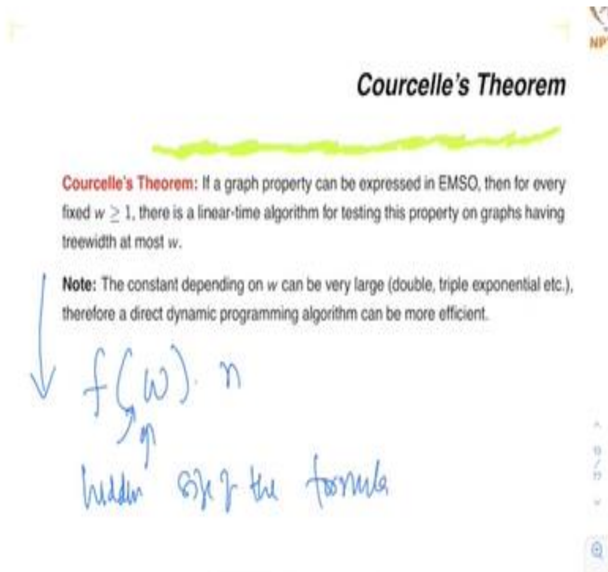
Parameterized Algorithms
Prof. Neeldhara Misra
Prof. Saket Saurabh
The Institute of Mathematical Science
Indian Institute of Technology, Gandhinagar

Lecture - 30
FPT Approximation Algorithm for Computing Tree Decomposition and Applications - Part II

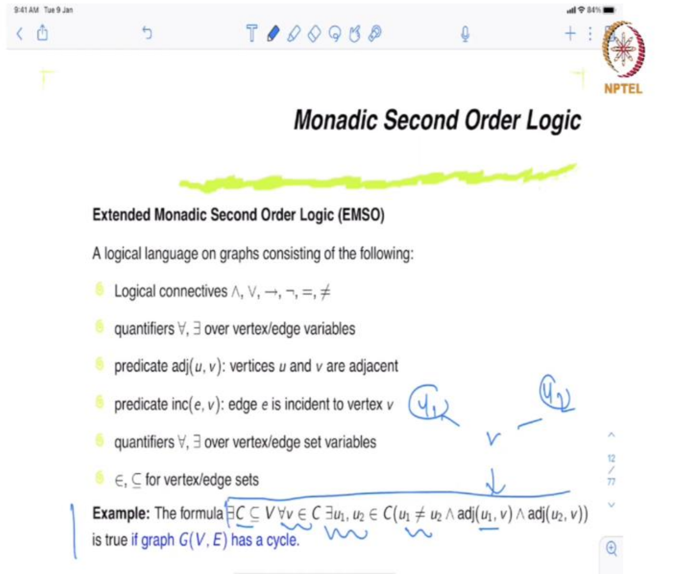
(Refer Slide Time: 00:16)



So, what is Courcelle's theorem proof? If a graph property can be expressed in this logic. Then for every fix w greater than equal to one, there is a linear time algorithm for testing this property on graphs having treewidth at most w. But here it is like f of w times n but this f of w is like very bad expression like it could be double triple exponential like the one which directly comes from this course theorem could be used as a classification theorem.

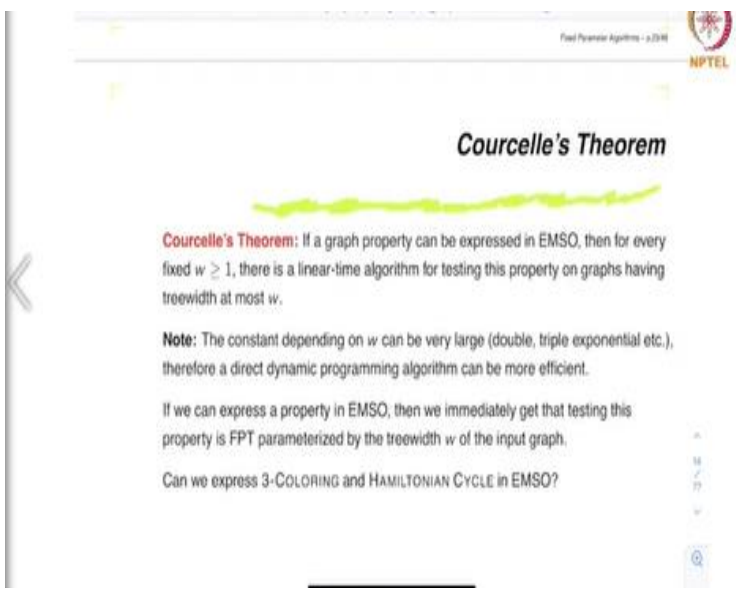
In the sense that it will tell us that this definitely means that the problem is fixed parameter tractable parameterized tree width. So, now that we know that fact, we will try to apply a direct dynamic programming algorithm to design algorithm, which could be way more efficient. So, there is something hidden here, so what is hidden? Hidden is size of the formula.

