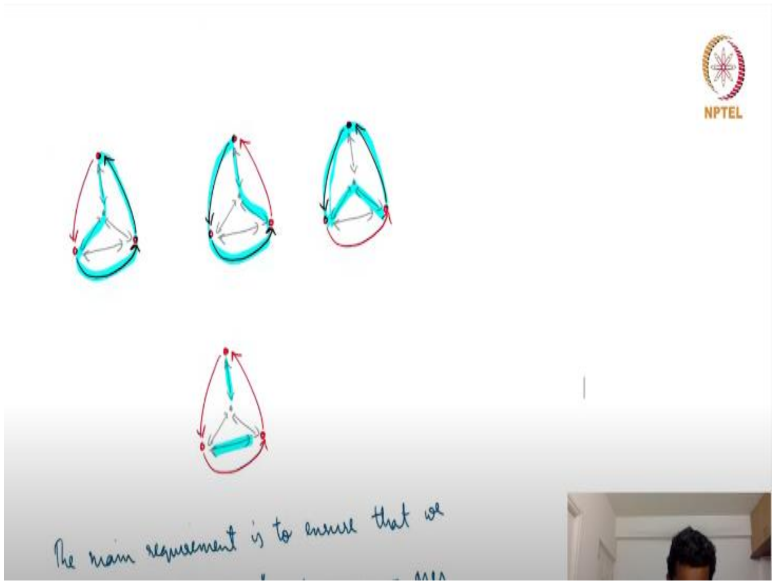


So let us say only one of them is false. Let us say l j, let us say the bottom one is included. So now let us see the cycle cover. If the bottom one is included and so now look at the figure over here. This one. So now the bottom one is included but not the other two. So now let us see what are the cycle covers. So you can include this but you cannot include the other vertices. So what can you do? Other red edges cannot be included.

So one way is to use this edge and then these two can go like a cycle. There are two cycles of length 2. One includes the bottom two vertices and one includes the top and the central vertex. Now look at this one, the second figure. Now I have to include this edge and we can do this and then we can do this without including the red edges. Now something similar can be done here. So there is exactly one cycle cover in each of these cases.

Now consider where two of the l i, l j, l k are chosen. Again there are cycle covers. This one and I think something like this. So there are cycle covers that include all the four vertices of this gadget.

(Refer Slide Time: 42:07)


The main requirement is to ensure that we

And let us say if none of the outside vertices are chosen what you can do is you can do this. So you can again do this. So there is a cycle of length 2 between the bottom two one is the same cycle of time two with the top and central vertex. So just that here in this case we just use this bi-directional edge to complete the cycle without using the bottom red edge. So whichever way you pick right as long as you do not pick all the three black edges there is exactly one cycle cover for this gadget.

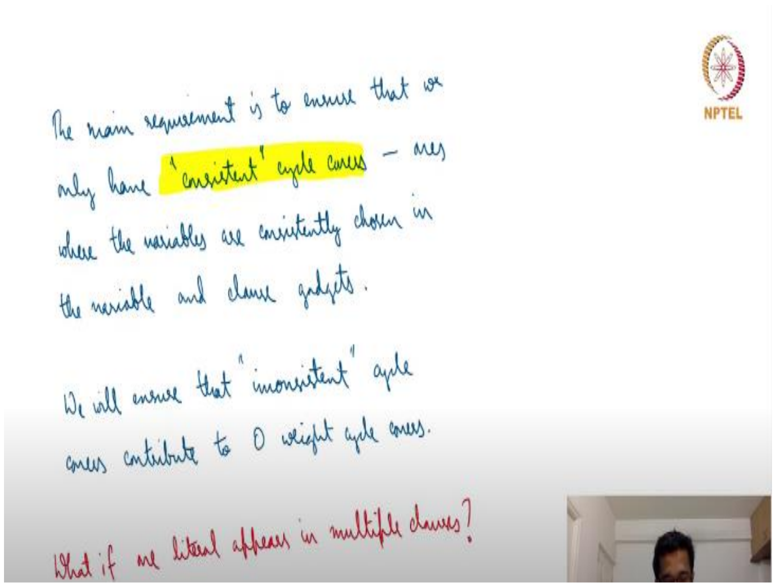
And if you pick all the three outside edges there is no way to make a cycle cover. So now you can see the correspondence. So if you had picked l i, l j, l k all of them or all of them were false there is no cycle cover. But in any other case there is exactly one cycle cover. So now the main idea is you will have a variable gadget and you have these clause gadgets and suppose there was some way to connect this meaning;

