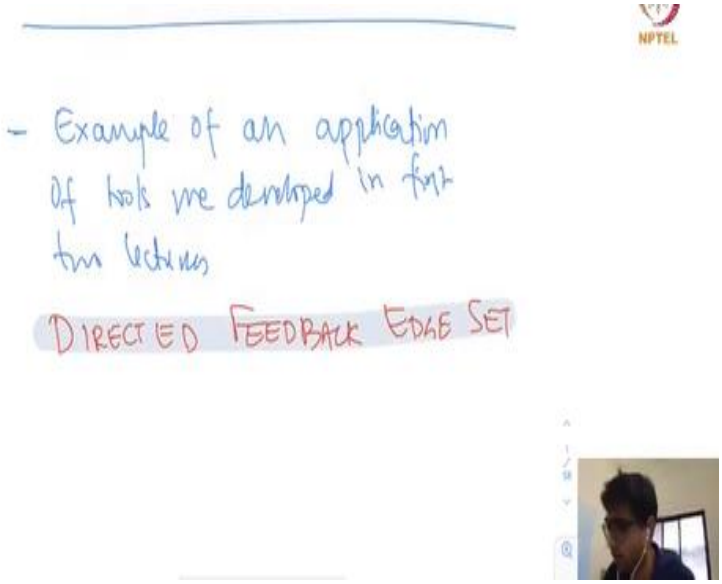
Parameterized Algorithms
Prof. Neeldhara Misra
Prof. Saket Saurabh
The Institute of Mathematical Science
Indian Institute of Technology, Gandhinagar

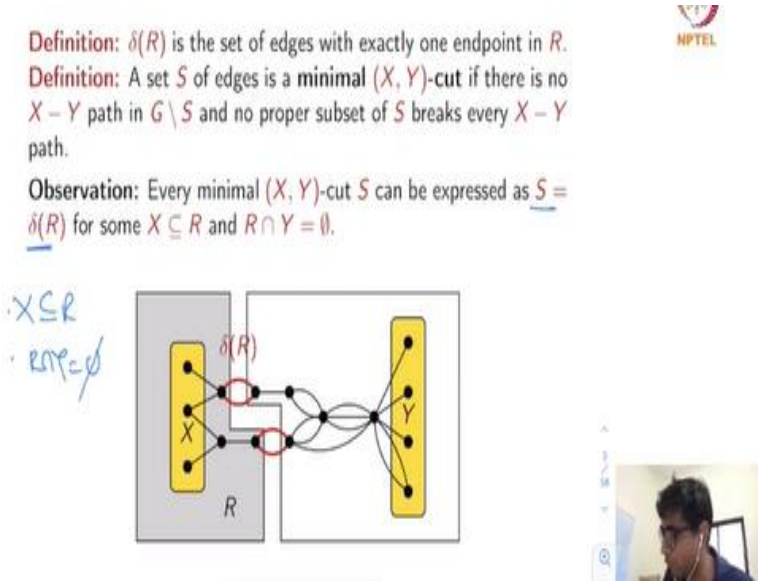
Lecture - 38
FPT Algorithm for Directed Feedback Edge Set

(Refer Slide Time: 00:15)



Welcome to the last lecture on important cuts. So, in the first two lectures or first three lectures you saw about important cuts, and in this we will use this to give another algorithm for one of the most important problems in parallel complexity which was whose parametrize complexity was an open problem for a long time and that was directed feedback Edge Set. So, that will be a goal to design for this.

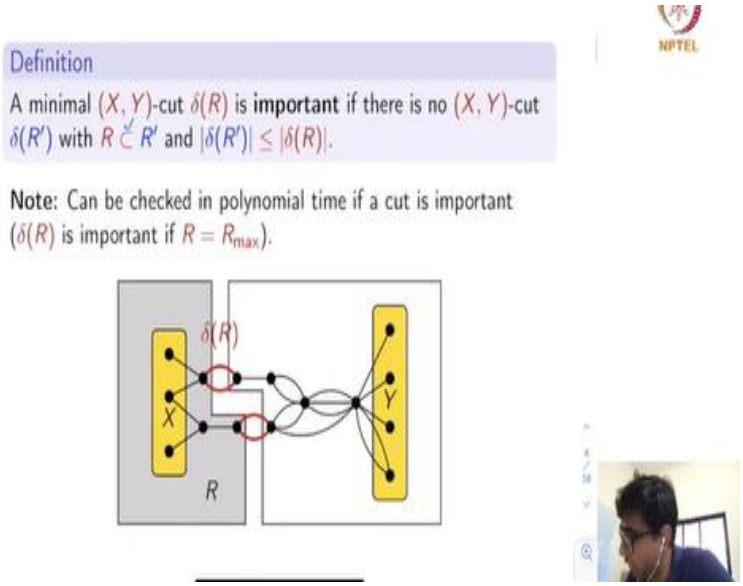
(Refer Slide Time: 01:03)



But before that let us just revise the some of the topics that we covered in the previous lectures, was that what we did first was if we define this notion of delta R, which will basically set of ages with exactly one endpoint in R, then we talked about minimal X, Y cut which was nothing but it is a set of edges, such that there is no path from vertex in X to a vertex in Y and no proper subset of S does this job.

Meaning if you only delete a proper subset of (()) (01:34), then there will be a vertex X and vertex in Y such that will be path. And an important observation was that for a minimal edge cut I can find a set R such that S is nothing but delta R. And what extra property is that X is contained inside R and R intersection y is empty. And so, for example, look at this for ages these are like areas emanating out of out of R.

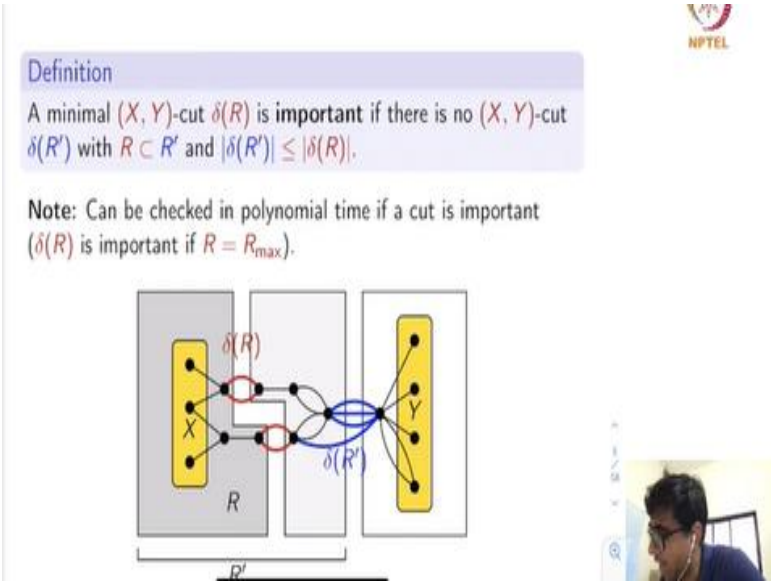
(Refer Slide Time: 02:06)



Then we define this notion of important cut. And what was an important cut? A cut was important if there was no other cut which dominated it and what do we mean by this a cut is called a minimal X, Y cut is important if I cannot find another X, Y cut, with a property that it covers more vertices that are like and number of edges that are leaving R prime is less than or equal to the number of edges that leaving R.

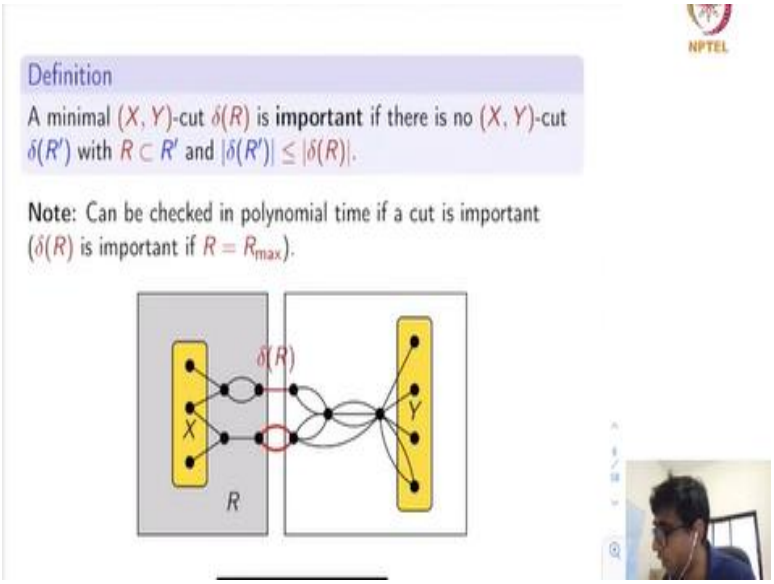
For example, look at this cut or is it an important cut and we also saw it showed that we can actually test whether a particular cut is important or not in polynomial. So, the question was is this cut important cut.

(Refer Slide Time: 02:50)



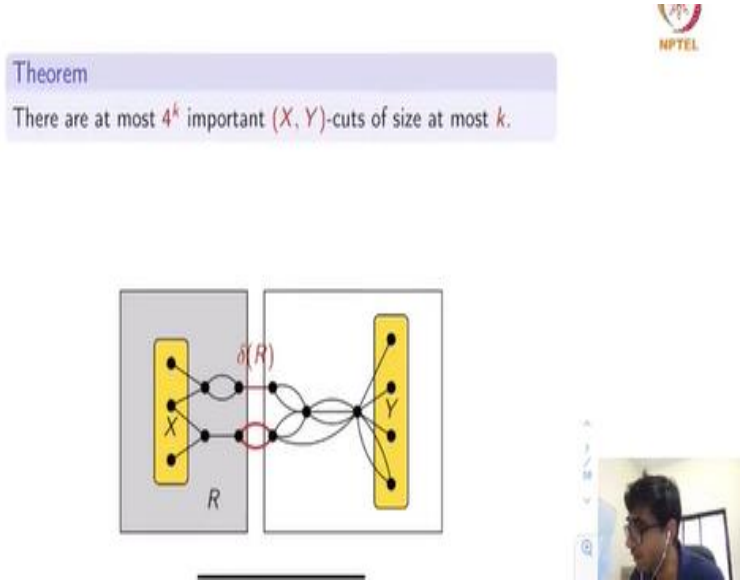
And answer was not because you can find an R prime which has as many as leaving as R. So, for example, there were four edges we have a four blue edges, but it significantly it contains R and also contains some other set of vertices. So, this particular art was not an important cut.

(Refer Slide Time: 03:10)



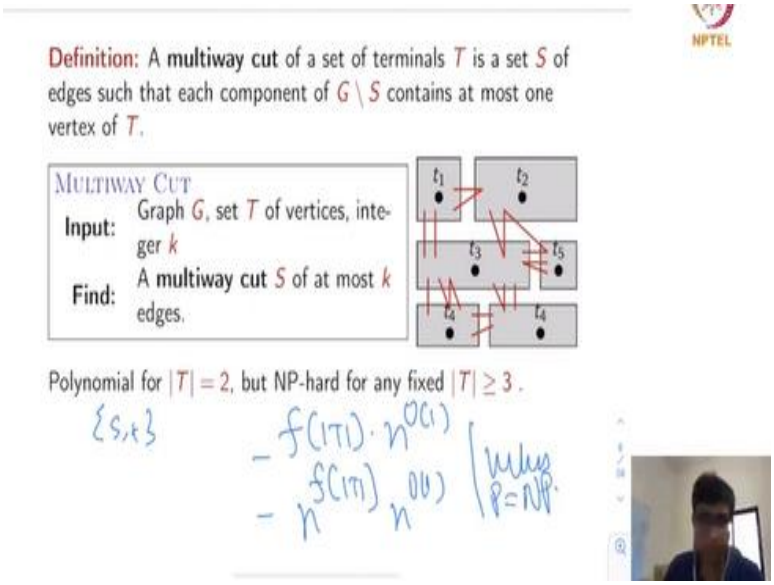
But you look at this R and look at these three red coloured edges, you can show that you cannot find any R prime which contains are properly and has at most three edges leaving. So, any other vertex you will put inside it will have more number of edges going out which implies that this particular R is an important cut.

(Refer Slide Time: 03:34)



And then we had this important theorem who showed that there are at most 4 to the power k important X, Y cuts at size at most k. And this is very useful because we could use this important cut to design an algorithm for a problem called a multiway cut.

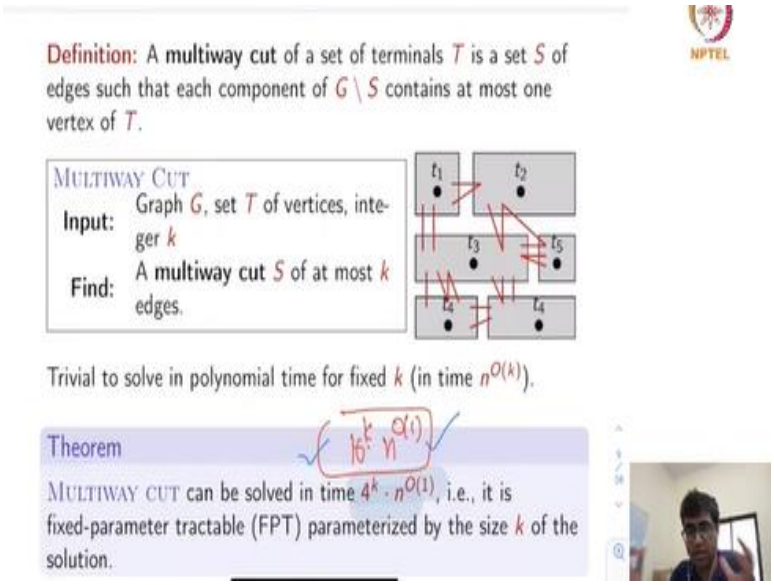
(Refer Slide Time: 03:48)



So, what are the multi way cut? So, multi way cut of a set of terminals T is a set of edges such that each component of G - S contains at most one vertex of T or alternatively we can say that for any pair of vertex in T there is no path after we deleted the edge set S. And it is as I told you that like for if you just give in to what he, say s and t, then this is polynomial because it is nothing but a min s, t cut problem.

But the problem was already NP complete for a fixed number of terminals T greater than equal to 3 and that is why we sold told that we do not expect to have an algorithm of this nature. In fact, we do not expect to have an algorithm for this nature, unless P = NP.

