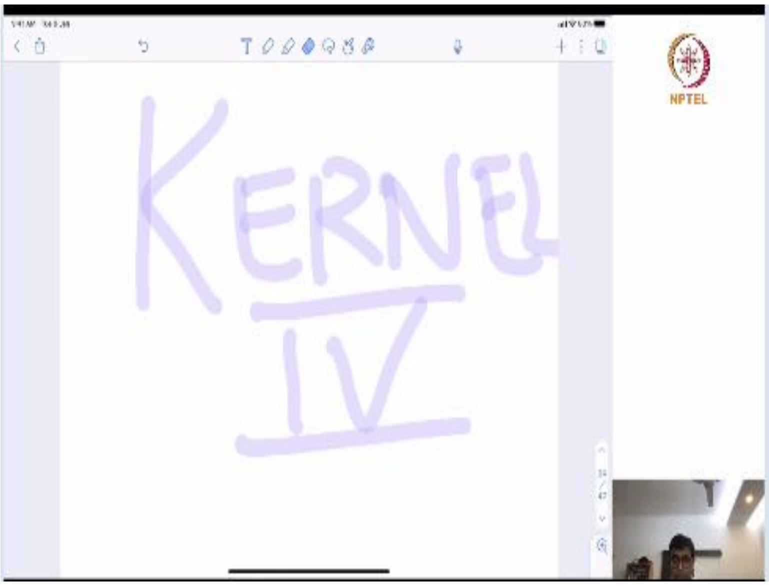
Parameterized Algorithms
Neeldhara Misra and Saket Saurabh
The Institute of Mathematical Sciences
Indian Institute of Technology – Ghandhinagar

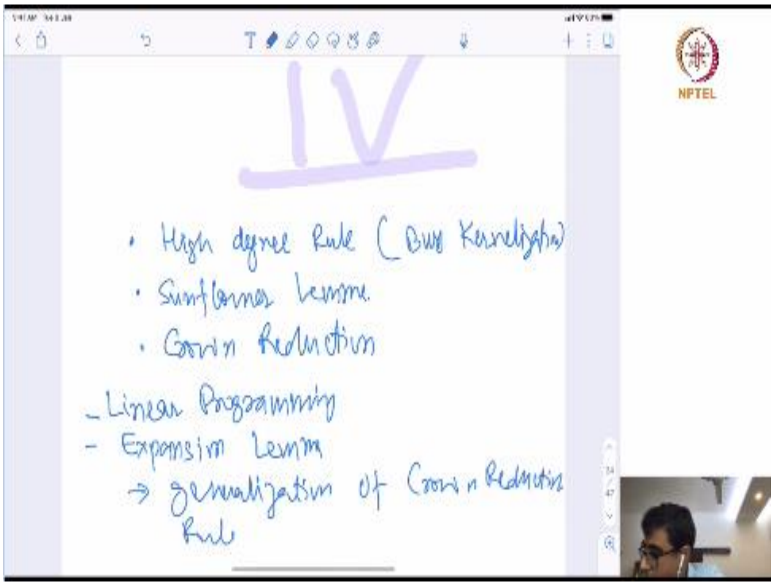
Lecture - 06
Kernelization: Nemhauser-Trotter and Expansion Lemma

(Refer Slide Time: 00:15)



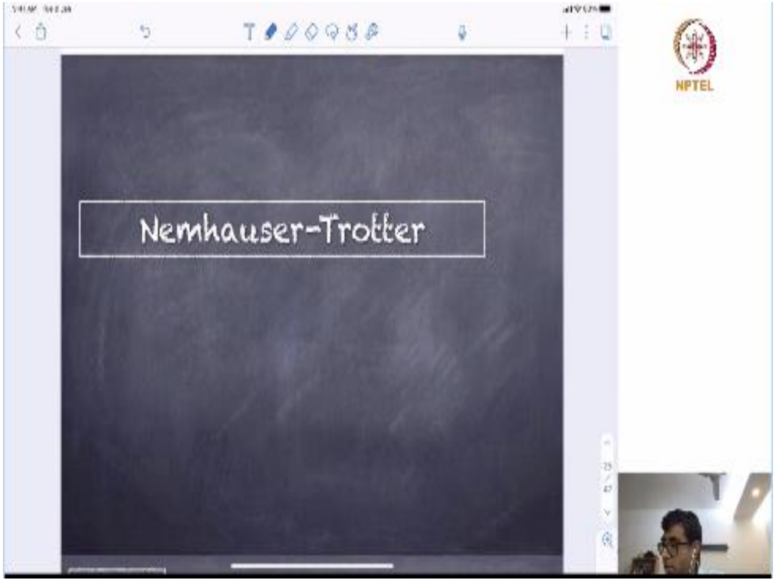
So, welcome to the last lecture of kernel 4 and this is the last lecture of kernel 4. So, until now, we have seen several kernels and we saw several tools to design polynomial kernels among them, but first was like this high degree rule based kernel and this is also known as bus kernelization.

(Refer Slide Time: 00:30)



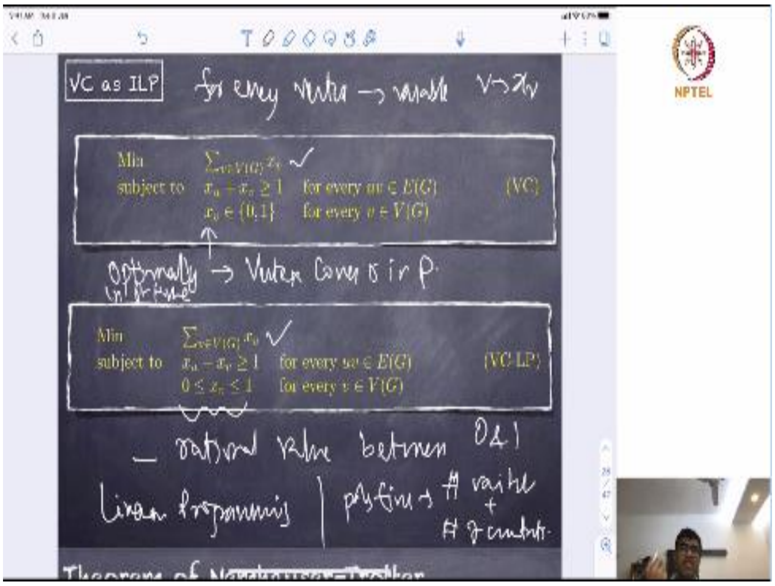
Then we saw some kernel based on sunflower lemma. Then we saw kernel based on crown reduction and now, in the remaining lecture, we will see 2 more tools. One is based on linear programming and the final one will be let us say, expansion lemma. And we should view this as a generalisation of crown reduction rule. So, let us get started.

(Refer Slide Time: 01:43)



So, the first kernel which we will learn or the first tool which we will learn is based on what is called Nemhauser-Trotter and it is from early 80s but 70s, I think.

(Refer Slide Time: 01:57)



So, how do you write integer linear programming for vertex cover? So, then that is so, what you do is that for every vertex, V associate a variable. So, if your vertex V, you associate a variable x v. What is our goal? We want to and I know that if you select a vertex, let us say that x v is assigned 1 and if you do not take a vertex in the vertex cover, let us say it is assigned 0. So, assignment of 0,1 says that what is the constraint.

You want to minimise the sum of X v, it means the number of vertices that you want to select in and what is the constraint that from every edge, you must have picked at least 1 vertex which is given by x u plus x v is at least 1 for every edge like so, at least 1 of the vertices has been assigned 1 and each vertex takes a variable between a value, between, the value not between, takes value either 0 or 1. So, this is correctly or faithfully determined the vertex cover.

But now, we will, but you know that we cannot solve this. We cannot solve this optimally because; why cannot we solve it optimally? because this will imply that or when I say optimally in p time, because this will imply that vertex cover is in p. So, in general, this also imply that integer linear programming or NP hard, because they can capture vertex cover as 1 of the example or as one of.

So, integer linear programming is extremely powerful framework or of constrained linear programming to capture NP hard problems. And there are lots of ILP solvers, what I was talking about the beginning of my course or the kernelization lectures for CPLEX. So, people solve these kinds of things like for millions and millions of variables every day, but still the problem is NP hard.

And in those kind of CPLEX programming, we use a lot of kind of pre-processing what we have learned, but have a different kind of based on how variables looks like, what is their interaction in the constraints and so on and so forth. So, in general, that is a comment about ILP. But what we will do is that we will relax this considerations. What we are going to? We are going to relax this condition slightly and we are going to demand that okay.

x v is not going to be integer, it could be any rational value between 0 and 1. So, when we relax this, this is what is called linear programming. Because now, it is a linear programming. Because my constraint is linear. It is not a quadratic in my variables. And like, my objective is also linear and my constraints are also linear. So, this is like linear programming.