If you pick a certain cycle cover from the variable gadget then suppose the same we have to consistently enforce consistently pick the edges from the clause gadgets. When I say edges I mean the outside edges and the ones that correspond to variables being false or true. Now suppose there is such a requirement then you will get cycle covers only when at least one of the outside edges are not picked.

Because if all three outside; edges are picked there is no cycle cover. So you will get a cycle cover only when one or the outside edges are not picked. So only the satisfying assignments will correspond to cycle covers. So till now we have not even talked about all of this at 0 1 edges. We are not talking about minus 1. So there is no need to talk about weights as of now but the main requirement is to somehow connect these.

So to make sure that if you pick x 1 to be true in the variable gadget then correspondingly we should also pick x 1 to be true in the clause gadget. There could be multiple clauses where x 1 appears either in the positive or in the negated form.

(Refer Slide Time: 44:42)



The main requirement is to ensure that we only have 'consistent' cycle covers – ones where the variables are consistently chosen in the variable and clause gadgets.

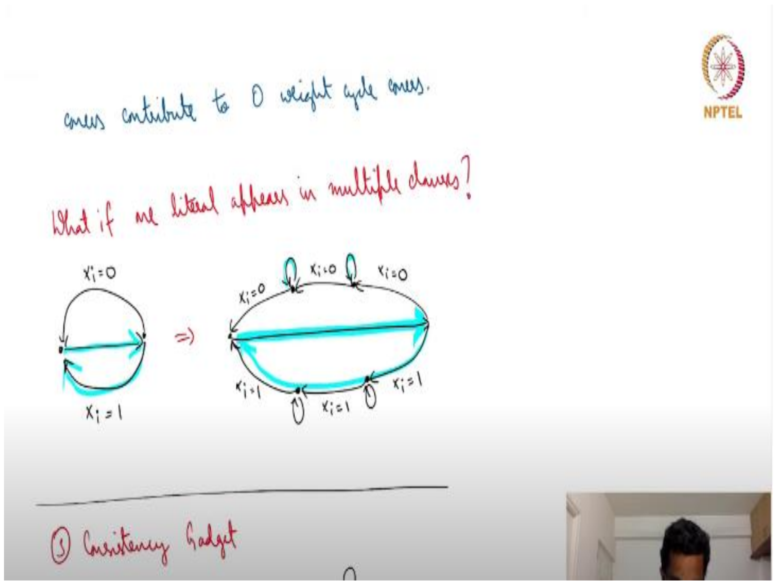
We will ensure that "inconsistent" cycle covers contribute to 0 weight cycle covers.

What if one literal appears in multiple clauses?

So there is a way to maintain consistency. So we need to have consistent cycle covers. So we will ensure consistency using a consistency gadget and we will have to allow for other cycle covers. But what we will do is to make sure that the other cycle covers they contribute to weight 0. So the totals add up to weight 0. And one question that you may ask is one variable may appear in multiple clauses. So x 1 may appear in multiple clauses.

But then there is only one variable gadget. So, multiple clause gadgets cannot enforce consistency with a single variable gadget. So we need multiple copies of this.

(Refer Slide Time: 45:36)


covers contribute to 0 weight cycle covers.