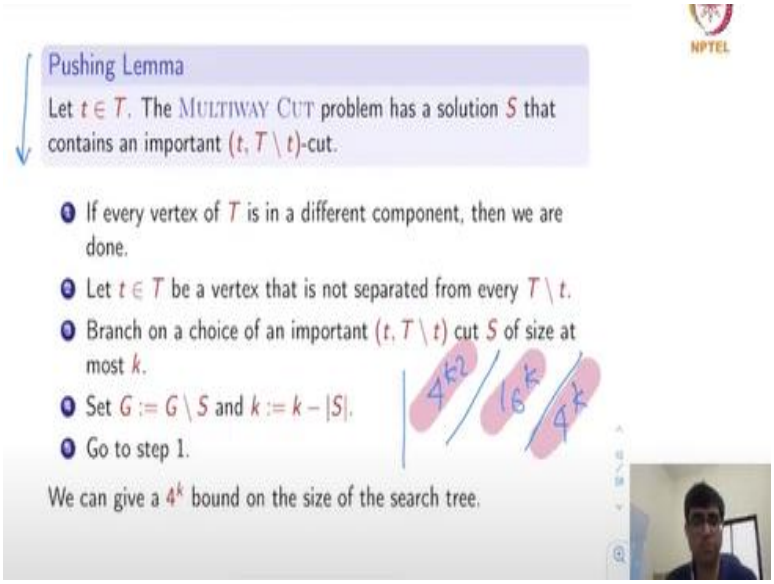
(Refer Slide Time: 04:44)



But if you recall correctly, we showed that it is trivial to solve the problem in polynomial time for fixed k, k you just try all possible cases subset and check whether it is a multiway cut or not. But we design an algorithm using important cut, with running time 4 to the power k into the power one. In fact, we did not design this algorithm, we actually we talked about this algorithm, but the running time which we properly analysed was 16 power k and to the power of big O of 1.

But that algorithm can be made to run in 4 to power k polynomially. So, this was a slight, so this is like covering of whatever we did previously.

(Refer Slide Time: 05:25)

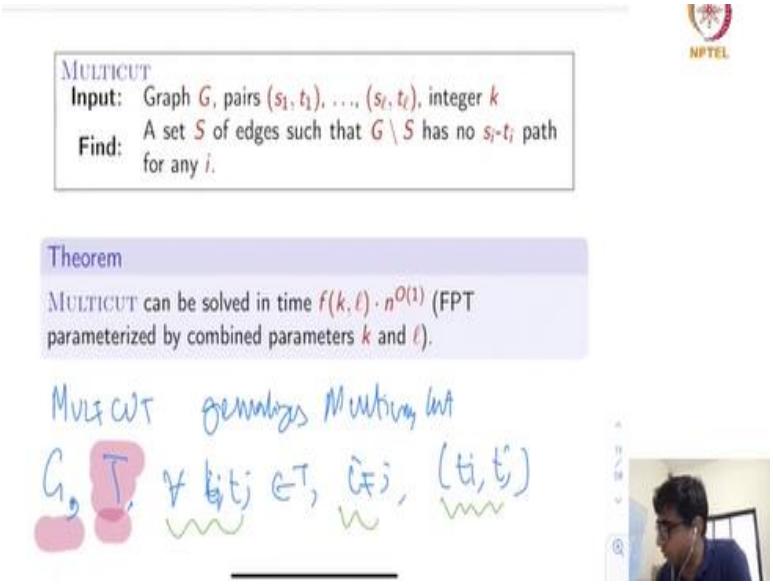


And now let us see what we do and here is your algorithm, so, recall the heart of an algorithm was the following pushing lemma. What are the pushing lemma? It says that look at any solution it definitely disconnects T from all of the terminals by some set of minimal cuts which is contained inside my solution. But what we showed that well, it is not any minimal cut we can find a solution where this minimal cut can be replaced by one of the important t, T - t cuts.

The moment we had we got this very interesting algorithm that basically said if every vertex T is in different component, then you are done else you pick up a vertex which is not separated from a vertices t, T - t, you enumerate one of the important cut of size at most k delete these records with a new parameter k minus cardinality state. And first, it is very trivial to design a 4 power k algorithm for this problem, 4 power k squared.

Because in each branch because the budget is decreasing by 1 the depth of the trees at most k and since each branching factor is 4 to the power k, the total running time is 4 to 4 k squared. But then we use us refine analysis to give 16th power k algorithm, but that algorithm can also be analysed even more carefully and we can get a 4 to the power k algorithm for the problem. So, that was the state of affairs till the first three lectures.

(Refer Slide Time: 07:02)

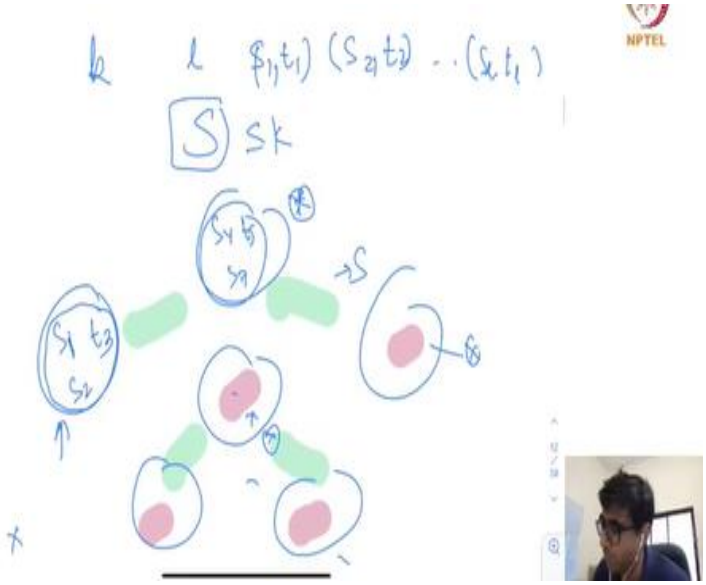


And let us see what up so, the first problem that we are trying to talk about today is what is called multi cut problem. So, what is a multi cut problem? So, unlike multiway cut problem, you

are not given a set of terminals, but you are given a set of request pairs, which are called s 1, t 1, s l, t l and you are supposed to delete at most k edges with a property that after you delete this there are no paths between s i and t i for anytime.

And why did they use the word that multi cut generalizes multi way cut? Because given a graph G and a set of terminal t we can create a request pairs by for every t i, t j in T, i not equal to G you will create a request were t i and t j. And what does that imply? That now you have reduced multi way cut to multi cut you could have each request physical with other giving a set of terminal. And now what I am going to show to you is an interesting algorithm.

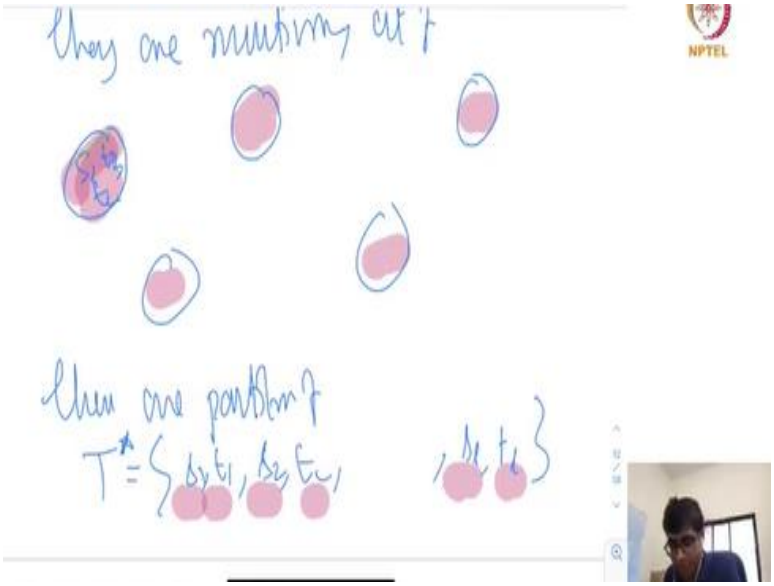
(Refer Slide Time: 08:00)



So, let us see how we can solve out multi cut. So, basically imagine that you have this set S and we have s 1, t 1, s l, t l here, once you delete this maybe there is a connected component which contains s 1, t 3, s 2, s 4, t 5, t 7 so on and so forth. So, after I delete this the set of terminal pairs s 1, t 1, s 2, t l gets distributed, but once this get distributed and this is your green colour or your edges what is the property of this.

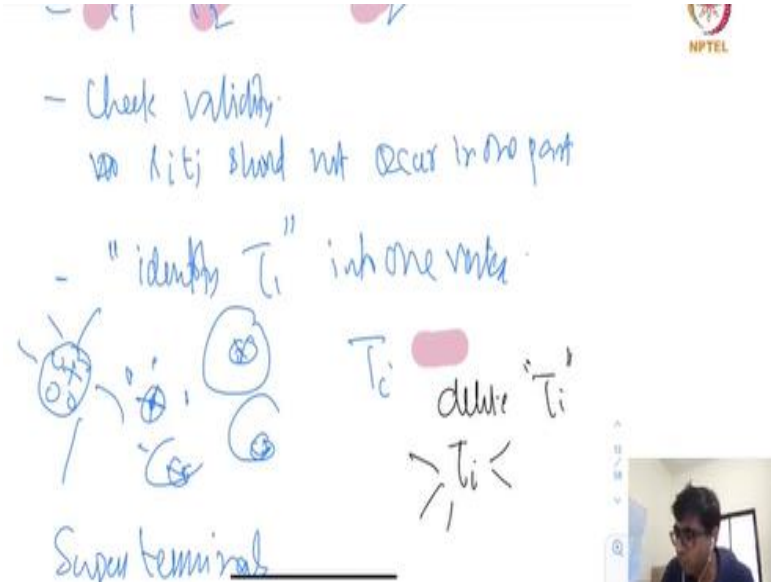
So, if a think of this set of s 1, t 3, t 2 as a subset of vertices, then my edges of S actually acts like a multi way cut.

(Refer Slide Time: 08:44)



So, this S is basically a multi way cut of this set of you know, this set of terminals here each of this. So, that immediately induces a partition of the T star, which we T start consists of s 1, t 1, s 2, t 2 s l, t l. So, what do I do? So, after I did the edge of s, the terminals in T star gets partitioned into different connected components.

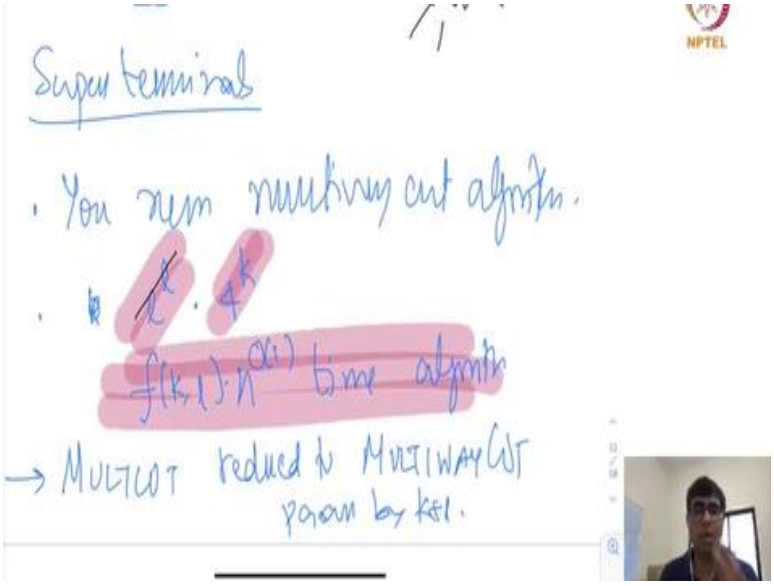
(Refer Slide Time: 09:17)



So, my algorithm is going to be you guess a partition and P start suppose they are T 1, T 2, T q, it is a partition of T star. Now, you first check is this a valid partition meaning if I delete the solution set S, can we get this partition. What is the meaning of it? S i, T i should not occur in the same part meaning they should not be the same part because what we are trying to do is to disconnect every vertex in T i from every vertex in T j.

But we are not caring about disconnecting vertices in the same pair. That is the model of describing this T i. Now we are going to identify T i into one vertex and which is very simple. You delete vertices of T i add a super node say T i and make him adjacent to every vertex any vertex in T i was adjacent.

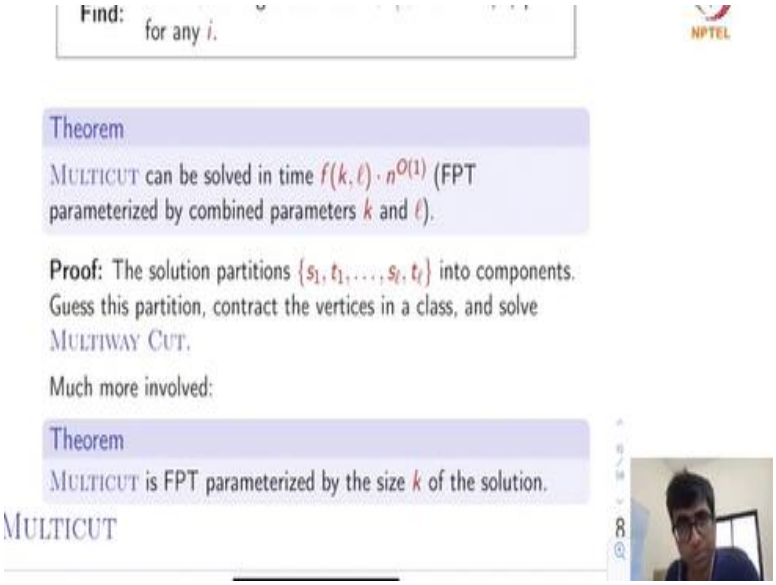
(Refer Slide Time: 10:11)



So, now for these super terminals what you are going to be done you run your multiway cut algorithm, but then what your running time of this algorithm because you know that at least one of the guesses is correct and for that, if you run this multi way cut algorithm then it should partition them into different parts of with utmost k. So, the running time of algorithm is guessing the partition which is l to the power l and then running this code to the power k algorithm.

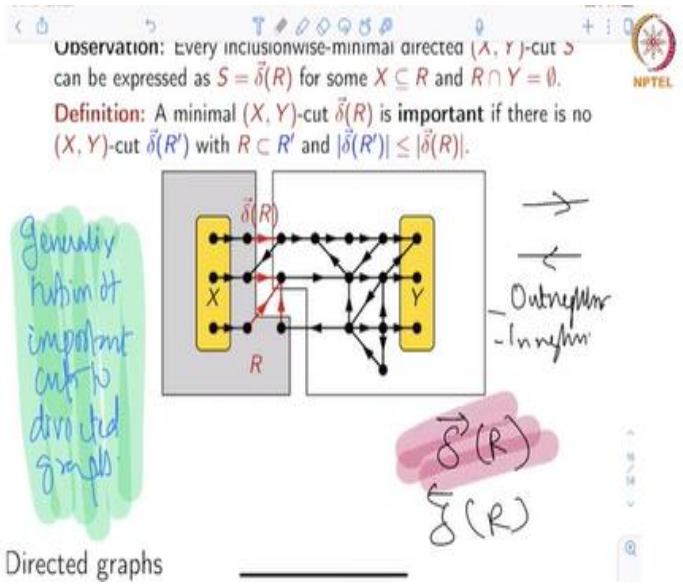
Which means that we can solve the multi cut in f k l into n power big O of one time algorithm. So, how did we solve multi cut? We solve multi cut by reducing it to a multi way cut parametrized by k + 1 k + l and this is how you show this theorem that multi cut which is a extreme generalization of multi way cut is FPT parameterised by number of request pairs as well as the solutions. So, what are the natural question to study them?