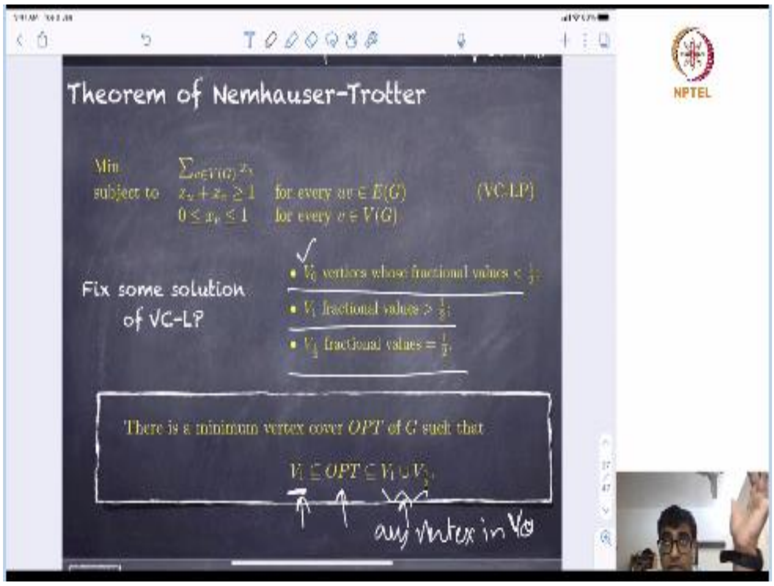
I am not going to go into this course, but it is well known that you can solve linear programming in polynomial time. So, then this is like poly linear number of variables plus number of constraints. I must tell you that like, we must make a remark that actually there are sufficiently many solvers that is just polynomial in number of variables. And the objective function, they do not need to know all the constraints at the same time.

So, the constraints can be exponentially many, exponential in the number of variables, but what they require is that these algorithms work as follows at any point of time, they solve, they say, hey, is this your value? Is this a feasible solution? That is all the required. They require that is this feasible meaning, does all this constraint satisfied? Then all my job is to take your assignment and then say, hey, no, no, no. It is not satisfied.

Look at this particular constraint, it is violated. So, this is what is called violation based, violation constraints. So, if we have an Oracle or if we are poly time algorithm, the given an assignment, I can test whether some constraint, whether all constraints are satisfied or not. If not, output a constraint that is violated, even then linear programming is in poly time. So, when I say is poly in variables and constraints, it is not like this is true of course.

But in fact, even much more is true and that is what makes linear programming extremely useful object because number of constraints could be exponential, could be much, much more, but we can still solve it in polynomial time, if we have Oracle or polynomial time algorithm that given an assignment can check whether all the constraints are satisfied or not. If not, out produce a constraint that is violated.

(Refer Slide Time: 07:10)

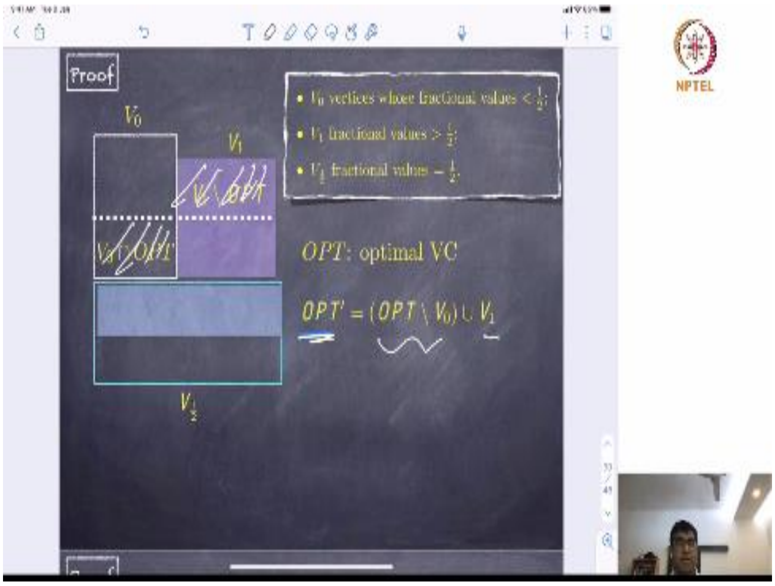


So, with all this in mind, so, what is the theorem of Nemhauser-Trotter tells us So, it tells us that let us solve this LP and let us classify the vertices based on the values they take. So, what is my V 0? V 0 is towards vertices whose fractional values are strictly less than half. What is my V 1? Those whose fractional value is strictly greater than half. And what is V half? All those vertices which is fractional values are half.

So, what Nemhausers taught to prove that there is a minimum vertex cover OPT of G with the following property. That OPT contains every vertex in V 1 and does not contain any vertex in V 1 union any vertex in V 0. So, basically it says that there is an optimum solution that contains every vertex which had been, which is in the set V 1. It does not contain any values, like any vertices, which is whose fractional value is strictly less than half.

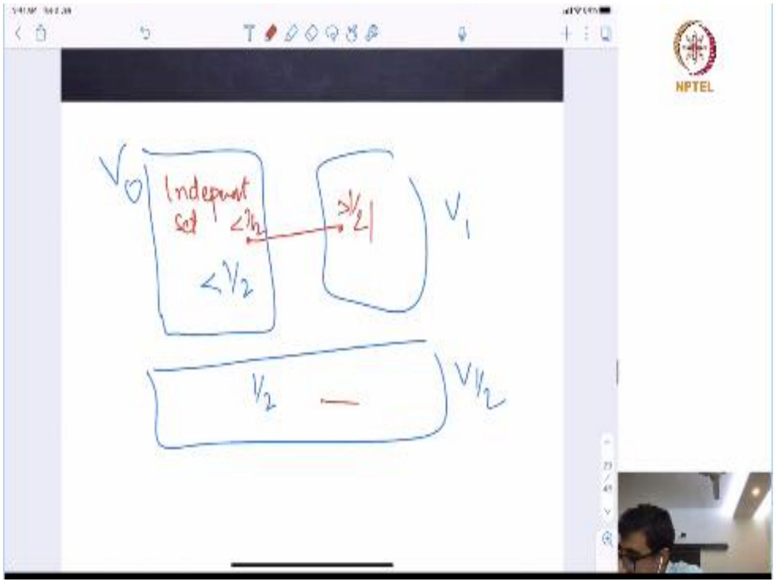
So, it is a very powerful algorithm. Because it just automatically provides a reduction rule. In the sense that delete every like, take every vertex in V 1 into a solution and delete V 1 and V 0 and you may ask haha, what you can you delete V 0. If you delete V 0, then you must pick V 1. We will see that why that will not happen.

(Refer Slide Time: 08:37)



So, now, here is my picture. So, this is my variables which has been assigned half. This is my variables which had been assigned V 0. And this is the variable which has been assigned strictly greater than half, strictly less than half and this is equal to half. So, before we go, let me draw another picture for you that will make that might make.

(Refer Slide Time: 09:10)

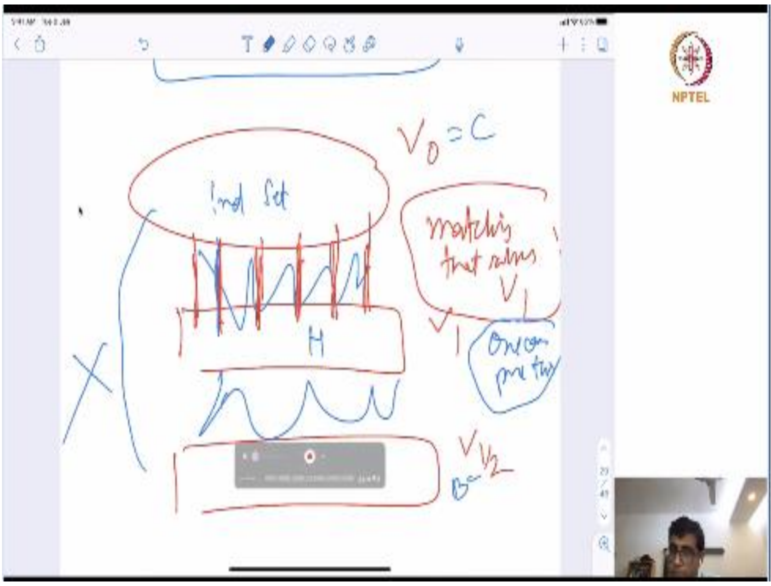


So, here is V 0. Here is V half and here is the V 1. Let us ask ourselves. Let us ask ourselves whether can there be an edge between here; can there be an edge between here. Notice that there cannot be an any edge between here. Why? Because, look, all these guys are getting value half and this is strictly less than half. But if I sum them, there this edge had value less than 1, but this is an LP solution.