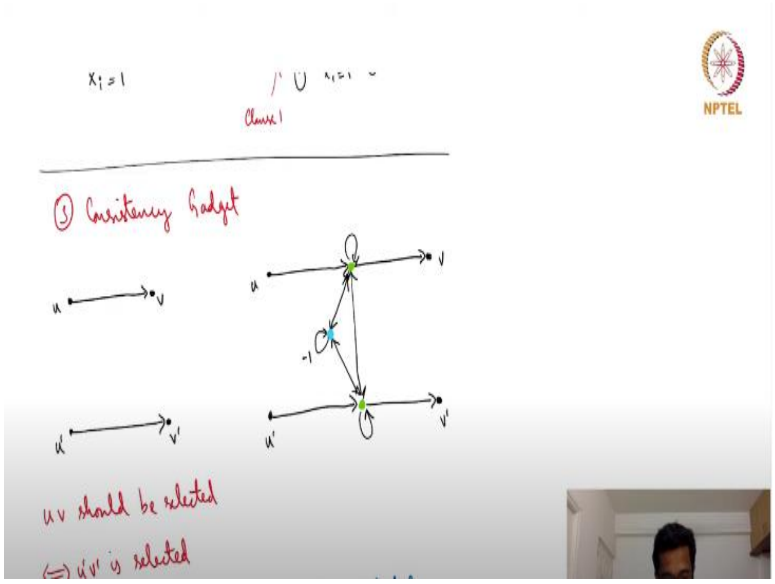
What if one literal appears in multiple clauses?

③ Consistency Gadget

So what we do is we will have different clause gadgets. So we have something like this. So we replace it with this kind of a picture on the right side. So the point here is that if x i was chosen to be let us say we want to pick the top part for this and then the bottom part for let us say we pick the bottom part. x i equal to one part and now there are two additional vertices. But then they can be covered by the self loops. Similarly for let us say if x i equal to 0 I am talking about so this is the correspondence.

Similarly if x i equal to 0 edge was chosen then you will choose the opposite of this and again there are self loops in the bottom. And this particular gadget, the one I have drawn, suppose x i as long as x i appears at most three times. If x i appears four times you can have one more of these self loops and one more edge like this, one more vertex like this. And likewise we can do for each of the variables.

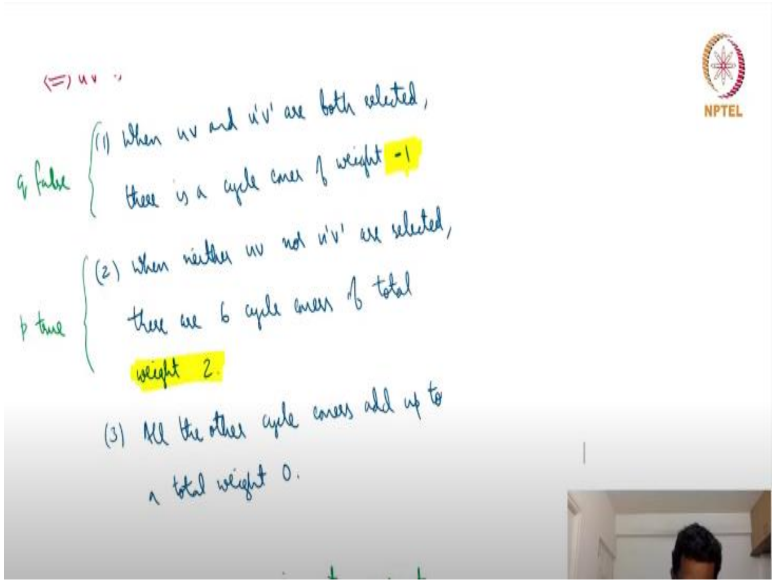
(Refer Slide Time: 46:55)



So now let us move on to the consistency gadget. So the consistency gadget will connect the relevant part of this variable gadget. So let us say this is corresponding to clause 1. It will connect this part. This one says x i equal to 1 with the relevant part of the clause gadget. So the clause gadget will say somewhere x i equal to 1 because maybe there is some literal that where x i was appearing as x i negative or x i complement.

So when that x i complement has to be made false x i has to be 1 and we need to ensure consistency. So basically it boils down to if x i equal to 1 was chosen here we should also choose let us say this edge was corresponding to x i equal to 1. We should also choose this edge from this gadget. So how do we do that? So it boils down to if this edge is chosen the other edge also should be chosen. If this edge is not chosen the other edge also should not be chosen. This is what we want.