(Refer Slide Time: 11:13)



So, the natural question to study this is just one line proof for whatever we have been talking about. So, the one line solution is that whatever parameterised by just k, well multi cut is FPT parametrized by the size k solution, but unfortunately that requires much more tools and techniques than just important cuts and hence, we will not be covering this in this course. But I must insist or I must mention that it also uses the idea of important cut very crucially. But it also uses much more advanced tools than just important cuts.

(Refer Slide Time: 11:53)



However, we will take a slight turn, and what we are going to do is we are going to generalize the notion of important cuts directed graph. So, it is a very natural that a natural direction. So, because we are talking about directed graphs, we have all kinds of edges that will leave some set

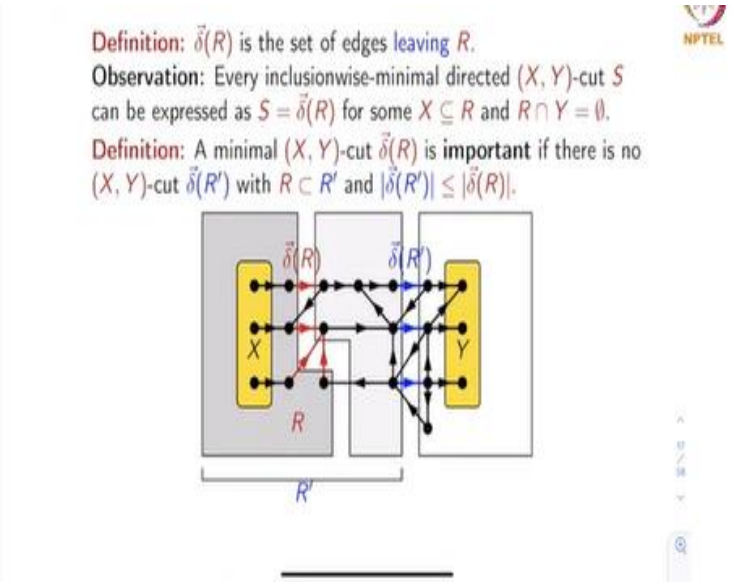
or edges which will, must be getting inside this size. Given on this you know that you have a notion of, out neighbours in neighbours.

And so, you can define delta out neighbours of set R delta in neighbour of set R but we will develop our theory without neighbours only. So, what is the delta R is a set of edges leaving R and as for undirected graph, we can show that every inclusion minimal directed X, Y cut S and by directed X, Y cut S you mean meaning you want to delete path from X a vertex X in X to a vertex of a vertex in Y. But you are not caring about deleting a path from a vertex in white to a vertex in X.

Then, you can actually; if you are if you want to cut then you can come up with a notion of S can be represented as Rs which are leaving a set R. So, for example, the set R is valid out, and then you naturally and of course, X is contained inside R and R intersection Y is empty set. Of course, there is a natural notion of extension of important cut. What is an important X, Y cut? An important X, Y cut will be a cut which is not dominated by other X, Y cut.

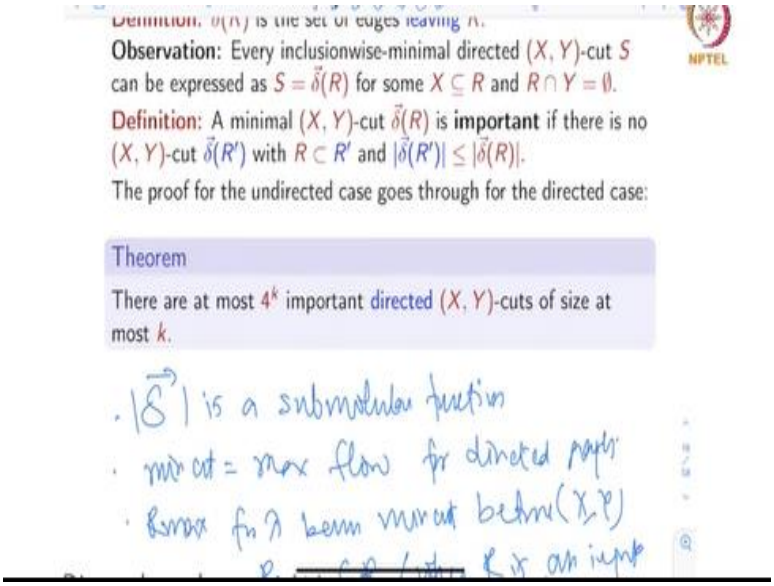
What is the meaning of this? I should not be able to find another delta R prime, R prime with a property that the number of out arcs from R prime is less than or equal to R and R prime properly contains the set R this we should be fine. So, you ask yourself is this R that you have drawn is an important cut.

(Refer Slide Time: 14:01)



Well, no, because you can come up with an R prime that that properly contains a set R and has at most as many edges leaving R prime as it was R, as in R we had 1 2 3 4 red edges but in R prime we have three blue edges. So, R is not an important cut but what about R prime? R Prime important cut. The moment you try to extend R prime by taking any other vertex number of edges leaving that set will be more than three and hence that is not an important.

(Refer Slide Time: 14:36)

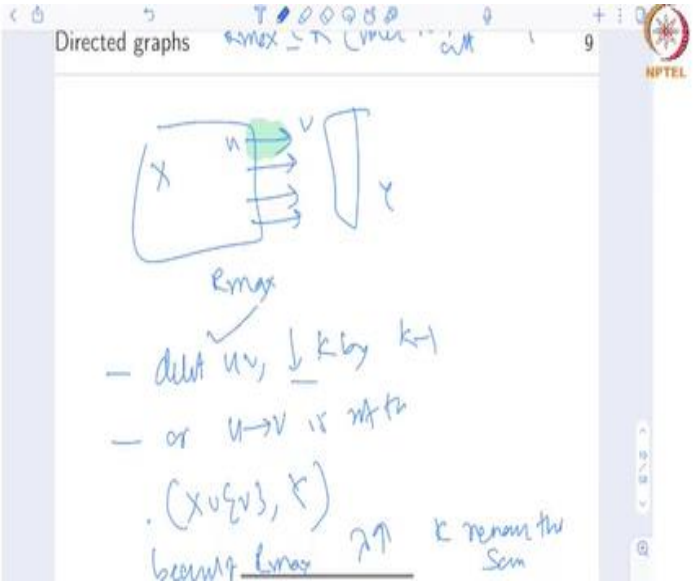


So, the proof for the undirected case also goes through for directed case and you can show that there are at most 4 to the power k important directed X, Y cuts of size at most k. And how do you go about doing this? Exactly like an undirected R if you show that delta hat is a submodular

function, min cut is again equal to max follow for directed graphs. You define R max executes respect to min cut lambda like extension.

Then you show that R max is contained inside every important cut in directed graph. So, it is a very, very good exercise that you go back look at each of the undirected lemma we proved and extend it to the notion of directed important cuts or directed min cuts the way we have defined.

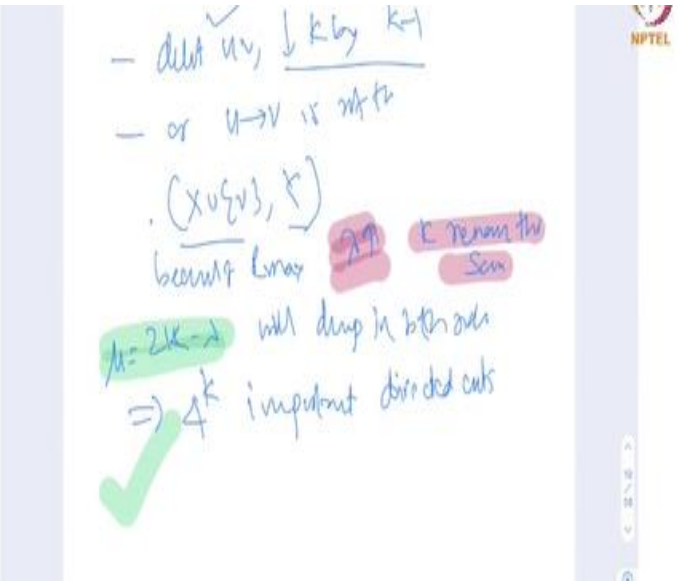
(Refer Slide Time: 15:26)



The moment you have this R max the different the algorithm becomes very same; you start with R max look at the edges which are leaving it say u, v. So, either the u, v is part of the important cuts you are thinking then you delete u, v you decrease k by k - 1 and recursively find all the important cuts of size at most k - in G – u, v or you know that u, v is not then you extend X by including v in and you look for all important cuts of X union v and y.

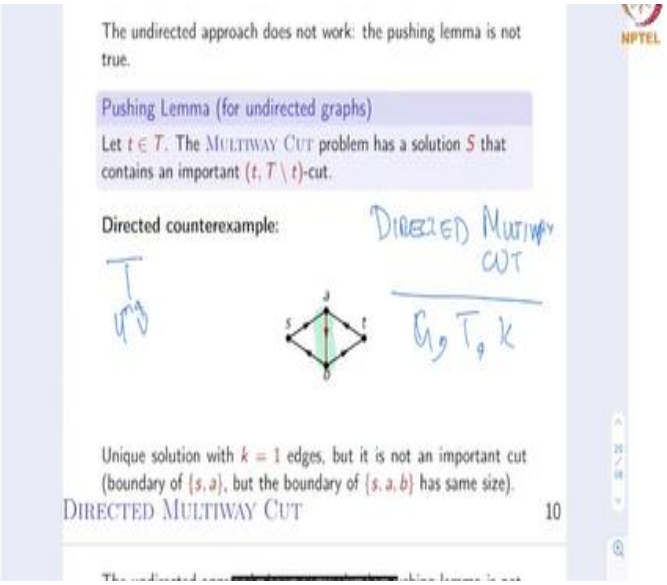
But because R max is there if you look at any set which contains R max the min cut from there to why is larger than lambda.

(Refer Slide Time: 16:05)



So, in this case though k remains the same lambda increases by 1. And so, if you look at the same major mu 2 k - lambda 2 k - lambda will drop in both case. And hence, you can show that the number of important directed cuts are upper bounded by 4 to the power. So, as again, it is a very important exercise to go back to every lemma of undirected graph apply with this new notion of directed cuts and check that it works.

(Refer Slide Time: 16:38)



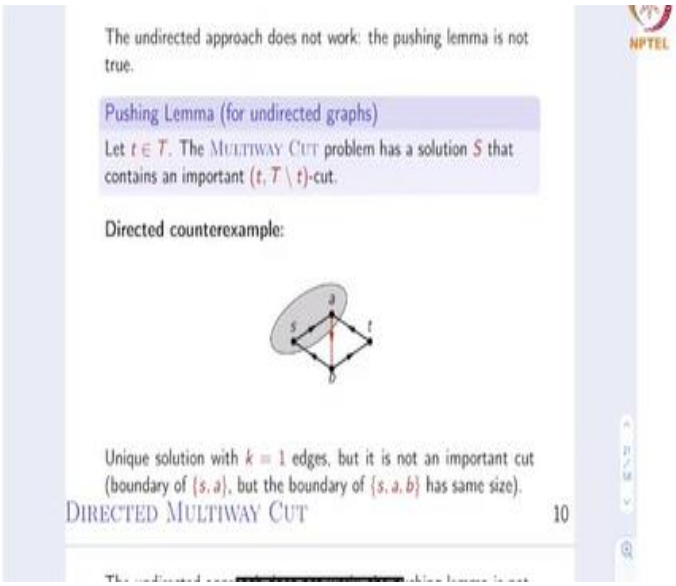
So, suppose I wanted to solve not undirected multi way cut, but directed multi way cut. So, what is the director multi way cut? So, in the directed multi way cut, you are again given a directed graph G, a set T and k and you want to delete k vertices So, that between any pair of vertices in T

there are no directed path. But notice that when it was an undirected graph T, if you have to what said u and v in T.

Then deleting a path from u to v is same as v 2, but in directed graph it is not same, you have to kill paths from u to v as well as kill paths from v to u. So, so now I am going to show to you that so the main crux why the undirected multi way cut algorithm works is that look, any cut must disconnect t from T - t by a some minimal cut, but we can replace that minimal it is cut from t to T - t by an important t T - t cut.

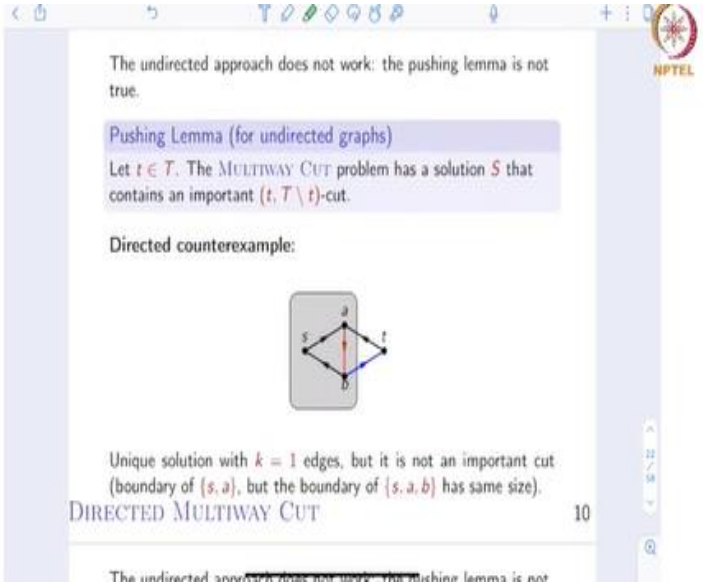
But we will show in a minute that such an object is not possible to show for directed multi way cut and hence, we cannot apply just an important cut algorithm and hope that it will work out perfectly fine. So, look at look at this instance of that multi way cut, s 2 t, we have to disconnect from s 2 t So, there is a unique solution with k = 1 edges a that is deleting the edge a, v.