So, that for every edge, if I sum the value this variable takes that is more than 1. So, knows if this is the, if my solution are the feasible solution, it implies that there is no such edges. Let us ask ourselves. What about can there be an edges here? No edges. Why? Because then this is also assigned less than half, this is also assigned less than half, then in total, this is a sign less than 1, not possible.

So, what is V 0? It is an actually independent set. And what is? So, but can there be an edges? Yes, I do not care, they do not. So, all the edges, which starts in here, we can, must go to V 1, because this is the only way I can have less than half and this is greater than half. So, this is possibility that they can sum up to 1.

(Refer Slide Time: 10:55)



So, in fact, if you notice, in the terms of crown, this looks like much more like a crown. In some sense, V 0. We have V 1 and V half and this is an independent set. And there are edges like this. And there are ages like this, but there is no edge like this. So, this is kind of crown. But, why it is not yet crown? Because we do not know if there is a matching, is the matching saturating to show that there is a; it is a crown. I need to show that there is a matching that saturates V 1.

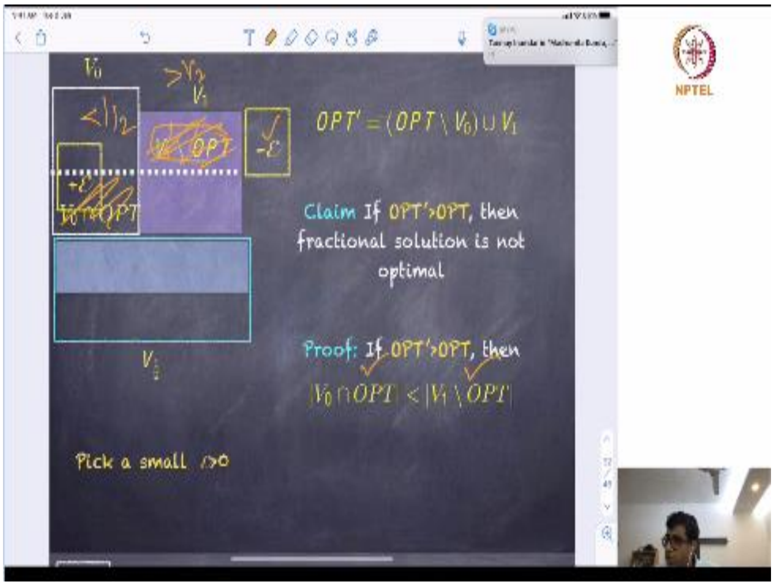
And in fact, one can prove this, one can prove this, but I will not. I will not. I will not get into that for a minute. So, I will not prove this way. But it is possible to prove that this partition forms a crown with V 0 being a, V 1 being head. So, V 0 will become in this is like crown, becomes a crown, this becomes a head and this becomes a body. But let us see what we go about. So, this is just to have a mental picture in your head.

So, let us try to see. But, what are we trying to prove? We are trying to prove, there exists an optimal solution that contains every vertex of V 1 and does not contain any vertex V 0. In fact, the optimum solution is completely contained inside V 1, V 1 union. Now, you notice why if you contain V 1, then definitely the moment you delete V 1, every vertex in V 0 becomes isolated vertex and hence you can delete. It exactly like a head. Exactly like exactly like a grounded option.

So, let us try to prove this. So, now let us look at OPT. So, OPT. So, the stated reason shows OPT. So, this is intersection of OPT here and do the vertices from V 1, which OPT does not pick and it picks some vertices from V 1. So, OPT. Now, let us try to do it like OPT is optimal vertex cover. I am now trying to make a new OPT. What is new OPT prime?

So, from OPT, the moment I make a OPT prime, I delete this portion that is OPT minus V 0. I deleted and I add everything from here so, V 1. So, what did OPT happens? So, OPT prime is like I delete all the vertices which were present in V 0 from this and I added every vertex in V 1. So, this is what I did. And this is an optimal vertex cover, there is no fractional optimum, this is just like real optimum vertex cover. Now, there are 2 cases.

(Refer Slide Time: 14:16)



First of all, let us notice that why OPT prime is a vertex cover? Because why OPT prime is vertex cover, first of all let us try to understand. Look at we had just now notice every edge which is incident on V goes here. So, only agent which V 0 can cover or incident to V 1 also and hence if I picked up all the vertices in V 1, we have taken care of all the edges which V 0 intersection OPT cover.

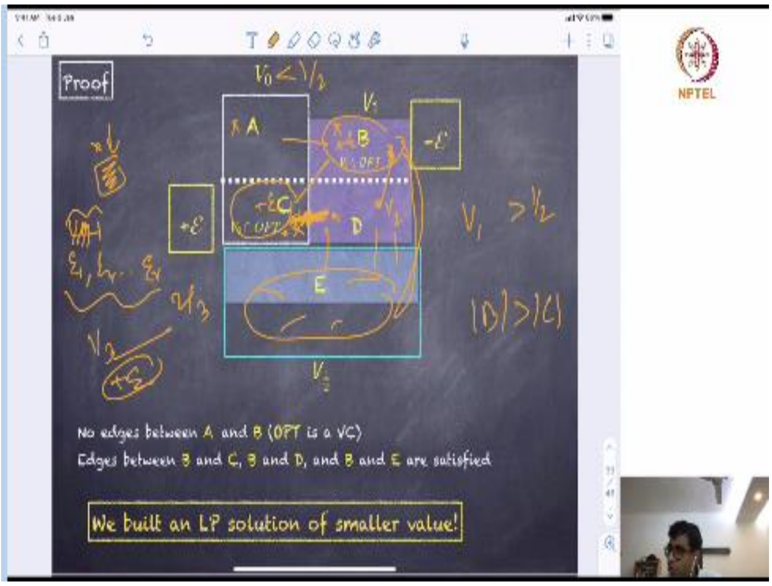
So, it is very clear that OPT prime is also a vertex cover, it may not be the sign OPT but since, OPT is a vertex cover what we know. OPT prime is at least OPT. So, if OPT prime would have been OPT, we are done because then we have shown to you that there exists an optimum vertex covered which picks up everything in V 1 and V half and does not picks up anything in V 0. Then we are very happy.

But, what could happen is that OPT prime is greater than or equal to OPT, then I will say Oh, if OPT prime is greater than or equal to OPT, then we can find, then we can actually reduce the optimization, reduce the linear programming OPT solution, which is a contradiction because what we had is an LP OPT, optimum value of the linear programming. Then let us try to understand what happen is. Why did?

So, in OPT, we deleted some vertices from, we deleted OPT, some vertex from V 0 minus OPT and added this. The only region of OPT prime can be larger than OPT because this set which we have added is larger than V 0 intersection OPT. So, V 1 minus OPT must be larger than V 0 intersection. Now, what we are going to do is that we are going to pick some epsilon greater than 0, pick some epsilon greater than 0, we will see how to find those epsilon. And what will I do?

From every vertex in V 1, I am going to subtract minus epsilon. And from every V 0 intersection OPT, I am going to add this plus epsilon. Note that these are like all these vertices are strictly less than half, these vertices are strictly less than half; greater than half. So, all we need to find an epsilon such that I mean, my constraints are still satisfied. Let us see what happens.

(Refer Slide Time: 16:45)



So, notice that there was no edge between A and B. Why? Because OPT is the vertex cover. So, it did not. OPT did not pick anything in B and anything in A. It means there is no edge. So, every edge is satisfied. Look at an edge here, but then look at this edge, we have subtracted minus epsilon, I have added plus epsilon. So, the current solution which I made is also a proper OPT.

Now, look at any edge here. This is where we have to choose epsilon small enough that. So, we will choose an epsilon is small enough such that the vertices in V 1 is still going to be strictly greater than half and what is in V 0 is strictly going to be less than half like every vertex So, like you choose a small enough epsilon that this happens. So, for example, here, you know that if you this is some value and this value you are going to subtract, what is your leeway.

So, you say okay, look at you have not given me some value, but look, you can only subtract me this much. So, that I still get you equal to half. So, this gives you like, if I fix an edge here, it will give you a constraint. My range of decreasing epsilon is only this much like. So, this will tell you. So, similarly, so I choose this number epsilon, epsilon 2, this is my range. I can subtract this much and still be greater than equal to 1.