(Refer Slide Time: 48:05)



$(\Leftarrow)$ uv

l_i false: (1) When uv and u'v' are both selected, there is a cycle cover of weight -1

l_i true: (2) When neither uv nor u'v' are selected, there are 6 cycle covers of total weight 2.

(3) All the other cycle covers add up to a total weight 0.

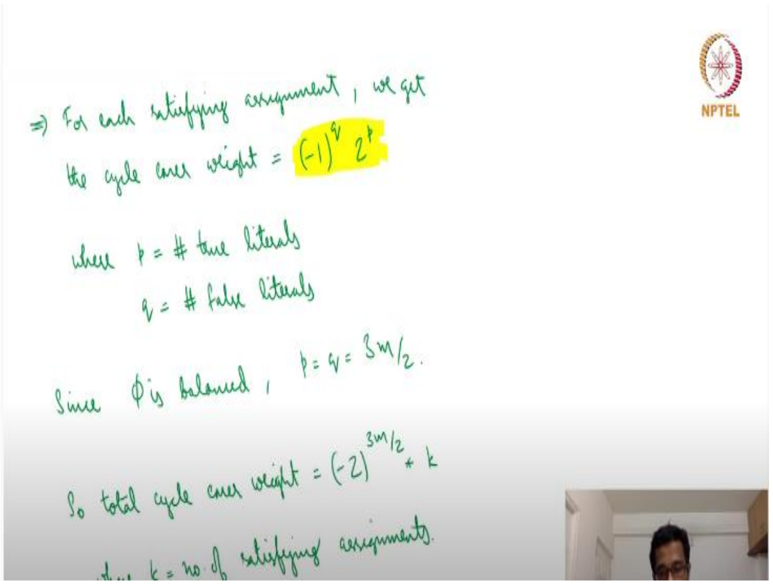
So now let us forget the gadgets that we have seen so far at this temporarily. We have two edges uv and u prime v prime. The requirement is that if uv is picked u prime v prime should be picked and if uv is not picked u prime and v prime should not be picked. But with these kinds of independent things it is not immediately possible to connect them. So what we will do is we will add an intermediate vertex or three intermediate vertices and add some connections between them and this will help us connect them.

So there are multiple cases and we will see that all of them turn out to be the right way. So what we will show is that let us say uv corresponds to let us say some x i equal to or let me just say uv corresponds to x 2 being 0. And this one was some u prime v prime was let us say some l i is false. So this means that l i was just x 2. So x 2 equal to 0 has to be the same as l i is false. So when uv and u prime v prime are both selected so this is what this gadget will accomplish.

Again the sum of this is a recap but there are a bit more details. When uv and u prime v prime are both selected this means there is a variable that is set in a certain way that causes a literal to be false. There are some false literals. Let us say there is a false literal and this results in a cycle cover of weight minus 1. So we will see how this happens with minus 1. When neither uv nor u prime v prime are selected which means x 2 was not 0 and this l i prime or sorry l i complement was not selected which means the literal l i was set correctly satisfying that clause.

In that case there will be 6 cycle covers and some of them will have weight 2. The total weight will be 2 and there will be other cycle covers also and all of them will add up to weight 0. They will cancel out. So this now hopefully tells you why we get this quantity. Earlier I said this. If the p true literals and q false literals the cycle cover should have weight minus 1 power q to 2 power p. So each true literal gives 2 and each false literal gives minus 1. So the total cycle cover weight is the product of all of this. So minus 1 power q times 2 power p.

(Refer Slide Time: 51:29)



$\Rightarrow$ For each satisfying assignment, we get the cycle cover weight = $(-1)^q 2^p$

where p = # true literals

q = # false literals

Since ϕ is balanced, $p = q = 3m/2$.

So total cycle cover weight = $(-2)^{3m/2} * k$

where k = no. of satisfying assignments.