(Refer Slide Time: 18:21)



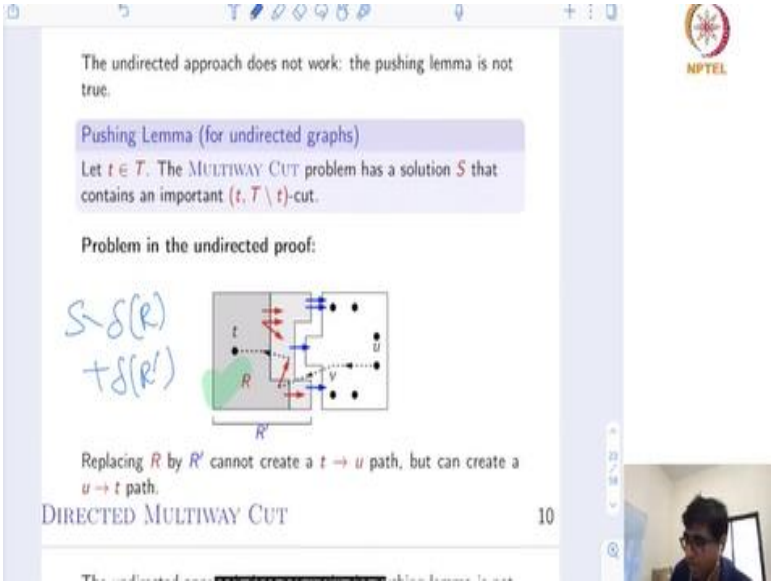
But what happens look, so, this solution is given by the Rs s, a. But it is not an important cut because important cut boundary of s, a is this R, but the boundary of s, a and boundary of s, a, b has the same one S curve.

(Refer Slide Time: 18:36)



So, definitely s, a, b is like s, a is not an important cut because s, a, b dominates. But you know, but like if you delete these blue edges is not part of any solution of size at most one. Because there is a unique solution of size at most k, but then you may ask why is it failing?

(Refer Slide Time: 19:02)



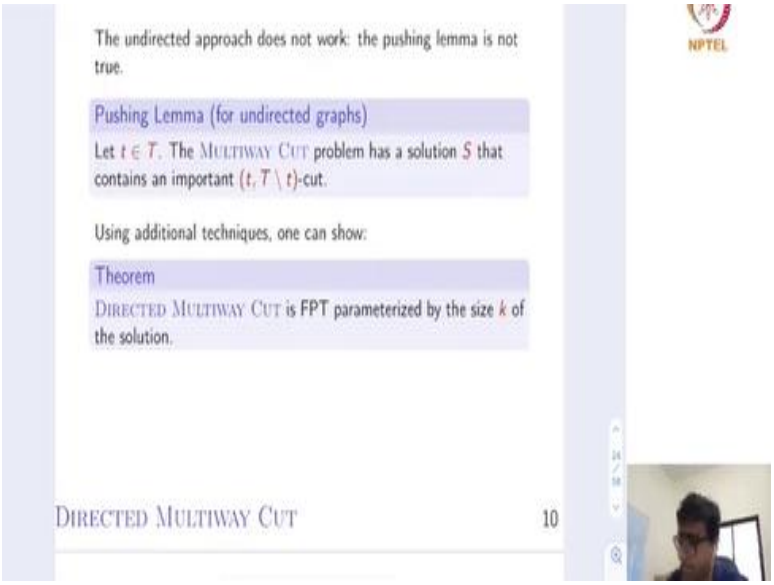
So, if you just try to look at the undirected graph lemma and try to mimic the proof you will be able to see why this proof does not work. So, how did the proof work? So, he said look, look at t look at T - t look at all the vertices which are reachable from t the let us call that R these are reachable vertices. If delta if R is not an important cut, then there is another important cut R prime. So, let us extended.

And what you did the way you created a solution is that from S you deleted you deleted delta R and added delta R prime. Now and then you argued that this alternative solution but we have to be very careful, it is indeed true that there is no direct path from T to any other vertices T - t but nobody stops from having a path from some u using like I can, I could use like I could have a path from u to t.

How because I mean I could have a path from u to some other vertex inside R prime or R. And since R is a edge because any such path must contain red edges, but the moment it will contain a red edge you can jump into R and from there because this vertex is reachable from it you can find a path from here to here and you can reach but create a u, t R. And like I mean this is not right, but what I am saying that it could possibly happen that I can jump from you to this.

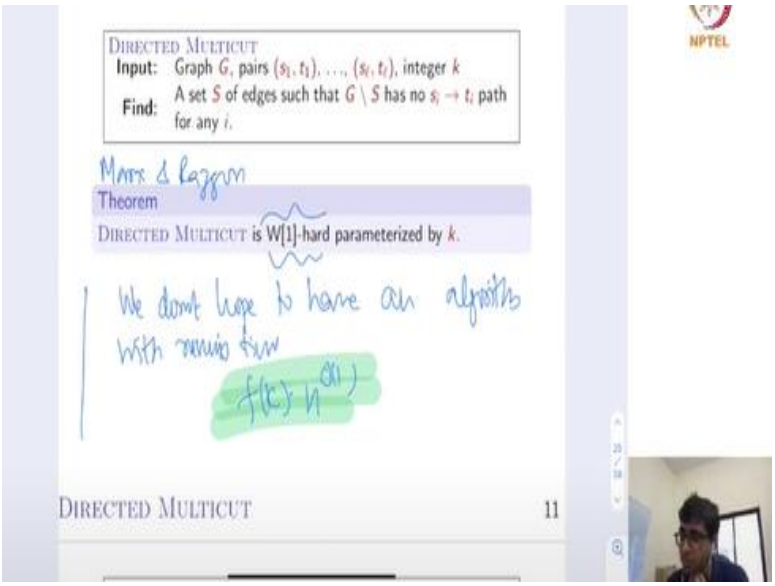
And you do this I can come inside are because of the backward arc and from there I might be able to reach it. It is possible I may not be able to reach but I might also be able to reach and that will violate that s prime that you have constructed like this is a solution.

(Refer Slide Time: 20:45)



It is indeed possible to show the director multi way is FPT parametrized by size k of the solution. But it requires again additional set of techniques which is very similar to the techniques we need for undirected multi cut, but again it is beyond the scope of this course. So, we will not talk about it.

(Refer Slide Time: 21:04)

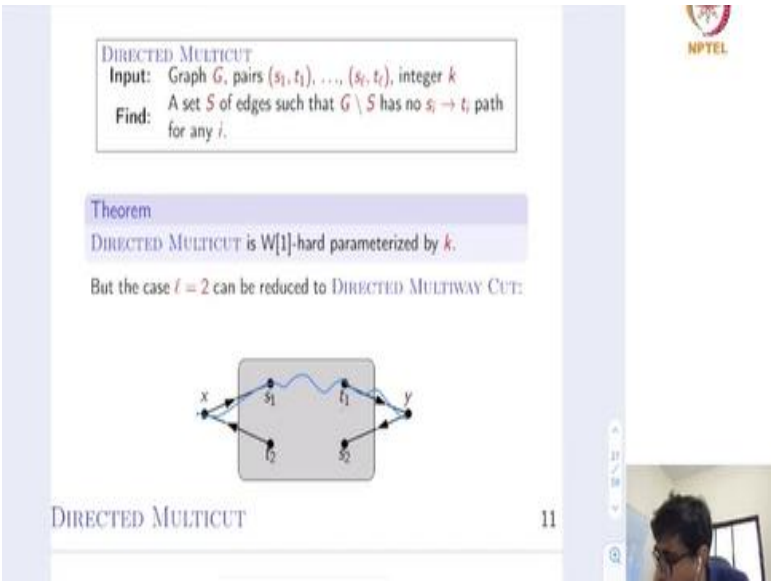


What about directed multi cut? So, what is the directed multi cut again? It is like an undirected multiple cut, you are given a directed graph D, you are given a request p or s 1, t 1, s l, t l interior k and you need to delete a set of edges that G - s has no s i, t i path for any. If you notice, this also generalizes directed multi way cut, as a gate; because for a set p you write all possible pairs like on ordered pairs as a request pair.

And that that is an instance of predicted multi cut and then you are done. But marks and Rajgon showed the directed multi cut is in fact W 1 hard parameterised by k. What is the meaning of this? It is just mean that we do not expect to have an algorithm which running type f of k only in and that is all that. So, whenever you see W 1 hard, W 2 hard, W 3 hard. It just means that you should not expect the problem to be fixed parameter tractable.

Parametrized by the parameter in which the; problem is shown to me some w hard. We talked about this little bit in our first lecture and we will cover this in detail in the last week of our course.

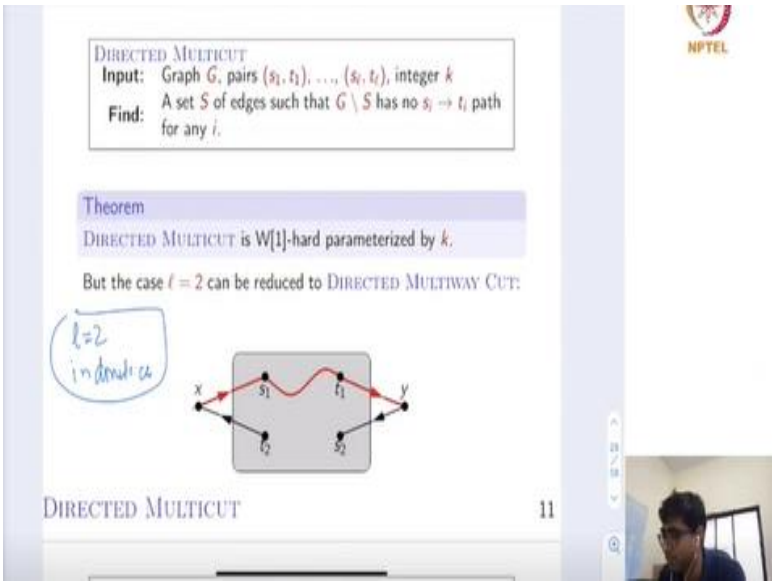
(Refer Slide Time: 22:21)



So, directed multi cut is W 1 hard parametrized by k, of course, we cannot expect the problem is FPT just parameterised by l, because even for l = 2 the directed multi cut problem is that one is NP complete and so on. But what about the case l = 2? What about directed multi cuts when the number of requests pair is constant or we can use number of request pairs to be a parameter what happens? So, l = 2 actually can be reduced to directed multi cuts.

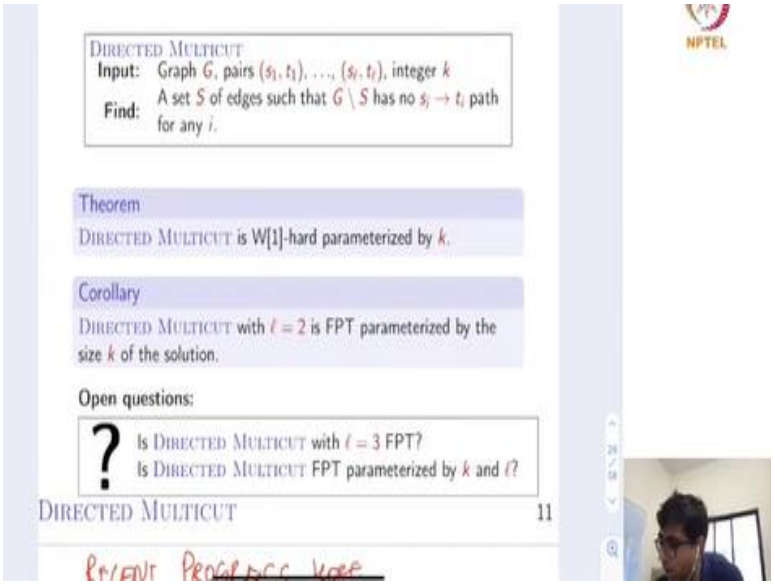
So, you have like you want to kill all the parts from s 1 to t 1 and s 2 to t 2. What you do? You add two additional vertex x and you add an edge from x to s 1. So, basically and t 1 to t y, so, basically this is trying to capture all the paths from x to y and you add t, t to x and h and y to s 2 and this is trying to capture path from y to x.

(Refer Slide Time: 23:18)



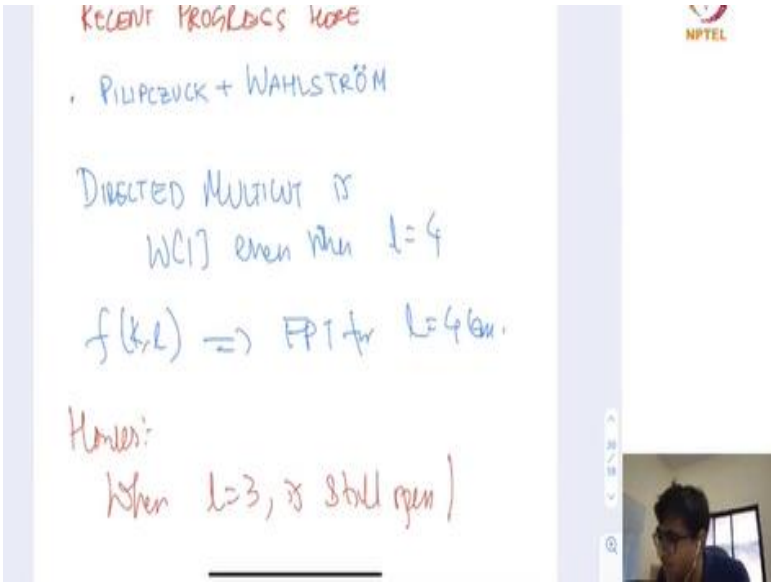
And you can show that directed multi cut is equivalent to directed multi-way cut with terminal set T x, y. So, it is a very simple reduction or they say please try to do it yourself. But, so, what does this means? That for what then the natural question that will arise.

(Refer Slide Time: 23:47)



Are whatever directed multi cut with l = 3? Whatever directed multi cut is directed multi cut FPT parameterised by k and l?