Now, let us, no problem with edges here. They are taken care of. Similarly, no problem with an aged here. And you notice that there are and what about ages here. Edges here could also give us a constraint that look you can subtract me only this much that even after I add this to right. So, like what I mean to say that look at an edge here, I only, so, it will tell me that.

You can subtract me only as much as like so for example, this would have been say, half minus, say for example, this is one-third and this is like, then the edge look at this like two-third. I think there are some inequalities here but like, you can just see that like, it will tell you by how much can be decrease. Sorry, I made a mistake. It is here, we are doing plus and here we are subtracting.

So, here, look at this, suppose, these have been one-third and two-third right. Now, look at one-third and two-third, like it does not matter. For here, you added some plus epsilon, any plus epsilon is good for us. So, because we have not decreased anything, so, this is perfectly fine. So, and then again you have an edge like this here, it will tell you though this is strictly greater than half, you can only delete some epsilon, so, that; so, you will get another.

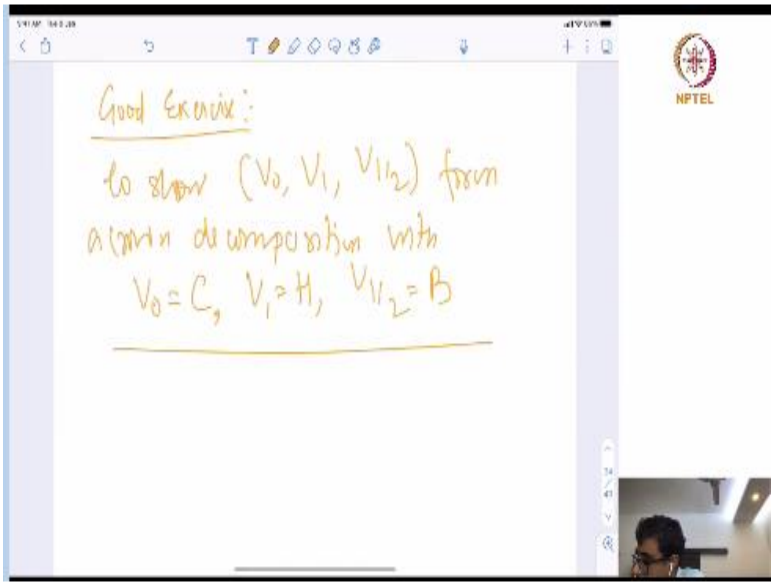
So, if you choose minimum of these numbers is still a positive because everything is positive and you can find a small epsilon such that you can still satisfy all the inequalities of LP. But, what is the point? Look the values of here did not change; values here increased by plus epsilon; valued here decreased by minus epsilon, but notice we have decreased more vertices compared to what we have increased because the size of B was larger than size of C.

So, we have built a LP solution of smaller value. So, the only thing which I like basically what you do to find this epsilon, you will choose. If I take this edge, how much epsilon leeway can I increase or decrease? You write down those numbers and then you choose the minimum and you perturb your LP solution slightly with this. So, that you still remain a feasible solution.

But because you have decreased more compared to what you have increased, your new LP solution is strictly smaller than what you started with. And that is a contradiction to the fact that you had an LP OPT solution So, what we have shown now is that because of all this, we have shown now that there exists a minimum vertex cover OPT of G says that it contains, completely contains V 1 and it is completely contained V 1 union V half.

So, what is our reduction then? The reduction will is very simple once we prove this lemma. I also make the point here it is a good exercise.

(Refer Slide Time: 22:32)

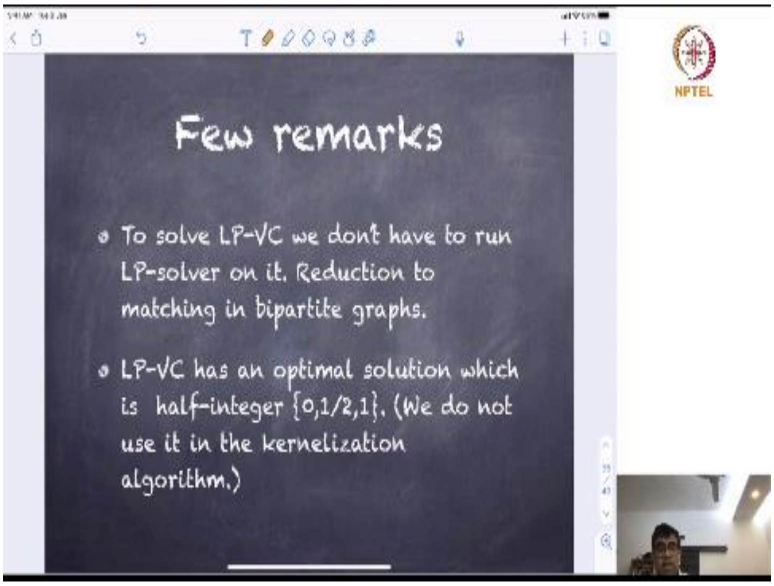


Let me add good exercise that to show V 0, V 1 V half form a crown decomposition with V 0 equal to C; V 1 equal to H and V half equal to B and this is not very hard. So, all you have to show at this point of time for to do that is that you have to show that for every set x in H, there is at least like look at this bipartite graph between H and I, this then every subset in A satisfies, what you call, it satisfies false condition.

Otherwise, if it does not satisfy the false condition, then you can get a smaller LP value again as before, because now then you will find a constraint here that it has, this is smaller neighbour. This is a larger guys. This is like more number of vertices in edge but smaller neighbourhood. Now, you give reduced their value by minus 1, like you will increase this value by plus or minus epsilon and then you get a smaller LP value which will contradict.

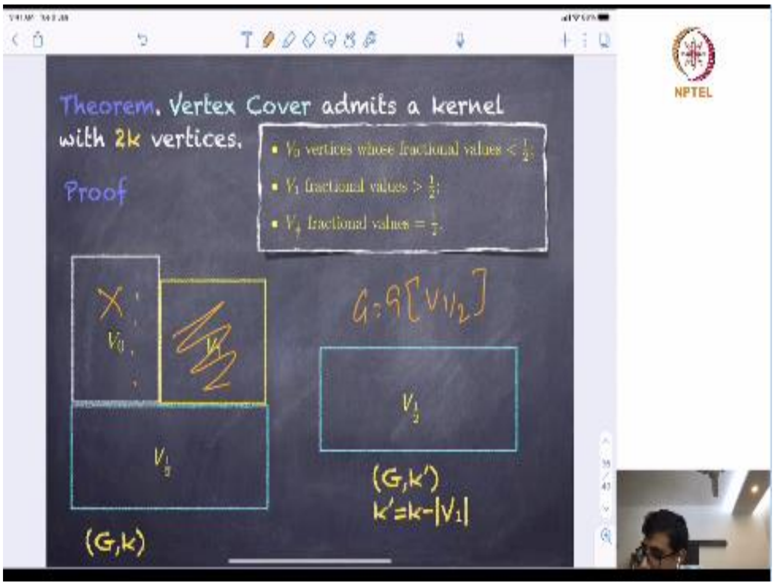
So, this is how you can show. So, this is exactly the same proof which I have told before, so, please try.

(Refer Slide Time: 24:04)



So, here to solve LP vertex cover, we do not have to run an LP solver that is a very nice reduction to matching in bipartite graph. In fact, what we can show that linear programming for vertex cover has an optimum solution which is half integer like 0, half, 1. We do not using this kernelisation algorithm, but it is known and you will use this in your branching algorithm in the next week.

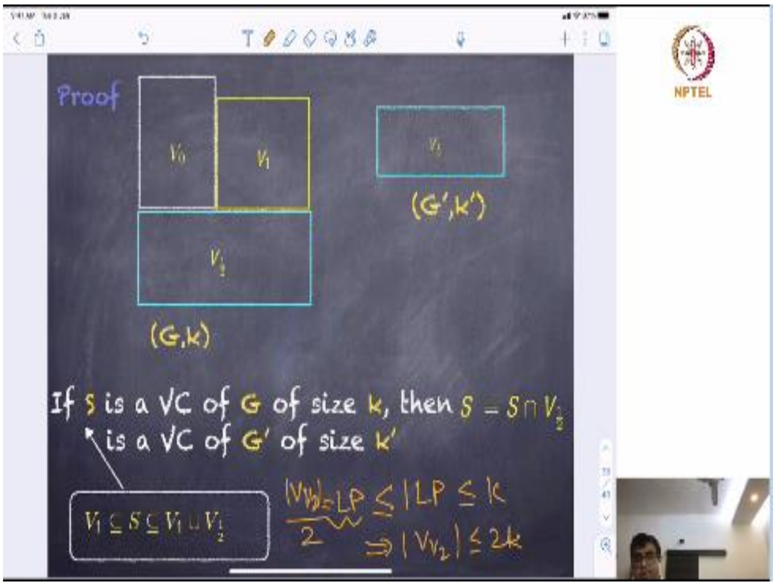
(Refer Slide Time: 24:26)



So, I will make that remark formally at the after this. So, I want to first with vertex cover admits a kernel with 2k vertices. How? So, we compute this LP solution. And so, we have a reduction like you pick all V 1, delete V 0 by our condition, the optimum solution that contains V 1. So, the moment you delete V 1, every vertex in V 0 will become isolated. So, what is left is V half.