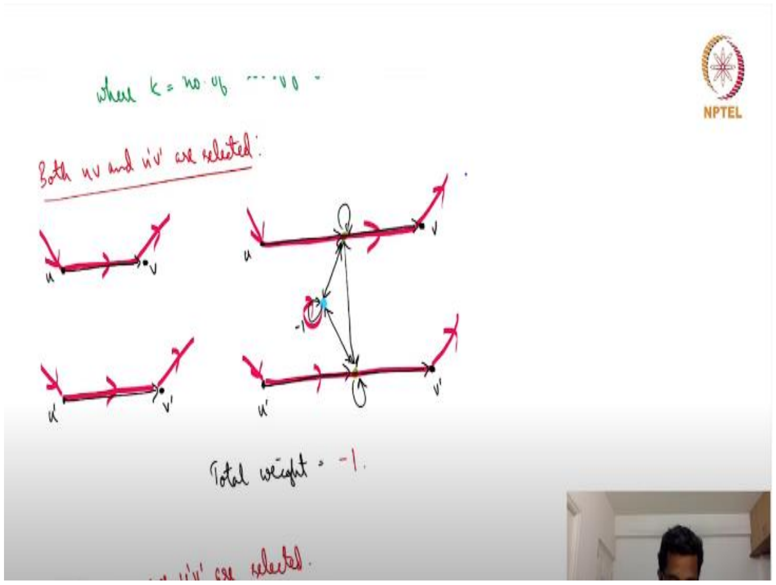
So this establishes this. And we have already seen that when the assignment is not satisfying there is no way to cover the central vertex of the clause gadget. That will not even contribute anything because there will be no cycle cover corresponding to that. So the cycle cover only contributes for the only satisfying assignments contribute to cycle cover. The weight will be minus 1 power q times 2 power p where p is the number of true literals and q is the number of false literals.

When I say true literals it is the literal that is set in the correct way. So if the literal is x 1 then x 1 setting to 2 is a true literal. If the literal is this x 1 complement x 1 setting to false is a true literal. And we have already seen since phi is balanced the number of whichever assignment you do the number of true and false literals will be the same which will be 3 m by 2. So the total each satisfying assignment contributes to a weight of sorry since p equal to q is equal to 3 m by 2 it will be 2 minus 2 whole power 3 m by 2.

This is the contribution of one satisfying assignment. If there are k satisfying assignments the total contribution is this. Minus 2 whole power 3 m by 2 multiplied by k. So now for the second part of the proof what we need to establish are these three properties. When uv and u prime v prime are both selected the cycle cover will be of weight minus 1 and neither of them are selected there will be 6 cycle covers that give a total weight 2.

Other cycle covers somehow cancel out. So this is the gadget. So notice there is only one vertex here with weight minus 1 but one edge with weight minus 1. The self loop at the blue vertex all the other edges have weight 1. And when I am drawing these two headed edges so these are actually between this green and blue, green and green, green and blue here as well. So there are two headed edges which means there are two edges. One going up to down, down to up. So let us see the cases.

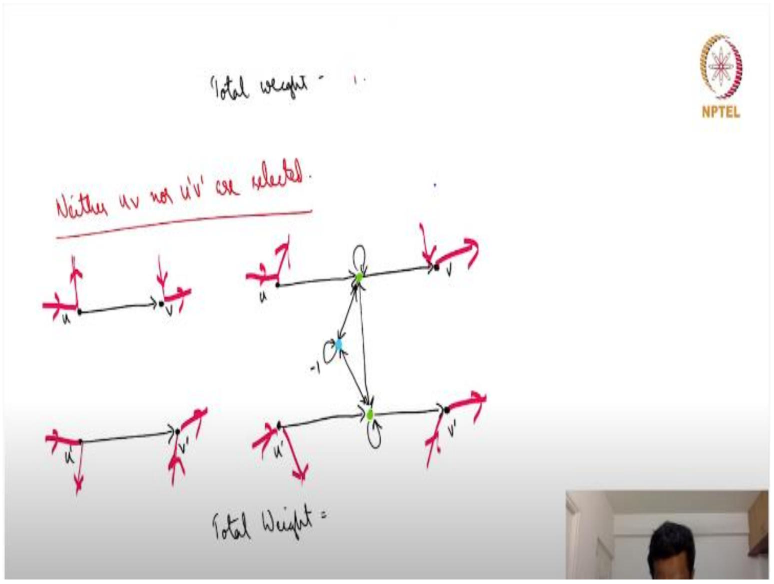
(Refer Slide Time: 53:48)



Consider the case when both uv and u prime v prime are selected, which means in the original graph. So the original graph is in the left hand side. You come to u somehow like this then you use this edge and then you go out of v. Similarly you come to u prime like this then you go to v prime and then you exit like this. So now let us see the total weight here. This corresponds to using this. So now all the edges in the gadget are already covered except for the blue vertex which we introduced and we need to cover that.