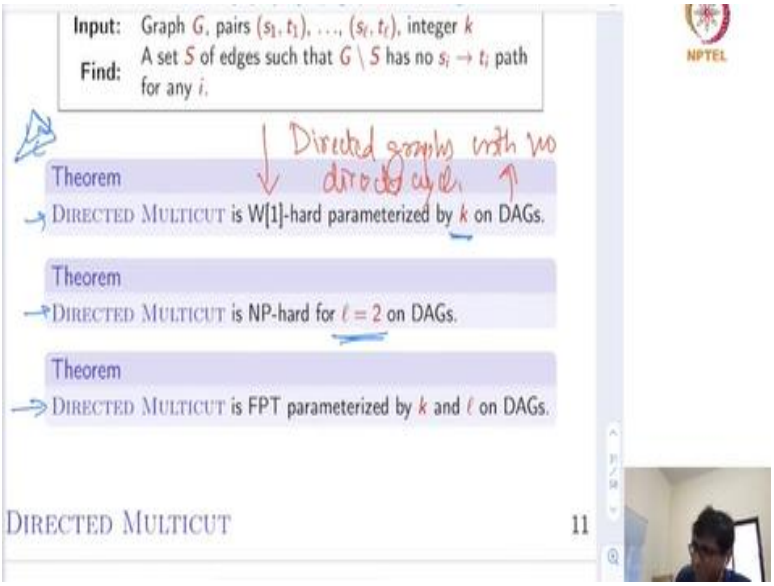
(Refer Slide Time: 23:58)



And some of the recent progress, not recent progress per say, but like in roughly in 2016 Philip juke and Wahlstrom showed the following, the directed multi cut is W 1 hard even when you have l = 4 terminals. So, what does it imply? It immediately implies that you cannot have an algorithm parameterize by f k, l because that will imply FPT for l = 4 case. However, when l = 3 it is still unknown whether the problem is FPT or W 1 hard.

So, this problem still remains open. So, what is the parametrize complexity status of directed multi cut? When the number of request pair is upper bounded by 3. And of course, if it is like all the six like this is like one of some of those cases are definitely N p, t because of the directed multiway cut.

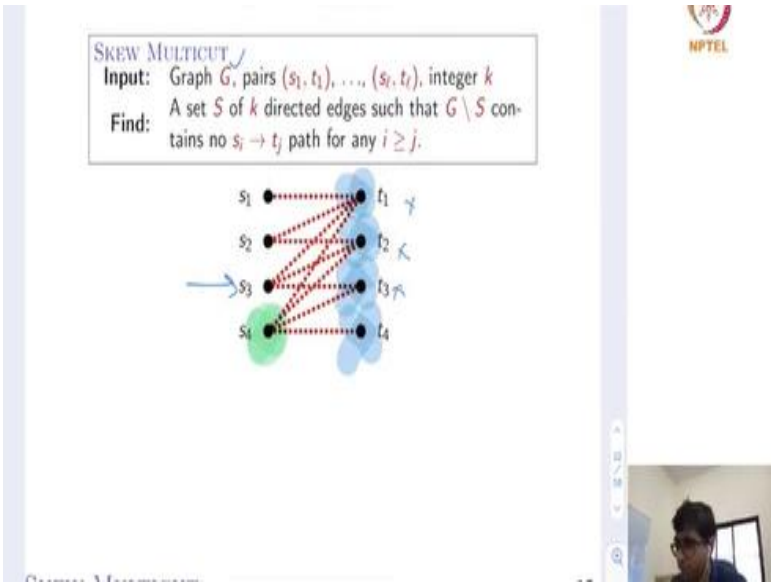
(Refer Slide Time: 24:55)



So, that led to the study of directed multi cut on DAG. So, what is DAGs? So, that is a basically directed graph with no directed cycles. So, for example, like this is not a DAG. This is a DAG, although it is an undirected size directed size, but there is no directed cycle in sense of directed. So, it is known that directed multi cut is actually W 1 hard parameterised by k even on directed cyclic graph.

Directed multi cut is NP hard for l = 2 on directed acyclic graph and directed multi cut is FPT parameterised by k and l on that. So, it was shown that though the directed multi cut may not be a is W 1 hard on general digraphs parameterised by k and l. In fact, even for l = 4 terminals, it is W 1 hard. But it is FPT parameterised by k and l if we take directed cyclic graph a graph that does not contain a directed cycle. So, extending this algorithm to other classes of diagrams is an interesting open problem.

(Refer Slide Time: 26:10)

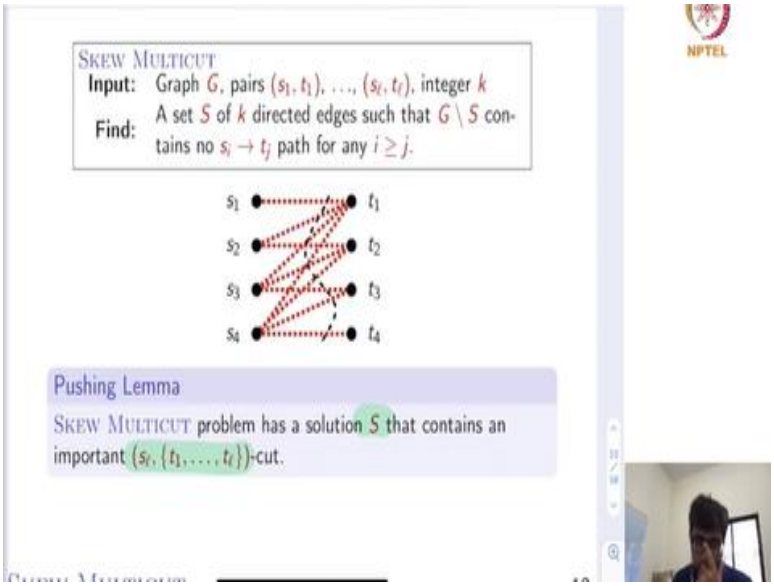


But here is a very interesting special case of directed multi cut which is called skew multi cut, which has been extremely useful algorithmic tool in directed graph. So, what is skew multi cut? So, in skew multi cut, you are given a graph G you have pairs s 1 t 1 s l t l as before and t r k. But you are looking for a directed edges says that G - s contains no s i t j path for any item. What does it mean of this?

So, basically what I mean to say is that look, if you have s 1, s 1 should not have a path from s 1 to t 4, t 3, t 2, t 1. Similarly, s 3 should not have path from t 3, t 2, t 1. But s 3 could have a path from t 4. So, like so, basically what it means that, if I look at s i, then s i should not have a directed path to any t i whose index is lesser or equal to the s i. But it does not care about killing paths to higher index. That is important point.

So, s 4 must, like we are looking to delete set S of edges so, that there is no path from s 4 to t 4, t 2, t 3, t 1, but if s 3 to t 2, t 3, t 1 but s 3 path maybe they are disconnected, or maybe there is one, we do not care. So, this is what skew multi cut is. So, I would request you that you pause this video for a couple of minutes and adsorb the definition of skew multi cut properly.

(Refer Slide Time: 27:51)

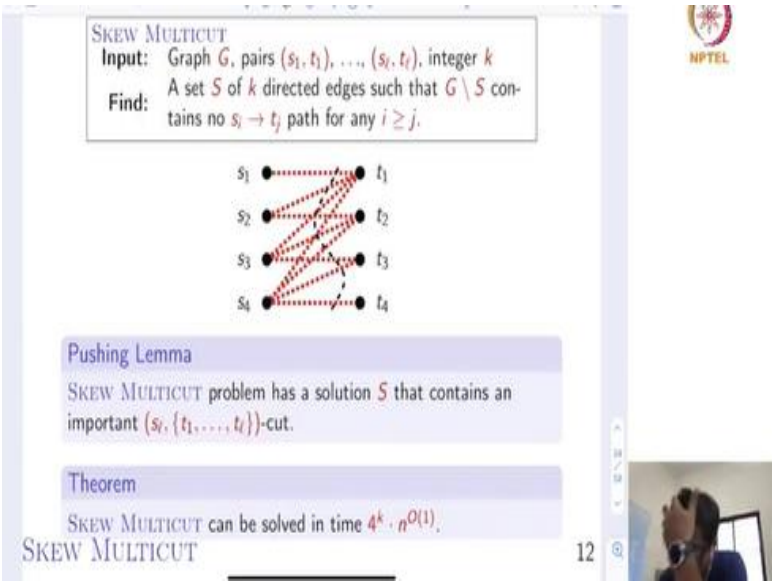


So, recall what was the main reason undirected multi cut was FPT because we were able to show to you, that you fix any terminal and fix other set of terminals. Then there exists a solution that contains one of the important s t T - t cut. Now, I cannot show this for any s i and the set of requests it needs to get. But I am going to show to you that at least the following statement is true the skew multi cut problem has a solution s that contains an important s l t 1 t l.

What is s l? s l is the highest indexed, highest index vertex s. So, s l needs to kill all the paths to all the T is. That is important, I am saying to you that look, this must contain an important s l t 1 to t l. Suppose we prove this lemma, then also we will be able to design the skew multi cut how because rather than picking any arbitrary terminal, we will pick a terminal which is highest and which still needs to be disconnected from everybody else.

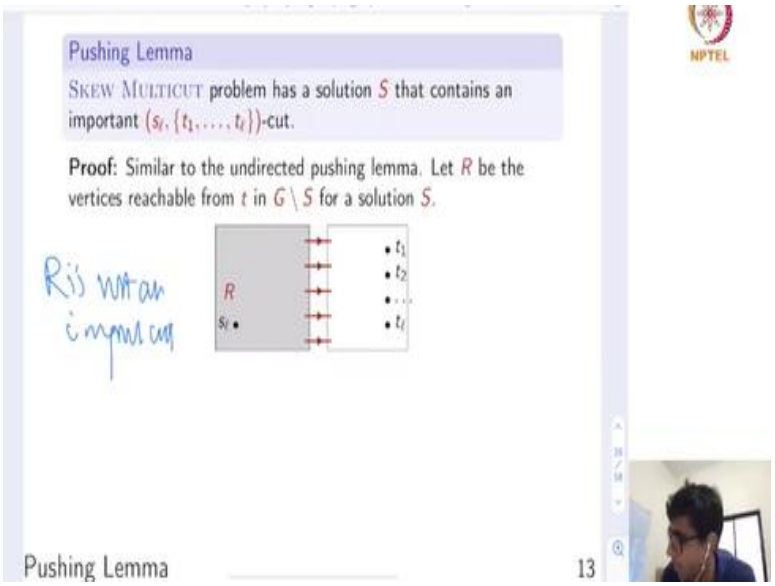
So, you rather than starting from anything, you will first take s l try to disconnect from everybody else, then when you are the job of s l is done, you pick up another then you ask for s l - 1, hey, are you disconnected from whomever you wanted to? If yes, then you move on to someone else. But so, you pick up the highest s i which is still connected to one of the T i s that it knows that it needs to be disconnected and you try to disconnect those. That is the point. So, let us try to prove this lemma.

(Refer Slide Time: 29:44)



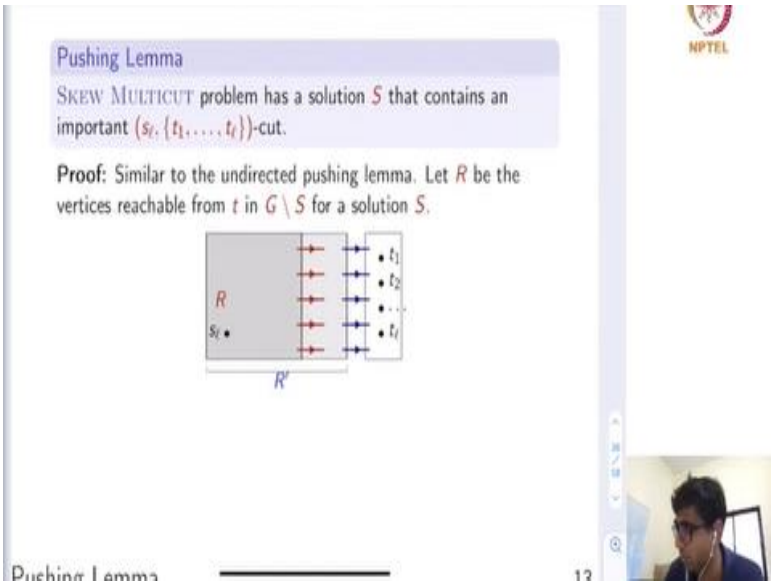
And the moment if we are able to show this, we will be able to get an algorithm with running time 4 to the power k poly n and for skew multi cut. Let us see how we go about.

(Refer Slide Time: 29:55)



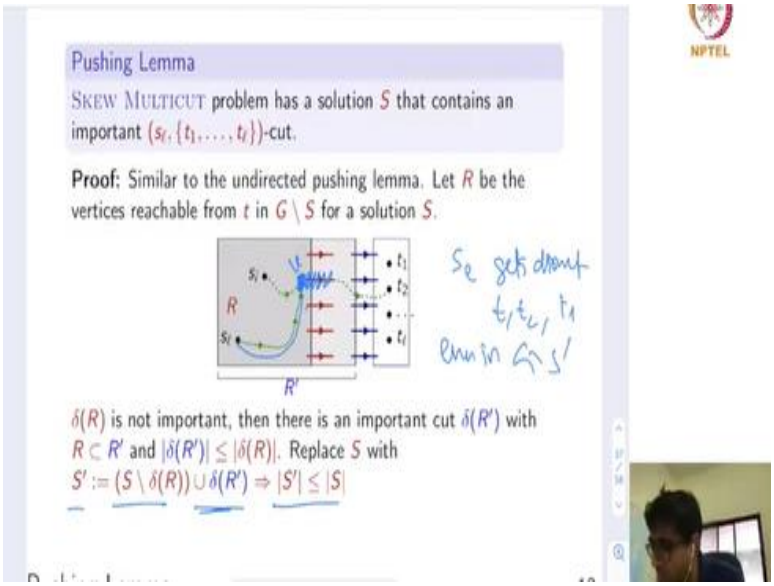
It is exactly similar to the undirected pushing lemma. So, let R the vertices reachable from T and G - s for a solution s what happens? Either R is an important cut or R is not an important cut R is important cut then you are good if R is not an important cut in what can we say.

(Refer Slide Time: 30:19)



If R is not an important cut, then there exists some R prime which strictly contains R disjoint from t 1 to t l.

(Refer Slide Time: 30:29)

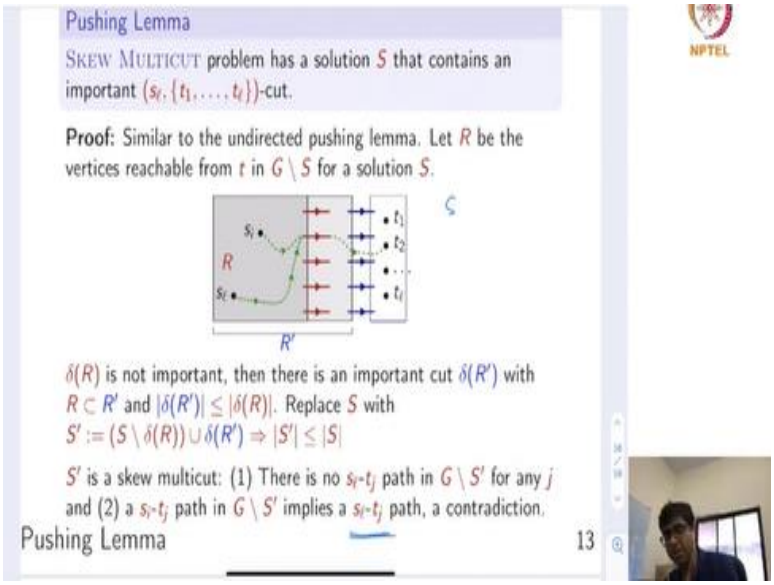


And delta of R prime is at most. So, now you create another s prime you remove delta R and you add delta R prime. Clearly you remove some set of edges and you added some set of edges but the number of edges that you added is smaller than what you replaced removed which imply that s prime at cardinality by is perfectly fine. Now, why is this not a solution? Suppose this is not a solution, what is the meaning of that not a solution.