Now, and what your new G into k prime; k prime is k minus cognitive V prime where G, I should write G of graph induced V half. Now, what do you know? So, if the number of vertices look.

(Refer Slide Time: 25:12)



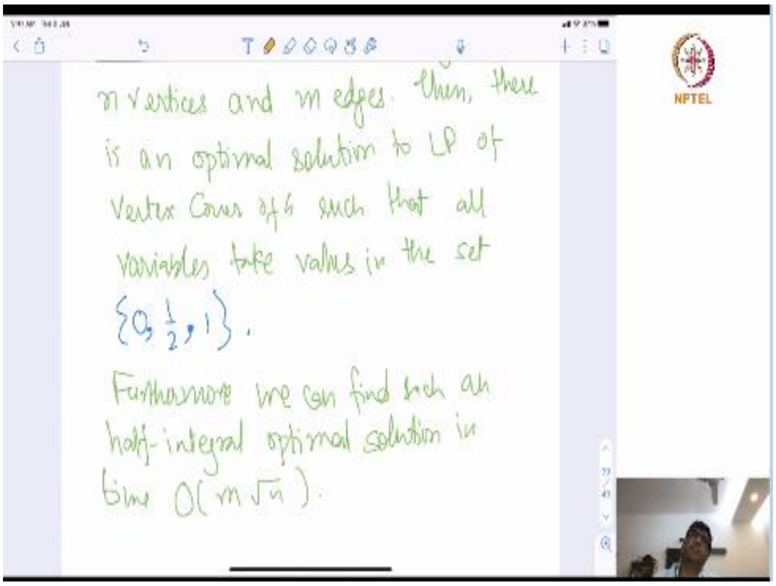
So, now, here is the proof. If s prime is the vertex cover of G prime of size k prime, then definitely s prime union V is the vertex cover G of size K clearly. And if s is the vertex cover of size k, then you know that s equal to s intersection V OPT is a vertex cover of G prime of size k prime and we know, because you contain everything here. So, if you delete this edge of vertex cover of size at most k for this graph in use and v half is the edge of vertex cover of size at most k minus side of Lemma.

So, that is very simple because this is an induced sub graph. But, you may ask,, but, sure, but why do we get a kernel with 2k vertices that is a very good question. And the reason is, notice it is an LP solution. So, LP solution is relaxation. What does it says? Look at integral linear programming must be greater like the integers is because you have relaxed this; it must be greater than equal to LP.

But, what is the value of LP? The value of LP is right. So, linear programming value is less than equal to ILP which is less than equal to k. And what is the value of LP? Its every vertex is at most exactly equal to half. So, V half divided by 2, which implies that V half better be at most 2k. So, if it is more than 2k vertices, you can say hey, this is no instance.

Otherwise, you know that the total number of vertices in V half is bounded by 2k. So, we have got a 2k vertex kernel for vertex cover and in some sense, I should not say that but like, it seems that we should not be able to do better but yeah.

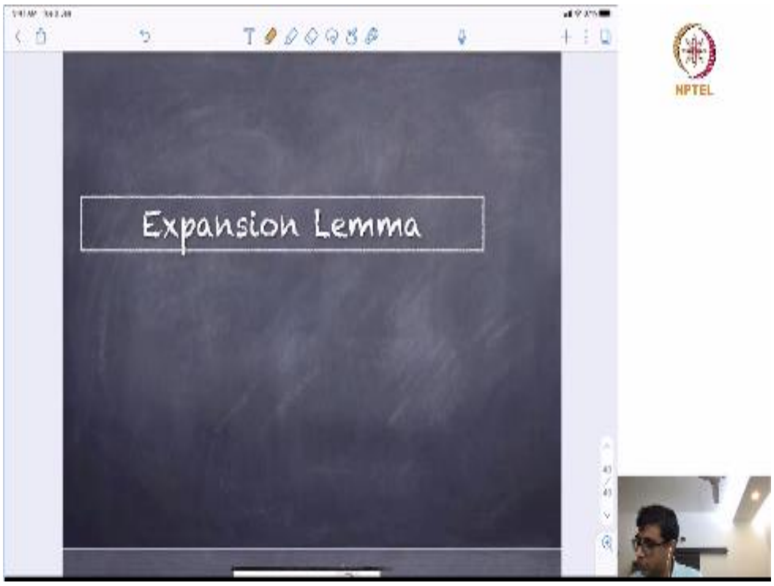
(Refer Slide Time: 27:22)



So, the remark is that how that half integral LP solution. The G be a graph on n vertices and m edges. Then the region optimal solution to linear programming of vertex cover G such that all variables takes value in the set 0, half or 1. Furthermore, we can find such a half integral solution in time m root n. So, we like, the way I created the sets V 0; all values strictly less than half; all values exactly equal to half and strictly more than half.

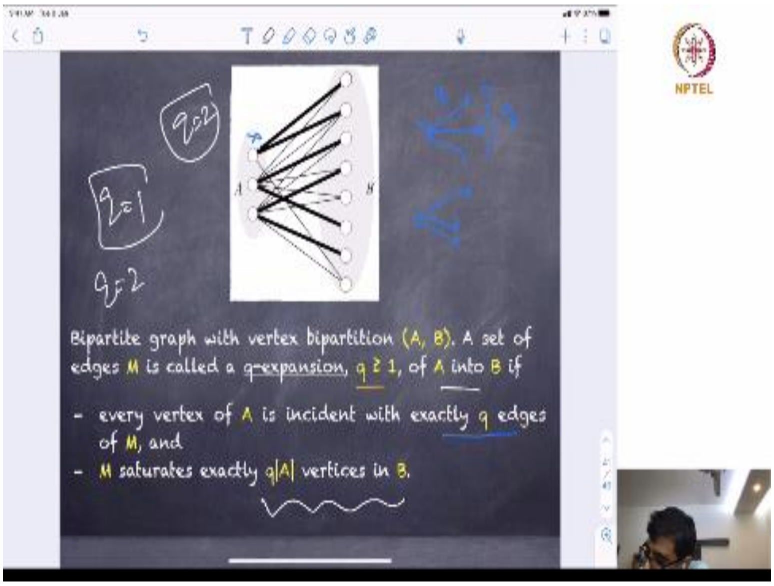
Actually, every you can make that V 0 set every variable set 0, every variables in top 1 and you can still show that this is good enough for your region. And noticed that if you show such a crown reduction like if you show that that forms a crown, then you should be able to do that properly.

(Refer Slide Time: 28:14)



So, the last topic that I would like to cover is what is called expansion lemma. What is an expansion lemma?

(Refer Slide Time: 28:22)



Expansion lemma is a generalisation of the false theorem or; and basically the crown reduction generalisation. So, suppose you have a bipartite graph, vertex bipartition A comma B, a set of edges of M is called q expansion. q greater than or equal to 1 of A into B if, what happens? Every vertex of a here for example, every vertex of A here is adjacent to q guys of edges M and M saturates exactly q times A vertices in B.

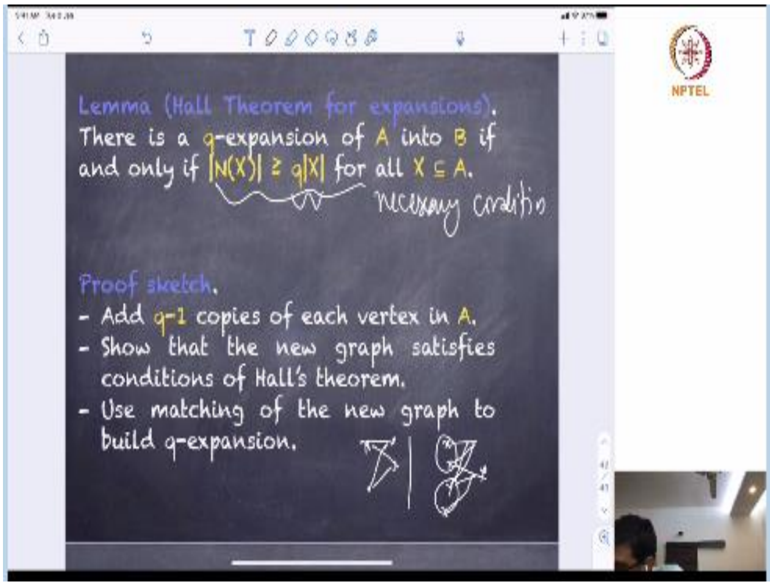
So, basically they are like a stars. So, every vertex of A is a vertex of some stars of size q and the property is that these stars do not intersect, don not like here, pair by vertex disjoint stars.