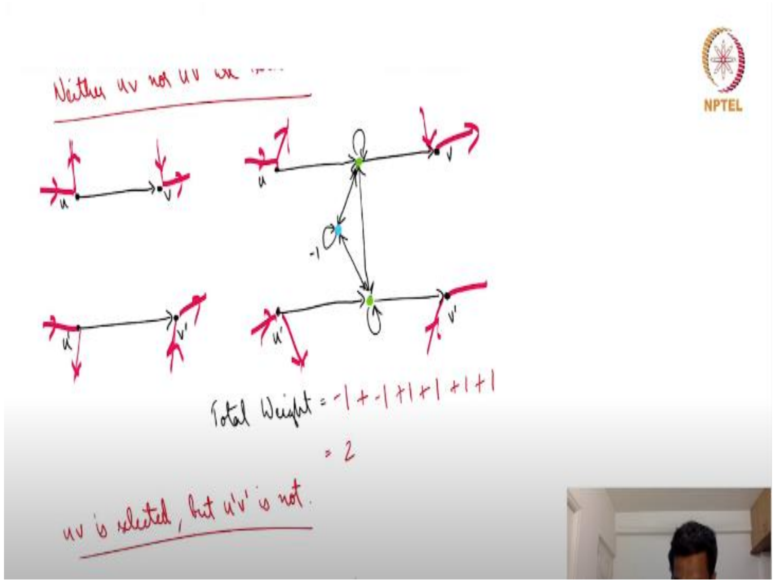
So what is the way to cover that? There is only one way. We just include this self loop. So the total weight of the cycle cover is whatever was the total weight multiplied by a minus 1. So this corresponds to minus 1 and there is no other way to include anything else. So this tells us how this claim was true. Now let us see this.

(Refer Slide Time: 55:36)



Let us see neither u sorry neither uv nor u prime v prime are selected which means in the original graph you come to u and you go out somehow, you come to v you go out somehow. We do not use these edges come to u prime, come to v prime. Something like this, same thing over here. So now these three vertices that we introduced the green and blue vertices somehow need to be covered. What are the ways in which we can cover this?

(Refer Slide Time: 56:15)

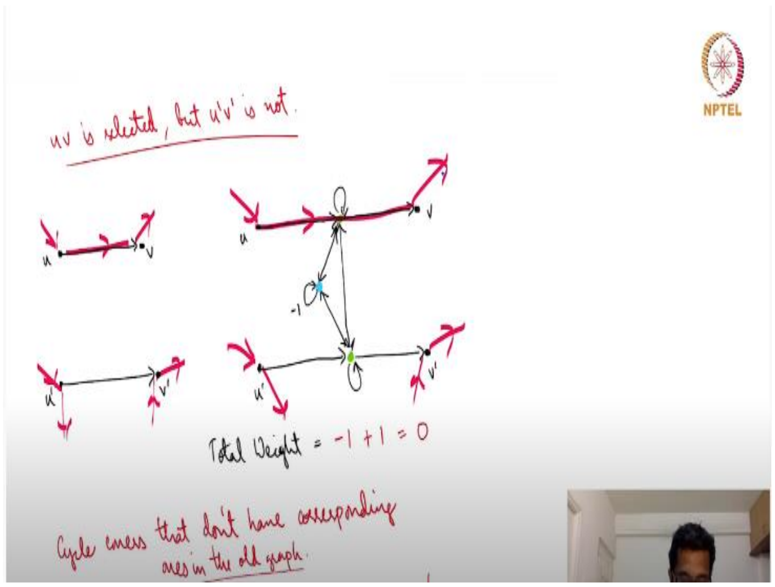

Neither uv nor u'v' are
u
v
u
v
-1
u'
v'
u'
v'
Total Weight = -1 + -1 +1 +1 +1 +1
= 2
uv is selected, but u'v' is not.
NPTEL

So if you look at it closely there are multiple ways. One way is simply to use self loops at each of the three vertices. So that will correspond to a weight of minus 1 sorry and that will correspond to weight of minus 1. Or you could use so I am not going to all the examples you could do at these two edges and then you could use a self loop here that will again give a weight of minus 1. Or you could use a self loop here and then these two edges like this.