So, definitely S l gets disconnected from t 1 to t l even in G - s prime. But what could happen that now, you might get a path from s i to t i for some s i t i, why? But look when I deleted red edges, it was still like when I deleted s minus, when we deleted s it was a solution when there was no s i t i path, the whatever part you need to cut. But certainly, you have got a path it means it must be using a red edges.

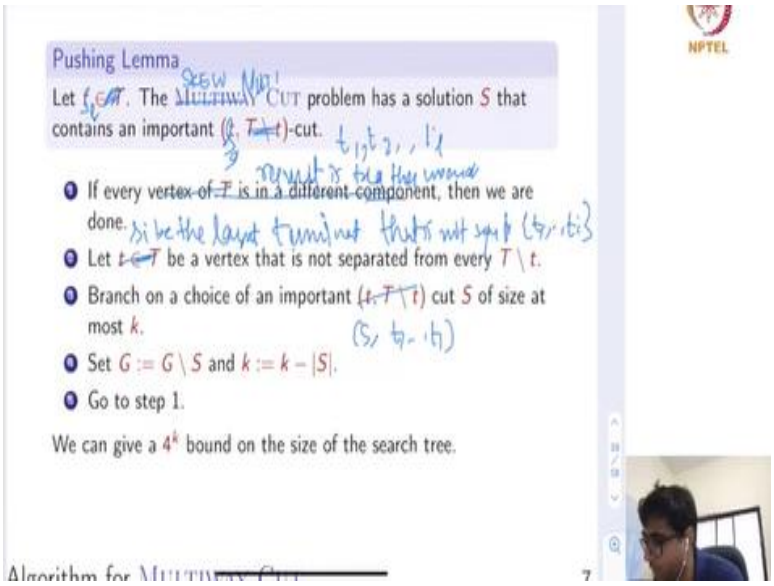
If it must be using a red edge, it must be using one of the red edges it means that path from s i t to t i must be coming inside this R prime or in fact in R but what is the property of that vertex in R, s l it this vertex is reachable from s l it means the directed path from s l to this vertex. Then I can get a path starting from you go to s l to v, now on this s i t i path you follow the path from v to t i. So, what you have been able to achieve?

(Refer Slide Time: 32:19)



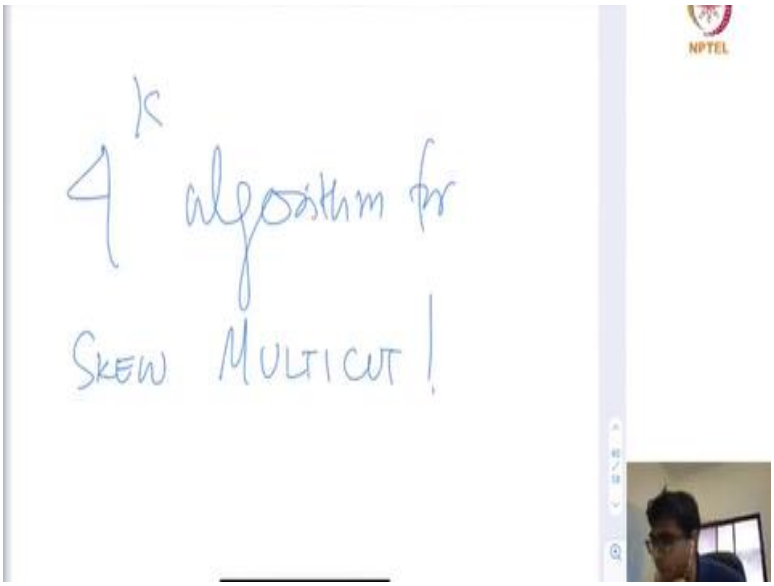
You will get a s i t j path implies an s l t j path. But you know that in G - S prime there is no s l t j path. And this is where we use the fact that s l is the largest because s l needs to disconnect itself from each of the t i. So, existence of any s i t j is a path implies s l t l path which also my S prime needs which also S prime takes care and hence S prime is a solution. So, this is how you can show that.

(Refer Slide Time: 32:53)



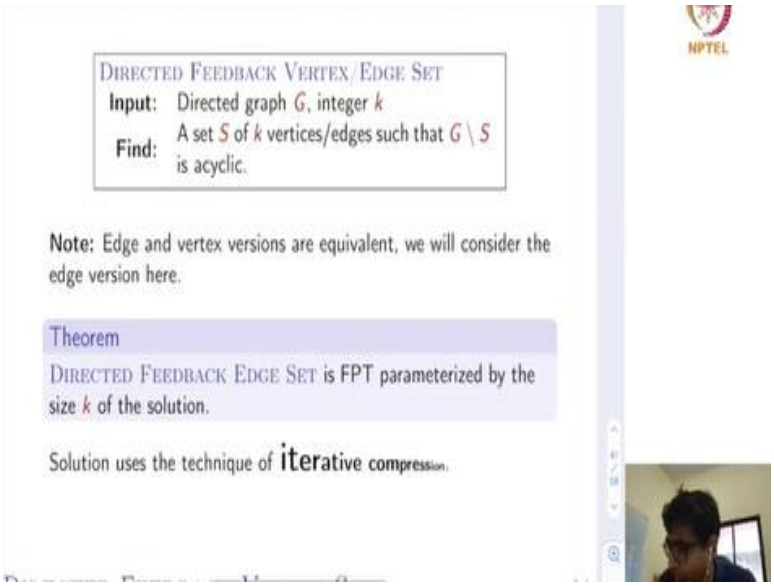
So, this immediately implies the following algorithm for this let you fix an s l the skew multi cut has a solution that contains an important s t 1 to t l cut. And if so, rather than having if every vertex like if every request pair is taken care of then we are done. As you choose the largest amyl that needs to be separated from t 1 to t i branch on a choice of s 1 to t 1 t i important cut of size at most k delete that records and this. So, this will give you a photocopy (()) (33:24) algorithm for such that and that is it.

(Refer Slide Time: 33:27)



So, that implies a 4 to the power k algorithm for skew multi cut. So, this gives us a 4 to the power k algorithm for a skew multi cut. Why do we care about this?

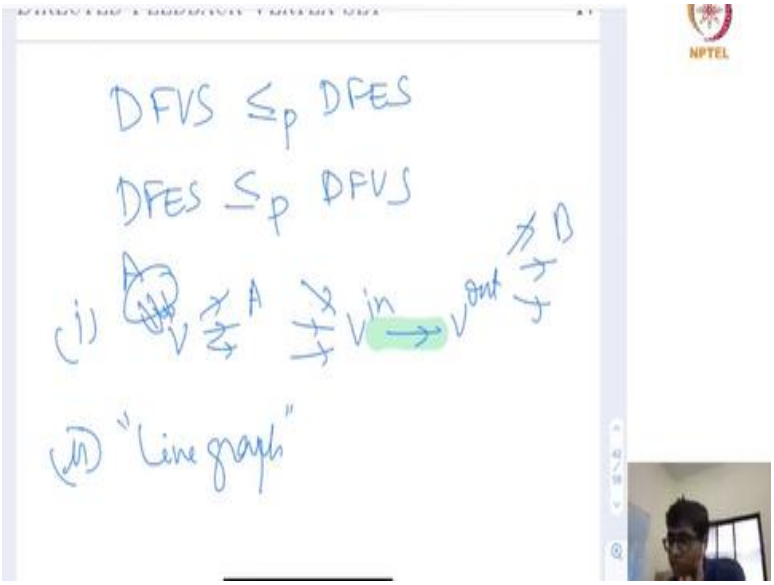
(Refer Slide Time: 33:38)



We care about this because it allowed us or it helped us rejoint the parameterized complexity of one of the most important open problems in the area about directed feedback for edge set. So, what is the directed feedback vertex or edge set? So, this is a counterpart of let us say, this is a counterpart of undirected feedback vertex. So, what is this? So, directed graph G integer k and you have an objective of deleting k vertices or edges.

So, that G - S is acyclic. It is known that the problem is what is that called it is known that the problem is FPT like not FPT but it is known that the edge version and vertex version (()) (34:23).

(Refer Slide Time: 34:23)

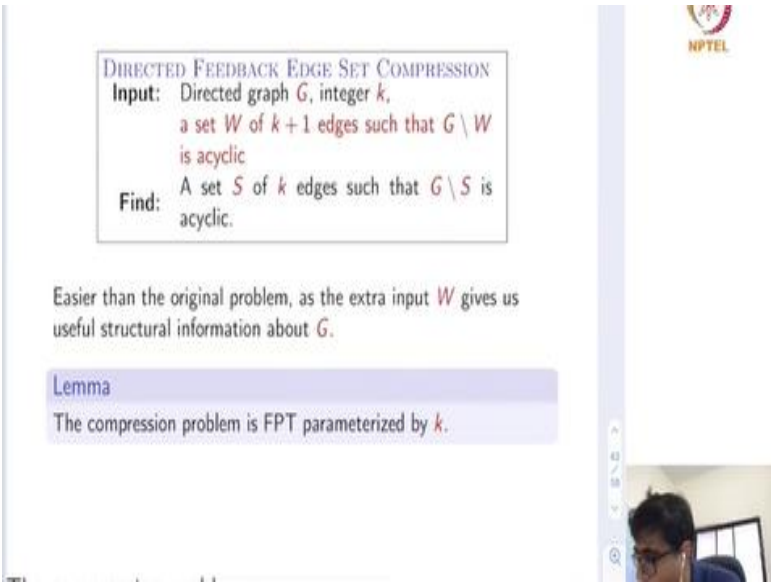


And what I mean by this. So, let us say directed feedback vertex set reduces in a parameter (()) (34:29) way to directed feedback S set and directed feedback S set reduce it in parameter (()) (34:35) in directed. The reductions are fairly simple. So, let me tell you the first one is basically based on the following fact that you look at any vertex V. So, you make V in V out and make every vertex which is in neighbours say A to V to V in and every out neighbour you make from V out.

Now you can show that if you have a k vertex if and only if there are k (()) (35:07) x and the x which you will form in your solution are these are. You should be, and for the second one what is known as line graph. So, I believe that and I will ask check asked you to check. So, in directed graph these two problems are equivalent which is not the same an undirected graph. In undirected graph if you want to delete edges it is nothing.

But just finding a maximum forex or basically, you find a forex you have to delete everything else. So, while edge versus poly time vertex versus NP complete in undirected or so, but for directed graphs, these two problems are basically same, but you can come up with the splitting operations.

(Refer Slide Time: 35:47)



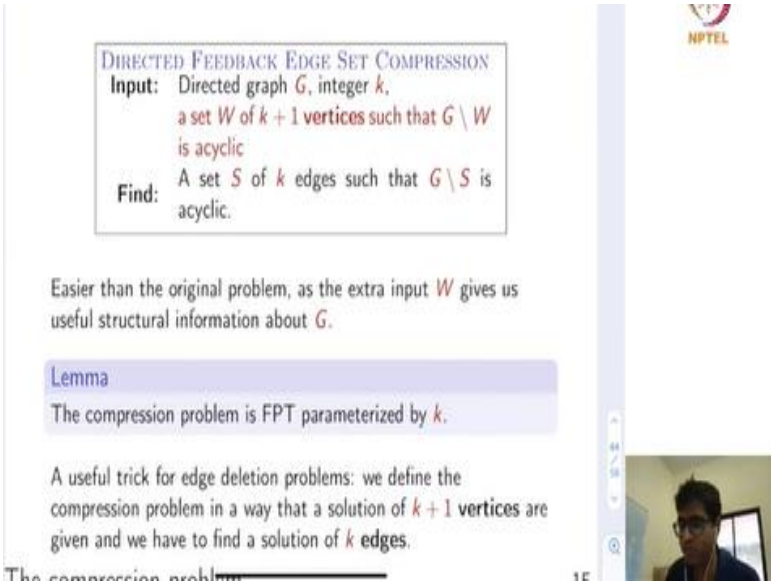
So, this the first problem where we will apply more than one tool to solve the problem and we are going to solve the problem by applying it FPT compression. So, what is going to be at FPT

compression? We are going to assume that you are given some k + 1 edges so, that our G - S is such that G - S is acyclic it is easier than the original problem as extra input W gives a useful structural information about G.

So, we are going to show that the compression problem is FPT parameterize by k. So, let us reread the problem, you are given a set W of k + 1 edges G - W acyclic a set of edges a G - S. So, the moment we will be able to solve this problem we will be able to solve directed feedback. Now, you may say from where are you getting this k + 1 edges so, if you recall correctly for the for the compression algorithm, we were having these vertices we want to v n.

And we took the first i vertices look at the graph induced on that so on and so forth. But now we will build this graph one arc by each and the moment we will add one arc will say hey. So, then you can if the smaller set of a smaller graph is a solution of size at most k, then that is correct you added the k + 1 arc and that will be freed up. But we will do this formula, but for now just assume that you are given a W, how can you solve this problem.

(Refer Slide Time: 37:27)

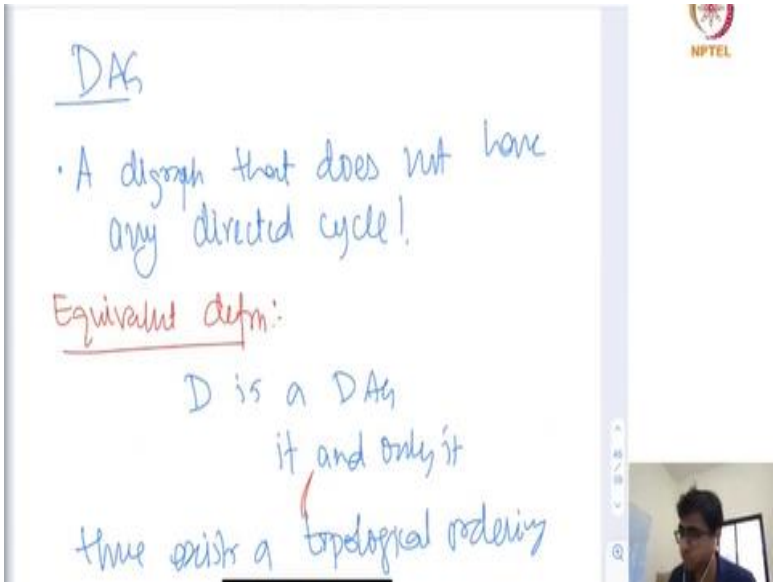


In fact, we are going to not assume, we are going to assume that what we are given is k + 1 vertices, the G - W is a set. And we want to find out k edges G – S is acyclic and this is a useful trick for to have an edge deletion. You define a compression problem in a way that the solution

of k + 1 vertices given and that is very easy, all you need to do is look at these k + 1 vertices. So, just pick one vertex each from this k + 1 edges.