So, and if the star had size q, then you call it q expansion. So, every vertex of A is incident with exactly q edges and M saturates exactly q times A vertices.

So, if you notice that q equal to 1, this is nothing but classical matching ; q equal to 2, it starts like think of this that I have a vertex A which has 2 edges pair to him which is private to him. So, this is like we have found some q times A edges such that if I look from the B's perspective, these edges are vertex joint from the edge. A's perspective, q edges of these are grouped into 1 vertex. So, this is why it is called expansion.

So, this is; if you look at the picture, this is an expansion with q equal to 2 because every vertex head to like black edges which are private to it and they form star.

(Refer Slide Time: 30:22)

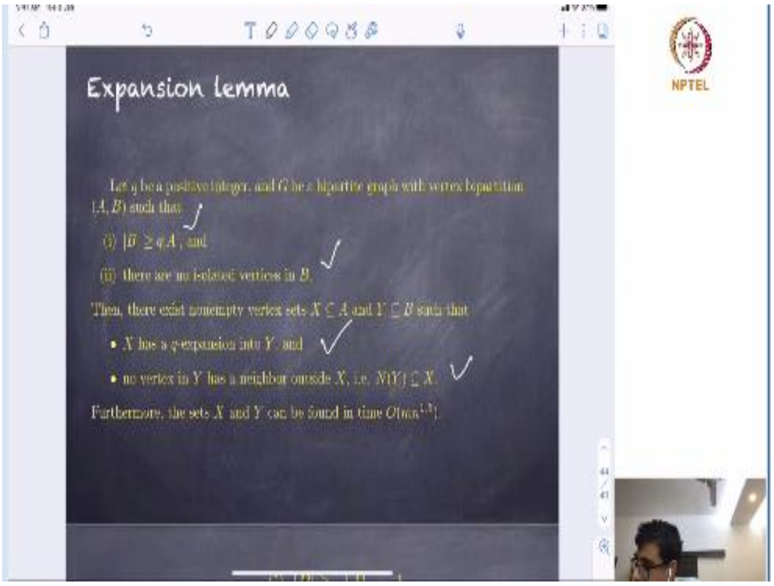


So, the degeneration of Hall theorem for expansion, the q expansion of A into B if and only if look at the neighbourhood of x, it should expand by q times x for all x subsets. So, if we expect that there is a q expansion of A into B, if for every subset of x there is q times that many neighbours and that is kind of; so, this is like kind of necessary condition.

And what this Hall theorem says that actually, it is a sufficient condition and the proof sketch is very simple, you basically apply Hall's theorem by making q minus 1 extra copies of each vertex in A and make them adjacent. So, basically what it means is that suppose here it is and another vertex, so, then suppose q equal to 2, so, I will take this graph, I make another copy of the same graph and make him addition to whichever vertex he was the adjacent to; use adjacent, I make another copy.

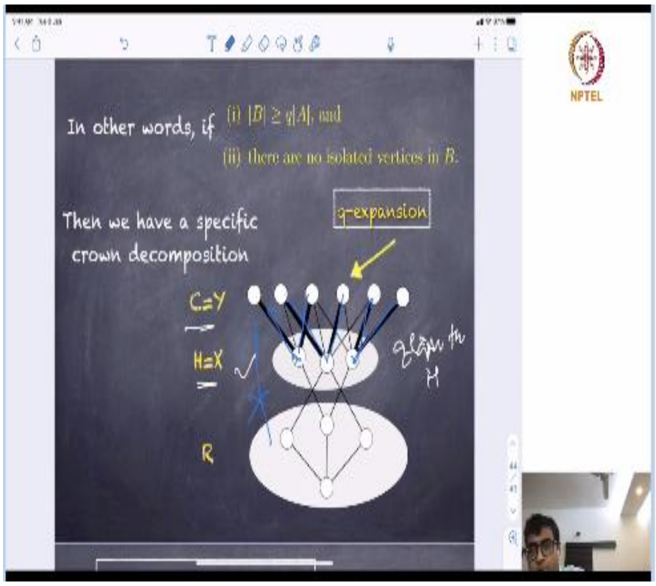
So, this is like a club or cloud whatever you want to call it. So, and show that the new graph satisfies condition of Hall's theorem and then you use the matching of new graphs build to expansion. Yeah, so, basically this is like, so, you take a matching here if I had a matching of this, if there is a matching which takes 1 vertex and some edge then I could have also find a another matching where every vertex in the group is matched and something of this. So, that should do the job. But even if you note, then that is perfectly fine.

(Refer Slide Time: 31:54)



So, what the expansion lemma says? So, basically, let q be a positive integer and G be a bipartite graph with vertex bipartition A comma B such that size of B is more than q times A. There are no isolated vertices in B. Then there is a non empty set x subset of A, y subset of B, so that x has a q expansion into y. And no vertex in y has a neighbour outside x. So, right and you can find such in poly time.

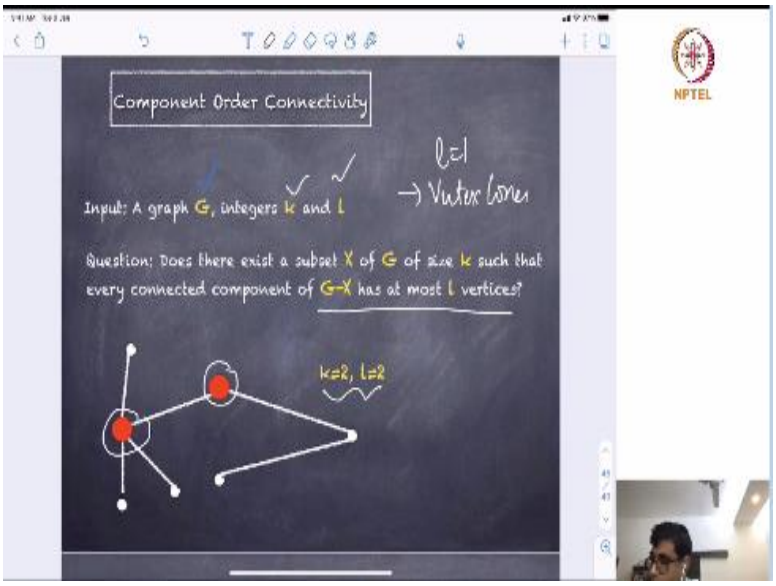
(Refer Slide Time: 32:24)



And so in a picture, how does it look like? So, in other words, in the bipartite graph, I can find the head which is like my x, my crown, which is y and what is your property? So, remember, for crown decomposition, we said that there is a matching that saturates the vertices of x; for q expansion, the q expansion that saturates head and the rest is rest. So, now, if I say oh, I have a 2 crown means for every vertex in the head, there are 2 private matching it in some set; 2 private neighbours in, 2 private neighbours in the independent set.

So, this is an independent set. And for every vertex, there are just 2 private guys and the rest is rest, there are no edges here. This is exactly like crown decomposition for q equal to 1 and it is the generalised it for q larger than 1 and you will see how we can use it.

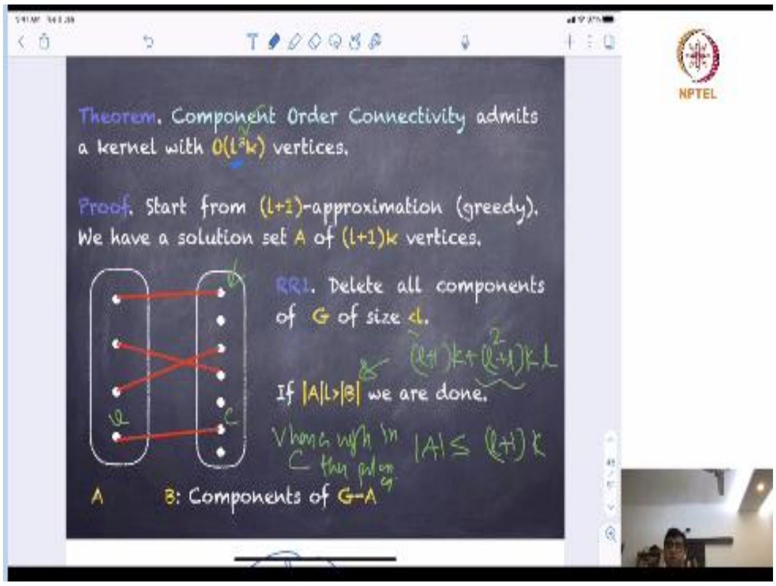
(Refer Slide Time: 32:24)



So, let us see an application of this theorem or lemma or the concept. So, this is the concept which will apply to what is called component order connectivity. So, your graph into a graph G; integers k and l. And what is my goal? Does it a subset x of G of size k? Such that every connected component of G minus x has at most l vertices. So, look at k equals 2 and l equal to 2. So, if I delete these 2 red vertices, what are your property?