So that will give plus 1 and if you look at it closely and there will be another cycle cover where there is a self loop here and these two are picked and there will be two more cycle covers, one where you do something like this and one where you do something like this. So a total of six cycle cover the first two being negative and the last four being positive. So you can verify this. So I am not explaining the details that slowly.

But I hope it is very clear. So the total weight of all this is plus 2. So now this part is also clear. Then neither uv nor u prime v prime are selected; there are six cycle covers of total weight 2. Now what about other cycle covers?

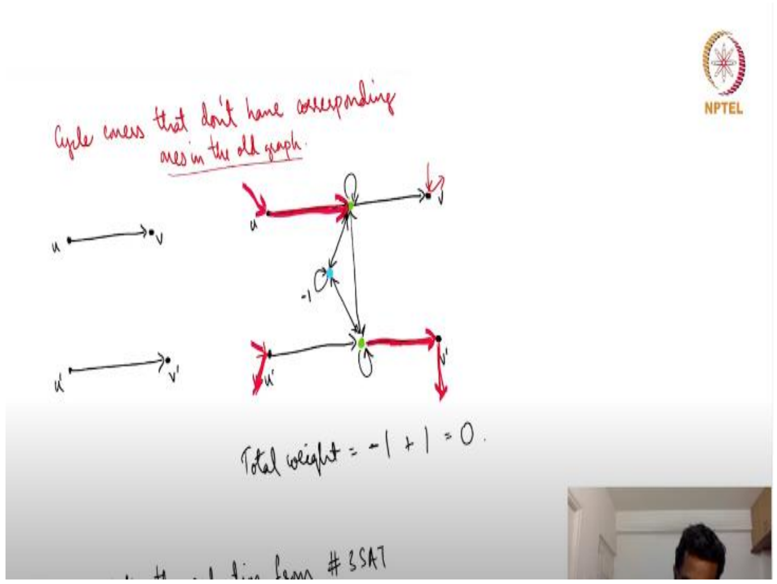
(Refer Slide Time: 58:17)


uv is selected, but u'v' is not.
u
v
-1
u'
v'
Total Weight = -1 + 1 = 0
Cycle covers that don't have corresponding ones in the old graph.

Let us say uv is selected but u prime v prime is not. Something like this which means in the gadget this happens. So now the top green vertex is already covered but the bottom green vertex and the central blue vertex are not covered. So two ways to cover it, one is you could use the two self loops at the green and blue vertices which will give a total weight of minus 1. Because it is minus 1 multiplied by plus 1 or you could use the edges across you could have a cycle of length 2 which will give plus 1.

So the total weight is 0 the plus 1 and minus 1 cycle covers cancel out and the same thing happens when u prime v prime is selected but uv is not.

(Refer Slide Time: 59:15)


Cycle covers that don't have corresponding ones in the old graph.
u
v
-1
u'
v'
Total weight = -1 + 1 = 0.
# 3SAT

And then there are some other weird cycle covers. Something like this where you come to u and then you do this and then you somehow exit from v prime. So these do not have any corresponding things to the original graph because the original graph does not have an intermediate vertex between u and v. If there is a cycle cover like this in the new graph this fortunately does not really cause any problem. Because how can you connect the top path to the bottom path?

So the one way to connect is through you can do this kind of thing and where the blue vertex has a self loop and that corresponds to a cycle cover of weight minus 1. Because of a self loop or you could do something like this where the blue vertex is included and without so self loop does not come into play here. So this corresponds to a cycle cover of weight plus 1. So total weight is minus 1 where the green to green you draw an edge and the blue one is covered with a self loop so minus 1.