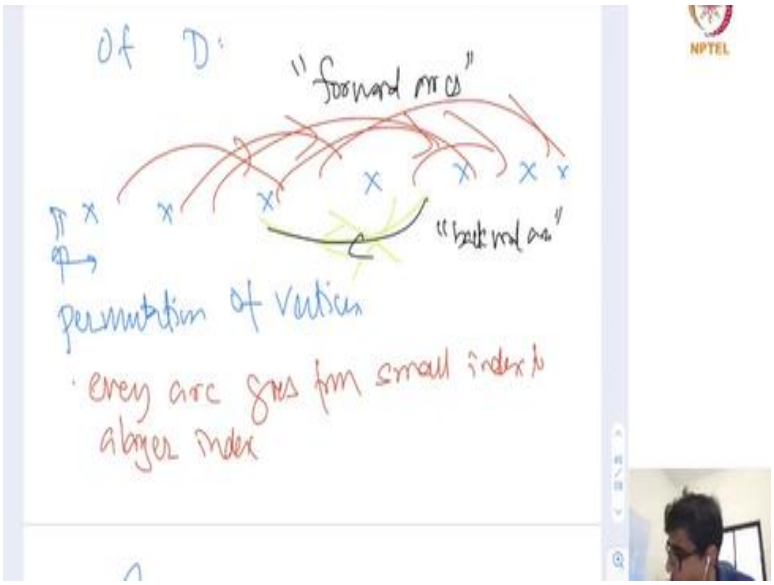
And then if you delete it, these edges are taken care of. And hence that forms a solution for many problems. It is not always true, but for many problems this trick will help you to reduce that you are not given edged but you are given a set W and you try to bring your compression problem with respect to edge. So, the way to look at it compression is that the extra set W is like a help or it provides an extra structure to our problem can be explained that a structure to design our algorithm.

(Refer Slide Time: 38:41)



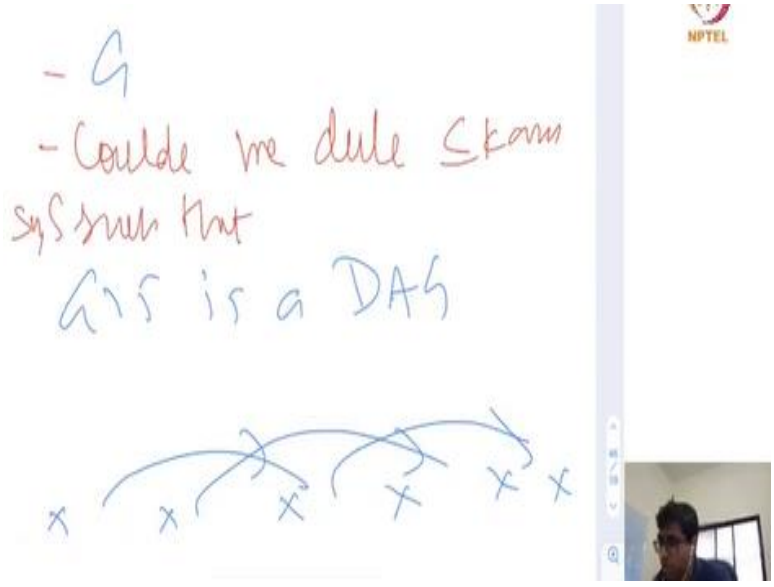
So, before we go further, let me tell you what was a DAG? A DAG is a digraph that does not have any directed cycle and what an equivalent definition? That these a DAG if and only if there exists a topological ordering of D.

(Refer Slide Time: 38:57)



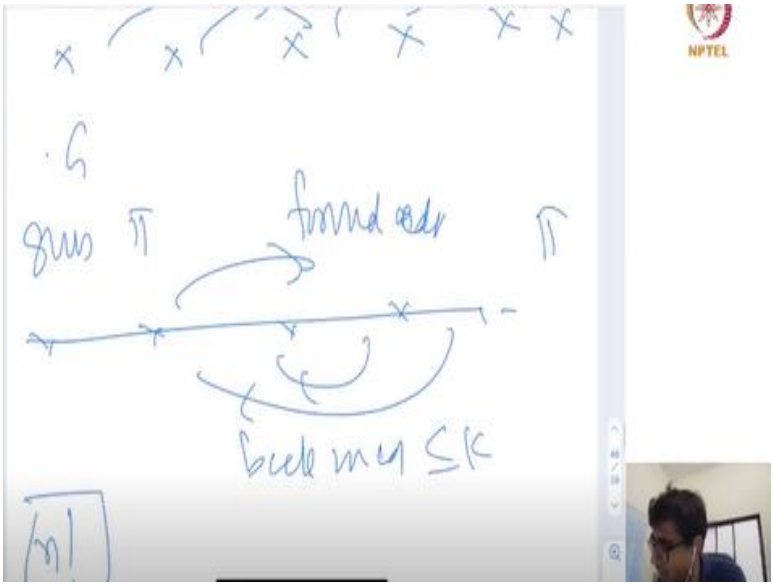
And what is the topological ordering? It is basically a permutation of vertices such that every R goes from a small index vertex to the higher index vertex. And so, these arcs are also called you must have seen these are arcs are also called forward arcs. But if I give you a normal digraph, I may not have I then it cannot have an ordering so, that every arc is forward. Then some of the arc may be going from higher to lower and these are called backward arcs.

(Refer Slide Time: 39:44)



So, if you are looking for G and we want to delete at most k arc and G - S is that.

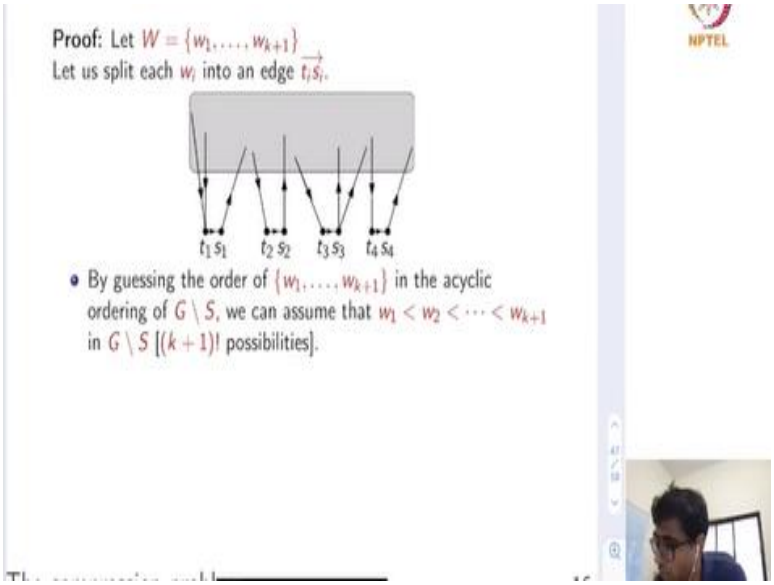
(Refer Slide Time: 39:54)



Then basically what we are asking is that, can we find a permutation pi such that number backward arc is upper bounded by k. So, this automatically gives us a very simple algorithm which is like n factorial you look through all permutation and check if there is one permutation with utmost k backward arcs. Now we are it is an n factorial algorithm but what we are going to do is that we are going to explore this to design not n factorial but FPT kind of algorithm.

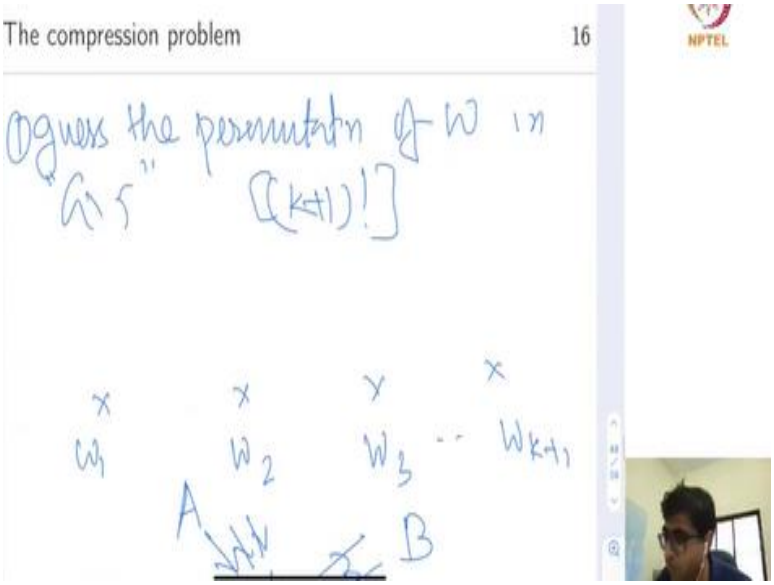
But we will take this view of feedback arc set that give me a permutation of my vertices such that the number of backward arcs bounded by.

(Refer Slide Time: 40:38)



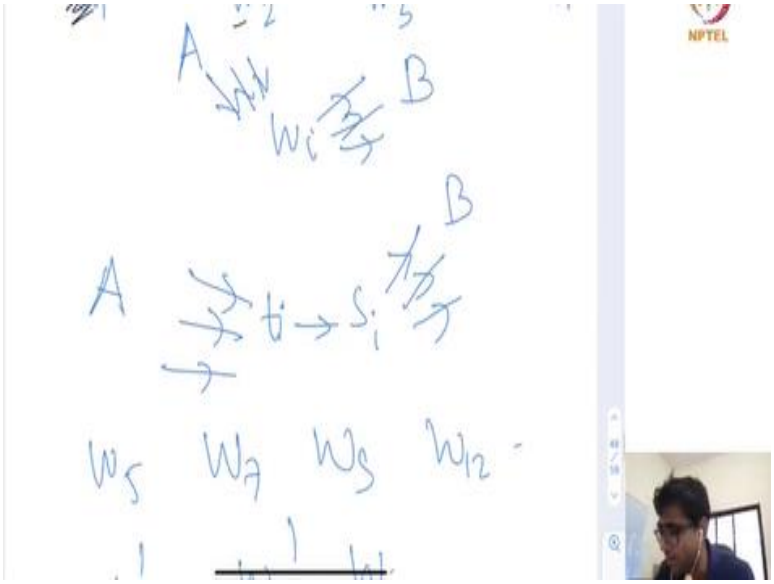
So, now, so, this is what you are given W 1 to W k + 1. So, we will talk about this splitting later, but we first I said look. If there is a solution of size at most k then so then what I mean by this.

(Refer Slide Time: 40:56)



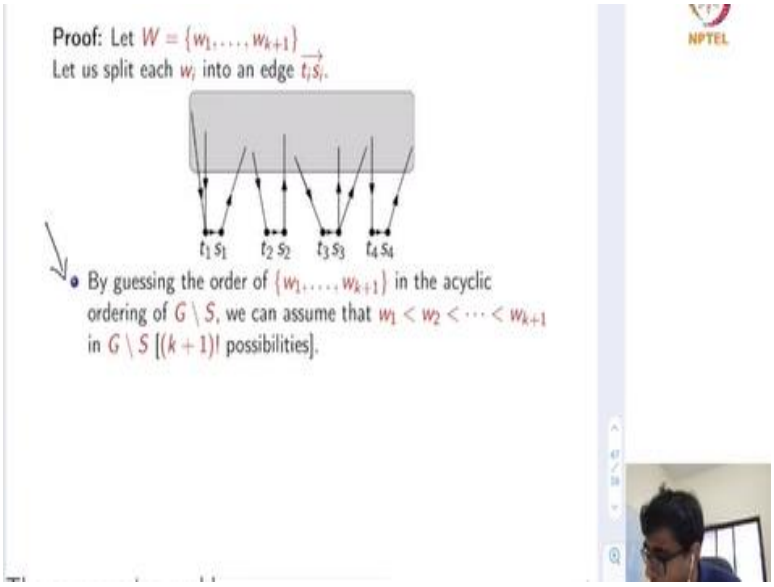
So, we guess the permutation in G - S then I know that it is a permutation. I cannot guess all the vertices but I can guess how the vertices of w are ordered on then, so this is like a k + 1 factorial. Now, once I have achieved this, I guessed may be some vertices are here, I do not know what is vertices here only thing which I am guessing is w 1 comes before w 2, w 3, w k + 1, look if the vertices w are named w 1 to w k + 1 it may not be that this is the right permutation.

(Refer Slide Time: 41:37)



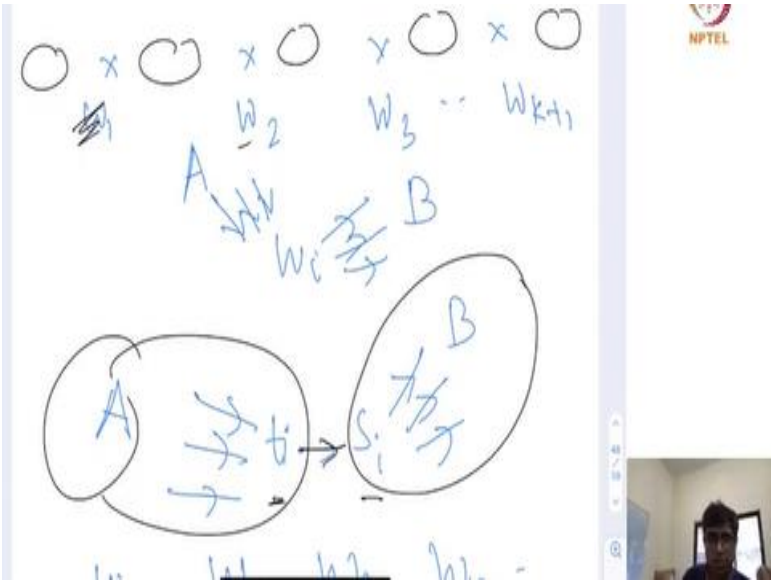
But I mean, you can assume that suppose the permutation is w 5, w 7, w 9, then let us call it the written w 5 as the first vertex w 7 is the second vertex so on and so far. Once you have done this you do the following trigger.