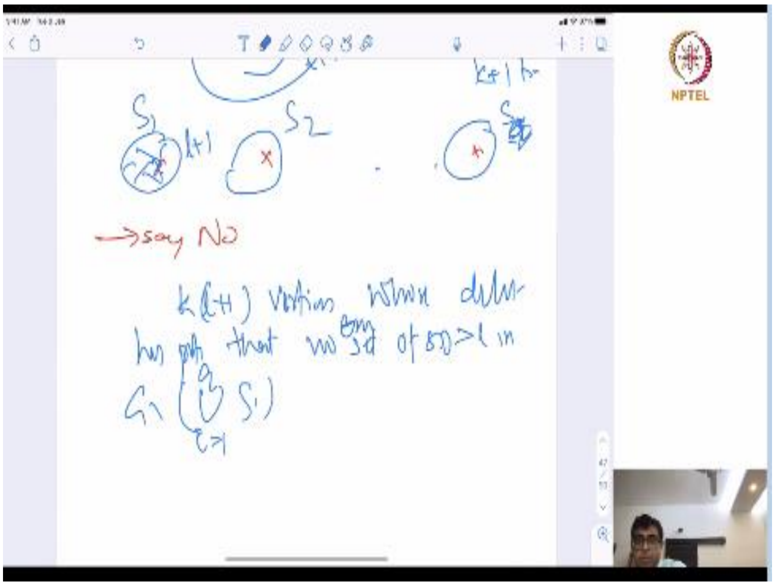
Every connected component contains at most 2 vertices. Notice that for l equal to 1, this is nothing but vertex cover. So, you want to delete minimum number of vertices such that every component has size at most 1. But now, what we want to do? We want to delete k vertices so, that every connector component has size at most l.

(Refer Slide Time: 34:21)



So, here is the simple kernel. So, first what we will do, so, let us say, starts from l plus 1 approximation. How do we start with l?

(Refer Slide Time: 34:39)



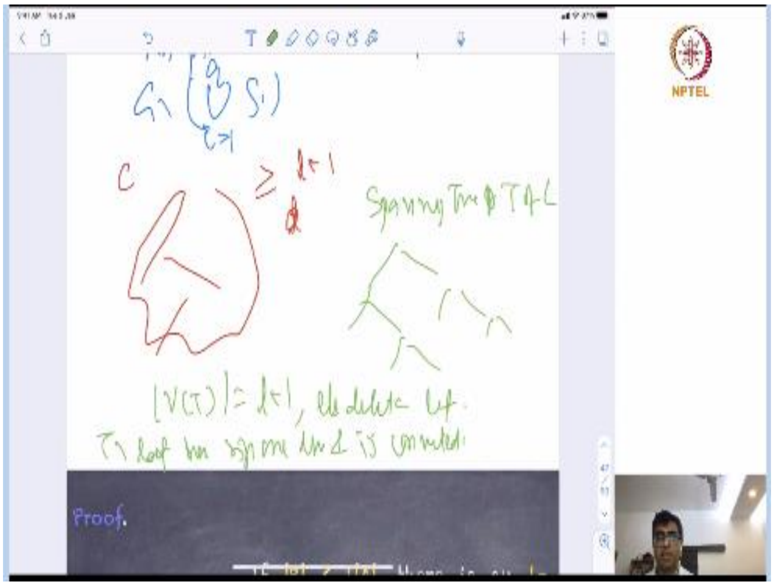
So, basically you say, you find a connected say a you find a connected set of size l. You find a connected set any connected set of size l. And for example, if there is a vertex with degree more than l plus 1, you pick this vertex and l. What do you know? Because this a connected set of size l plus 1, then what do you know about it? You know that every solution must pick this.

So, you basically pick some connected set of size l plus 1 greedily, you keep doing this until you succeed. If you succeed this k plus 1 times, then what do you know? That your solution must pick 1 vertex from each of these guys. So, you can say no. Else, what do you know that you have found k times l plus 1 vertices whose deletion has property that no set of no connected set of size greater than equal to l in G minus union of, say call this sets s 1, s 2. **(()) (36:15)**.

And we do not say yes. let us say q i equal to 1 to q, because otherwise we could have greatly added it. So, is this like a generalisation of maximum matching. So, what are the properties maximum matching that was like a connected set of size to say, greedily. Did it? If you succeeded k plus 1 times, then you so there is no vertex cover of size k. Now, you have generalised to connect a set of size l plus 1.

And you know that if none of these exists greedily, then the remaining graph has no connected component of size greater than equal to l plus 1 because any connected component of size greater than equal to l plus 1 does contains a connected set of size l plus 1. Why? So, that is very easy to show.

(Refer Slide Time: 37:03)



Suppose I have, this is my C and it has sides greater than or equal to l plus 1. So, what you can do is basically find a spanning tree T of C. If vertex set of T is exactly equal to l plus 1, nothing to do. Else, delete a leaf. What a property if you delete a leaf? The property is that T minus leaf has size 1 less and is connected. So, either 2 minus is leaf size l plus 1, then you are done, else, delete another leaf and you can keep doing this process till you get a connected component connected set of size l plus 1 in poly-type.

So, this is how you can greedily find connected set of size l plus 1. So, look at a character component of which size l plus 1, apply this process, get a greedy connected set of size l plus 1 and take this. So, what is; so basically this is why you have a set A of size l plus 1 k vertices. So, given this, now, I am going to; so, first of all, first basic rule, if there is a connected component in my graph whose size is strictly less than l or at most l.

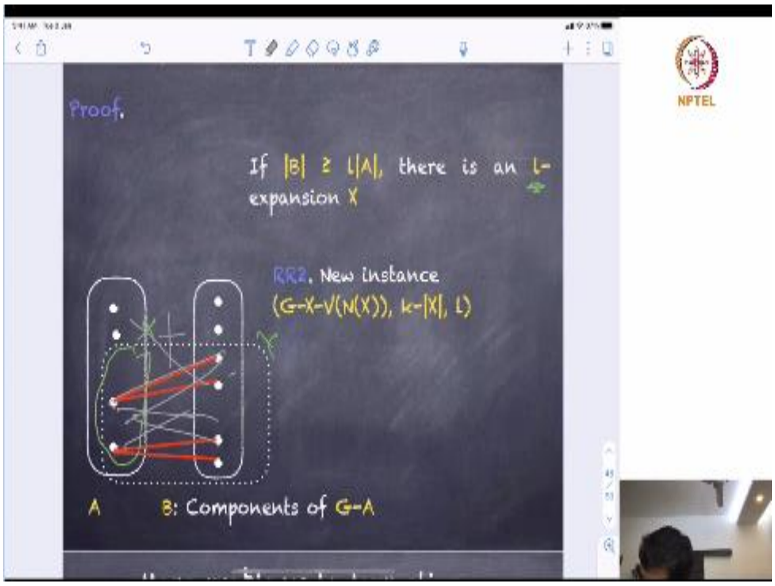
So, maybe there should be at most l, then you just delete that component because you do not need to delete any vertex in the connected component. Otherwise, we are going to make a bipartite graph which is define a follow. What is A? So, A is basically let us call this Z which is union of a S i, i is going from 1 to q. So, this is my set of vertices which appears in one of the obstruction which we have picked up greatly.

So, the size of A is upper bounded by, size of A is upper bounded by l plus 1 into k. So, this is like the vertices. And what is B? It is a component connected components of G minus A. And basically, there is one vertex for each connected component of G minus A and if this vertex has a neighbour adjacent to some vertex is connected component, you put an edge. So, what is the edge represents?

Suppose, this v and if v had a neighbour in C, then put an edge. Now, I am saying, so, first of all notice that there is no isolated. There is no isolated vertex in this graph. Because we have deleted all components G of size of d. So, now I check. Is number of components bounded by l times A or not? If the components of B are bounded by l times A, why are we done? Because notice that every component here in B has size upper bounded by l.