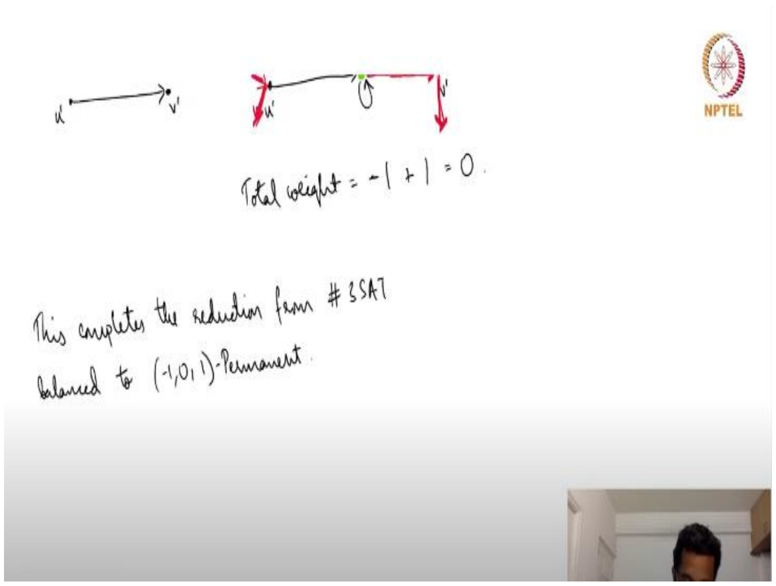
And two is plus 1 where you go from the green to the blue and then to the next green and then follow this. So the self loop does not come into place. So the weight is plus 1. So the total weight is 0. And similarly you can think of another cycle cover that is where you go to u prime then v then you go climb up top and then exit through v. But none of this does not really impact the; total of cycle covers because the total weight is 0.

So what is happening here is that only the consistent cases where both uv and u prime v prime are picked or neither uv nor u prime v prime are picked, they give the positive total weight. And whenever one of them is picked or and the other one is not and other nonsense cases such as this all the cycle covers cancel out. So leading to the total weight 0 which is what we said over here. So, all that remains is to note that all the positive, all the true literals which correspond to cycle covers of weight 2.

All the false literals correspond to cycle covers of weight minus 1 and the rest of the cycle covers do not really the nonsense cycle covers. And only the consistent satisfying assignments contribute to the cycle cover. So the total weight is minus 1 whole power q times 2 whole power p where p is the number of true and q is the number of false literals. But then we know that this number of true and false literals is always 3m by 2 regardless of whether the assignment is satisfying.

So the total is minus 2 whole power 3m by 2 and we get different cycle covers for each of the satisfying assignments.

(Refer Slide Time: 01:02:56)



Total weight = -1 + 1 = 0.

This completes the reduction from #3SAT balanced to (-1,0,1)-Permanent.

So this completes the reduction from sharp 3 SAT balanced to minus 1 0 1 permanent. So just to summarize we reduced we said what is sharp P complete. We explained what is permanent. We saw the corresponding permanent and the bipartite graph perfect matching of a 0 1 permanent. Then we saw the correspondence between permanent and the sum of the weights of the total cycle covers.

And then we saw the first two parts of the reduction where we reduced sharp 3 SAT to sharp 3 SAT balanced where each variable appears in the positive and negative form equal number of times. And second we saw that when such a formula is given where each 3 SAT instance where each variable appears equal number of times in positive as well as negative, positive and the negative form.

You can reduce it to a permanent. Is minus 1 0 1 permanent? This was done by constructing a graph with weights 0 minus 1 and plus 1. So as I said 0 corresponds to all the edges that are not there plus 1 and minus 1 we saw. So, such that if you compute the; permanent you can count the number of satisfying assignments of the 3 SAT formula. And we explain the gadgets of this reduction, variable gadget, clause gadget and the consistency gadget.

And what remains is to show that you can reduce the minus 1 0 1 permanent to a 0 1 permanent. And that I said follows very two fairly straight forward steps but since this video has probably gotten fairly long I will show that in the next part. Since this reduction is kind of a bit long I did not want to split it into two parts. But the remaining parts I will show in a separate video. Thank you.