(Refer Slide time: 41:50)



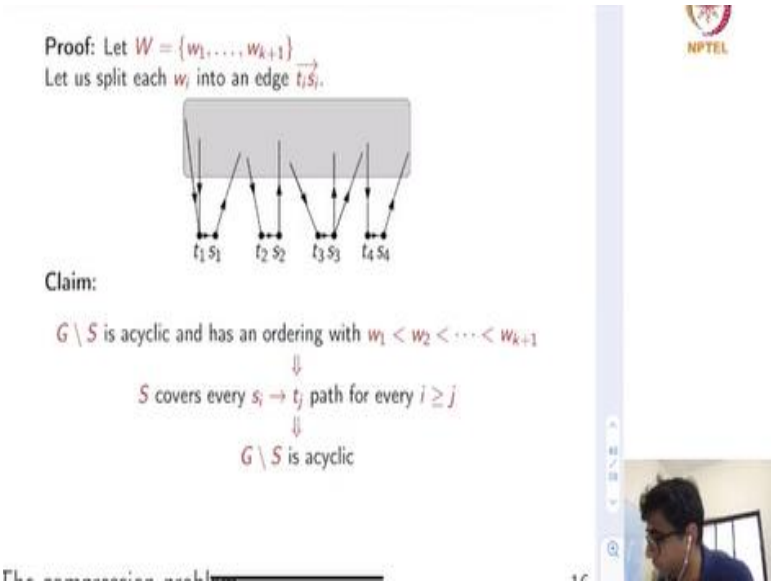
Off splitting the way when we give me our reduction some time back.

(Refer Slide Time: 41:55)



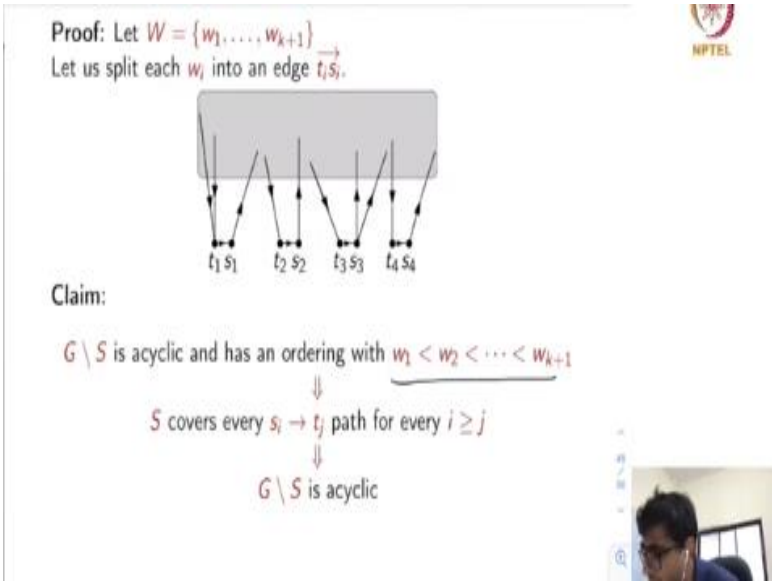
So, what do you this? Suppose W i have in arcs from A and B then you make, t i and s i given arc from t i to s i make A adjacent to t i, make B adjacent to s i. What and why are we doing this? We are trying to move towards giving or reduction to our skew multi cut you will see.

(Refer Slide Time: 42:18)



So, what is our claim? Our claim is G - S is our acyclic, and had an ordering with w 1, w that implies s covers every s i, t j path for every. So, if G - S is acyclic and this particular thing then s must intersect. So, let us see and suppose I deleted S G - S into ordering and this is this. Then why does S covers every t i t j is a path let us see.

(Refer Slide Time: 43:01)

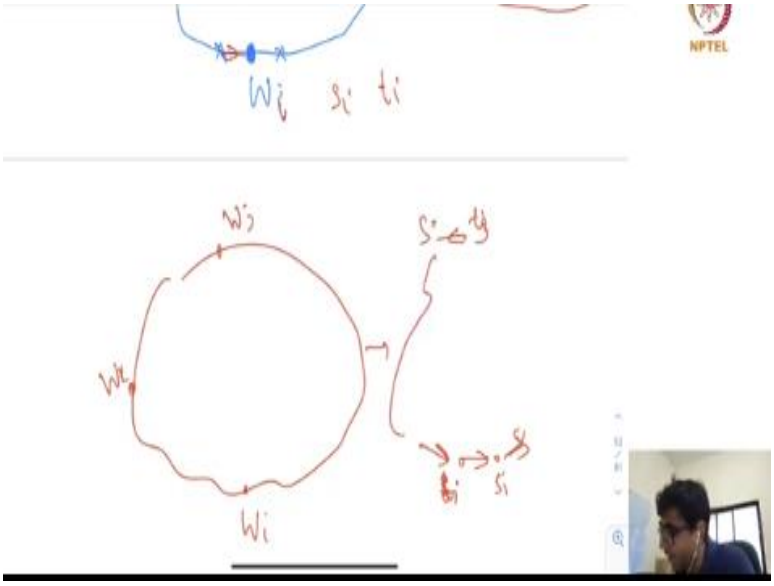


Look at G - S is a cycling and so a cyclic and so look at G- S again a cyclic, so it has an ordering of vertices, of all the vertices. Not only this and W 1 is here W 2 is here W like which is like this come. Now I am claiming to you there is no path from s i to t j for every i greater than equal to j, what is the meaning that I have a path from that? So, think of this where this WI, I can write it, I can replace this WI with t 1, s 1.

And you will notice that this that property is still holds t 2 is 2 so on and so forth. So, you replace W T by their t 2, s 2 and make every hour coming to t 2. Now, what is the meaning? So, and look still this is it. So, what is the meaning that I have a path from some s i to t j, where j is less than equal to 1 let us try to ask ourself. It means I start from here and say t j is somewhere here, I can only go forward I do not have backwards arcs.

So, definitely s has killed all the s i, t j path that is it very simple, because there are no backward arc. So, this is very simple and now suppose you are given this instance of s i, t i and s covers every s i t j a path for every i get in the continuum then i claim to you G - S is a psi and that is also very, very easy why this is very easy?

(Refer Slide Time: 44:55)

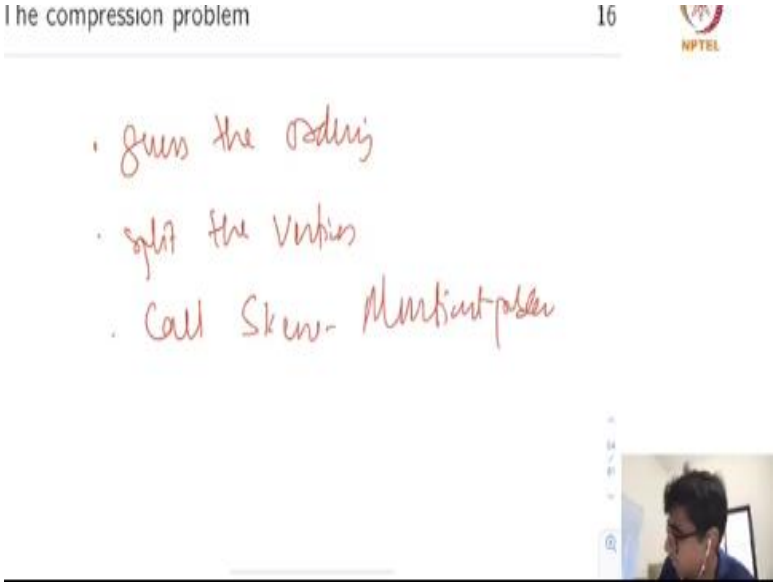


Because, if j minus s is not acyclic then fix some W then, but remember w is a directed feedback vertex n. So, if it is not a cyclic there is a cycle and that contains a vertex W i, if it contains a vertex W i then you replace W i with some s i, t i here, and then you go along and wherever you see so what I mean to say is that, what do you mean by this? So, look at this. So, maybe here the cycle and suppose, this is W i W j.

So, you replace W i with like suppose this is how it is then you know, you replace W i with t i, s i. Similarly, W j by t j, s j and you get a circular this. But it also a path from s i to t i and this is what you had assumed, this is what you have assumed that there is no power from s i to t j for

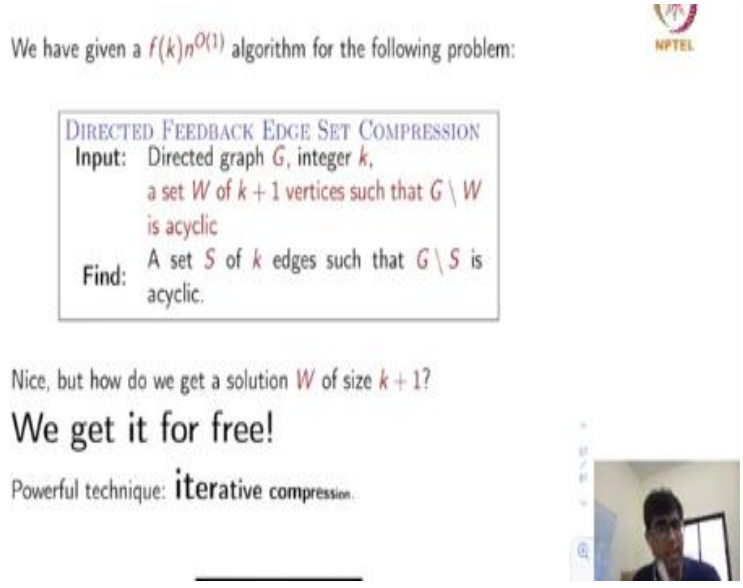
every which implies G - S acyclic. So, that is it, so we have shown an equivalent so what is our algorithm our algorithm is going to be very, very simple.

(Refer Slide Time: 46:20)



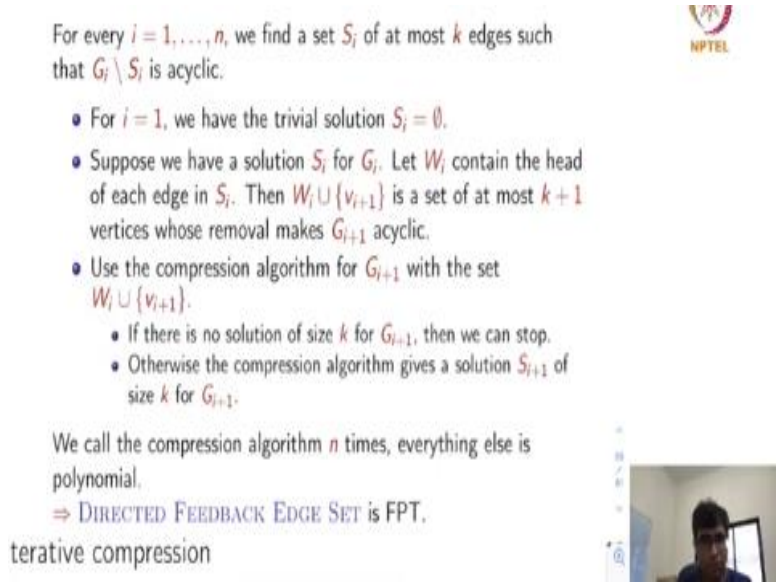
Guess the ordering, split the vertices, and then call skew multicut property instance. So, we can solve the compression problem by k + 1 factorial application of skew multicut. So, you guess the permutation is split this, I say now this is my skew multicut, and then you are done.

(Refer Slide Time: 46:51)



But as I told you so we have given an f of K poly n algorithm for the following compression. First of k + 1 vertices G - w, how did you get such. So, now we will get it for free. I try to compression we have seen it several times, but let us just do it once more.

(Refer Slide Time: 47:10)

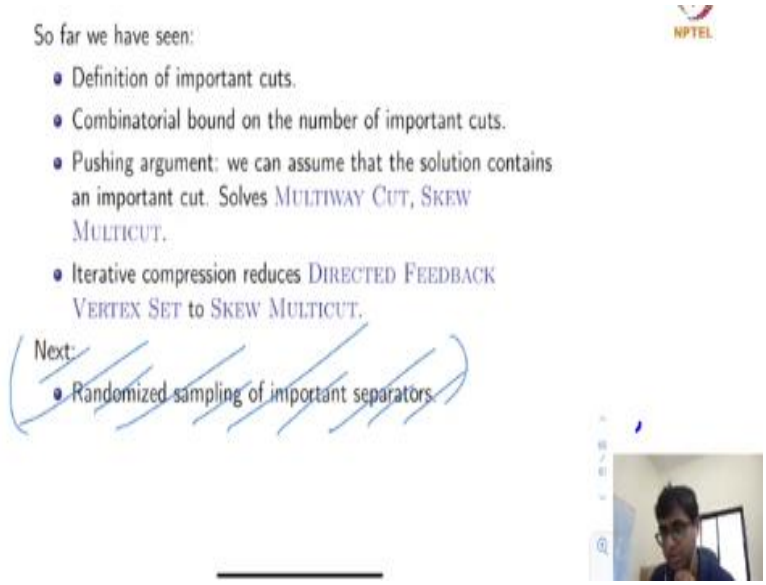


So, now we will let we want to be the editor G and the GI beta sub graph induced by first. So, what I am going to do for every i equal to 1 to n we find. Is a G minus SIS, I click and then do as follows. For I equal to 1 we have the previous solution SI equal to 5 suppose, we have a solution SI for GI and they WI contain the head of each other inside. Then clearly w 1 union. V i plus 1 is a set of at most k plus 1 vertices, which removal makes G. i + 1 is cycle.

Use the compression algorithm for d i + 1 with the set. W i union V i plus 1. If there is no, such k for G + I size at most keyword done otherwise the compression of them returns a solution s i + 1 or size, k but g i + 1 and you repeat the algorithms n times and everything else is polynomial. So, directed feedback is set is FPT using hidrotic operation. So, you saw these started with a multi-way cut algorithm we try to generalize it we could not generalize it to either undirected multi cut or directed multi-way cut or to direct advantage.

And what we were able to find a find or another something called skew multi cut and we were able to generalize that to a generalize our algorithm to that. And once we were able to do that we were able to show that one of the most important open problem; in the parameterized complexity can be FPT using this algorithm.

(Refer Slide Time: 48:40)



So, what we have seen in this week is definition of important cuts. Will bounce and the number of important cuts pushing arguments for that we can assume the solution. And decode and then we use iterative compression to reduce directed feedback vertex set or head set to screw multi cut. So, there is another technique called randomization of importance operators, which is there in the book, but we will not be covering. So, I request you to or maybe next version of this course if ever we do. So, I think with that we will end this week lectures.