And so, what is the size of A? So, what is the size of this? So, notice. So, the size of A is bounded by l plus 1 into k. Size of B is like mod A times l; mod A is like l plus 1 times k another l. So, roughly l square plus l times k and each component so, that number of components is upper bounded by this and each component has sizes at most L. So, total will be like order l q k which is. So, let us look at the condition. When the number of components are larger than l times A strictly larger than l times.

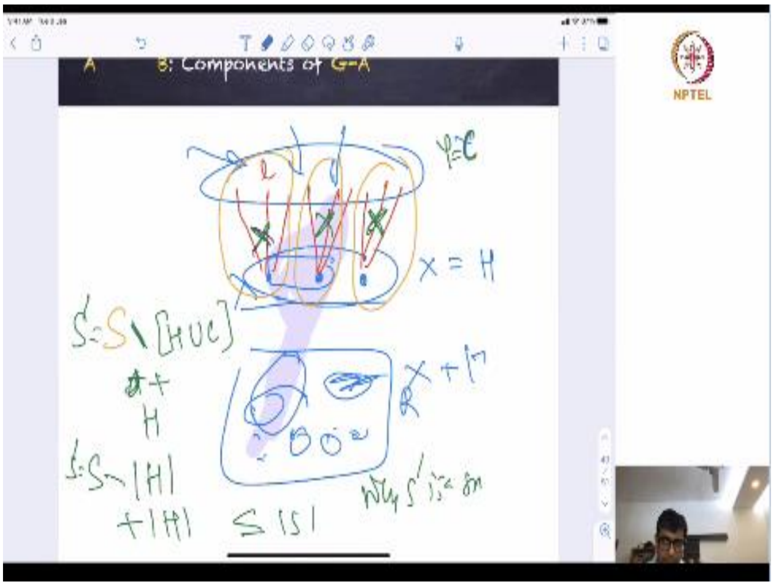
(Refer Slide Time: 40:47)



Then now, your expansion lemma with l being an expansion. So, what will you get? You will get here a set which is which I call say x and set y, what are the property of the set y? That this like they do not edges here, no such edges. So, every these guys are only connected here. So, just like a vertex cover, what my reduction will be now?

My reduction will be very simple. I will delete all these things. And reduce my parameter by the size of x and l remains **(()) (41:37)**. So, that is my new instance. Let us try to show finally that is the last thing we will do today is why this is correct.

(Refer Slide Time: 41:47)



So, the best way to think of this is like, here is your x. Here is your x, which like a crown, which like a head; here is your y and what the property of this? Everyone can be associated with l of 3. Let us say for example, it is a 3, l of 3. And remaining is the rest. So, forward direction is obvious. Like, if you can find me set of vertices, so that every connected component here had sizes at most l, then if you take this set plus H, then you delete it.

Then you know that every connected component here remained a connected component because you have deleted everything from head and the only new connected component is the connected component which is contained in y but those are also set of size at most l. So, that is no problem. So, it is only the reverse direction. But reverse direction, this is where I used local minimum, which can local OPT can be made.

So, what is the local OPT? You know that any optimal solution, this is connected set of size l plus 1, because there are stars of l plus 1. These are connected. So, any solution Z or let us say or any solution f, what your property of any solution S? S must be pick some vertex from here. Maybe use green, some vertex from here; some vertex from here; some vertex from here.

So, I know from this l plus 1 let us say, your solution must pick one vertex. So, what I am doing? So, from here, let us delete s minus x union y and let us add head, let us call it crown, which is y. So, let us, so, what will I do? I say let us delete head union crown and let us add all of head. Now, you know that from head union crown locally, you need as many as head of in vertices. So, you have deleted them.

So, you have definitely deleted say mod H vertices and you have added. So, new solution, let us call it S prime. Size of S prime is at most size of S. This is, I can show. What about correctness? Correctness is also follows. Why is this new solution? Why is prime S solution? Well, look at. So, why it is not a solution, because maybe there is like some component, which could be like this, okay, which could be like this.

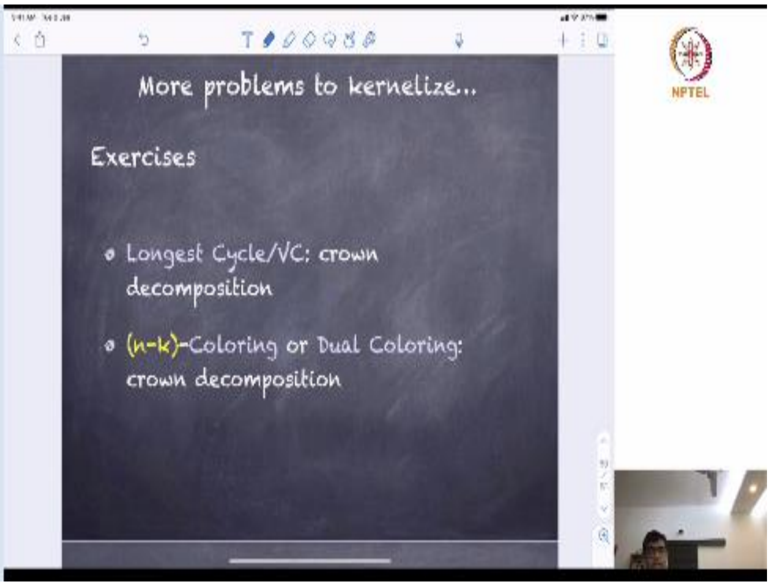
But now, if you delete more, if you delete verticals from here, then the component is only going to be broken into smaller pieces and the component which we will get from this inside, this that is anyway smaller. So, these are the only component that you should have been worrying about and those can only be broken much more for that. So, this also shows that we can have a local solution, which we can like, we have given a solution, we can make a solution, which is locally the way we wish and hence, we can use that for our reduction.

So, we have been able to use a local optimum to translate into a global optimum and use that to decide our calculation. So, what do we get? We get that either we are able to apply one of the reduction rule or we know, the number of vertices in my graph is bounded. So, this is how you can show that component order connectivity amidst l q times k vertices, there are very clever way of applying this component order connectivity to reduce this l to.

I think l log l or something of this nature, by looking at by grouping these components into what is the size is the size between 1 and 2; 2 to 4, so that you do not need to apply l expansion all the time. You say, for example, if every component has 5 l by 2 here, then too expensive we enough, because stay together will give me l plus 1 to guarantee that from that particular.

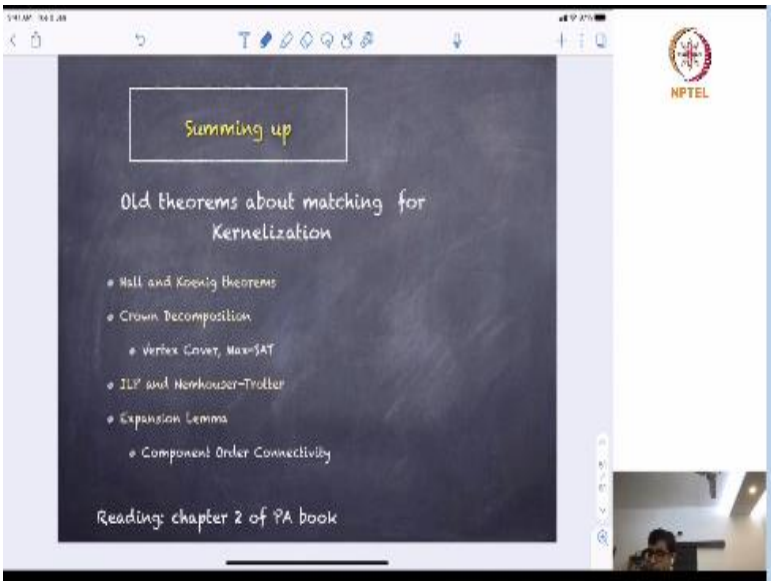
The only place to use the fact that we have applied l plus 1 expansion lemma to say that look, the side of this guy is at least l plus 1. Hence, you need to pick at least 1 vertex that is the only time when we have used that fact. Anyway.

(Refer Slide Time: 46:44)



So, there are more problems to analyse in this. So, you can do something called longest cycle parameterised by vertex cover using crown decomposition or n minus k colouring or dual colouring using crown decomposition. You can find some of these things in the book.

(Refer Slide Time: 46:56)



So, you read the chapter 2 of the parameterize algorithm book. So, we have done Halls and Koenig theorem, crown decomposition say, with vertex cover max set ILP, Nemhauser-Trotter and expansion lemma and we saw an example of component order connectivity. I think that is should end the last lecture of the first week. Thank you.