(Refer Slide Time: 01:22)



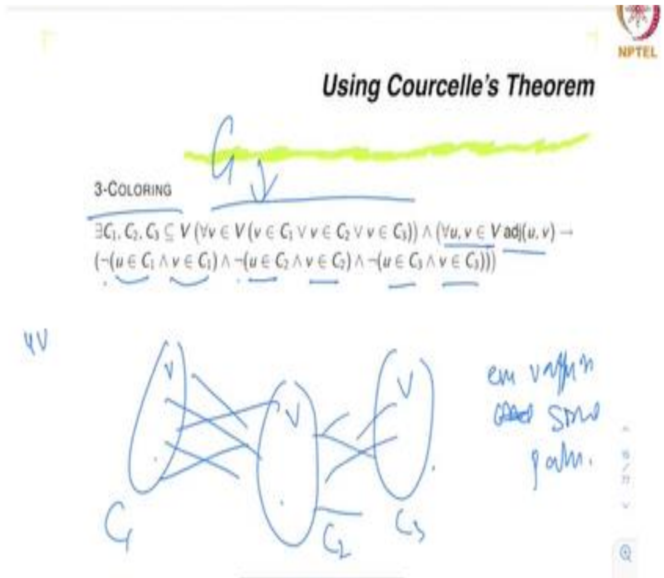
So, now look at this formula, what is the size of this formula? So, this is like a constant length formula. Because you just like want like you just count the symbols which you use number of symbol that is it. So, if the formula has a constant length, if the formula has a constant length, then this is like hidden so you do not care.

(Refer Slide Time: 01:44)



So, what does it mean? If we can express a property in EMSO, then we immediately get that testing this property in the FPT parameter by treewidth. So, now we can ask ourselves a problem, can we express 3 colouring and Hamiltonian cycle in this?

(Refer Slide Time: 02:03)

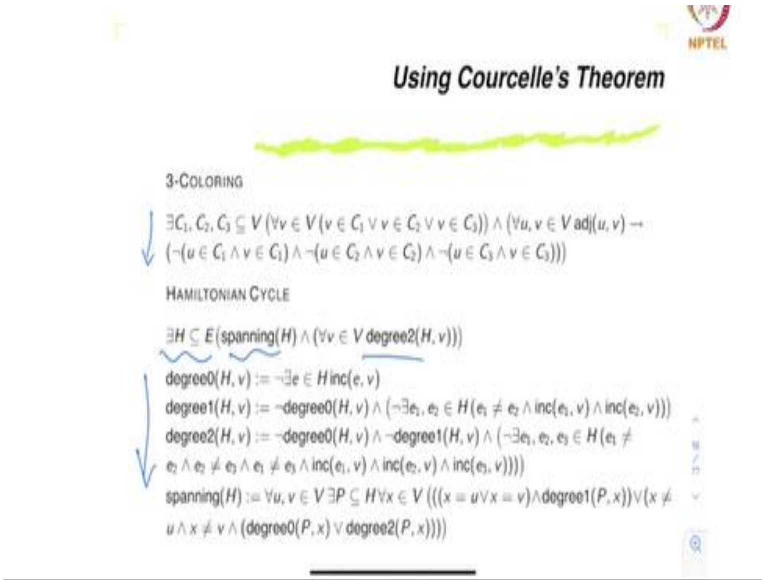


So, look at the 3 colouring, so what is the 3 colouring. So, whatever you think that exists, C 1 C 2 C 3. So, this is basically says that here is your C 1 C 2 C 3 it is a subset of V. And now what are we saying? For every V in V so, V either belongs to C 1 or V belongs to C 2 or V belongs to C 3. So, every vertex so this tells us that the first three things; tell us the existence of this subset. Third thing tells us every V appears in some partition may be more than 1.

And what will you know? For all V in V for all V in V adjacent of u, v look, so now look at any u, v suppose they are adjacent means there is an edge. Then what I want to say? Then both u and v do not appear together. And how do I show this? u in and belong to C 1 and v belongs to C 1 does not happens, u belongs to C 2 and v belongs to C 2 does not happen u belongs to C 2 and v does not belong to C 2 does not happen.

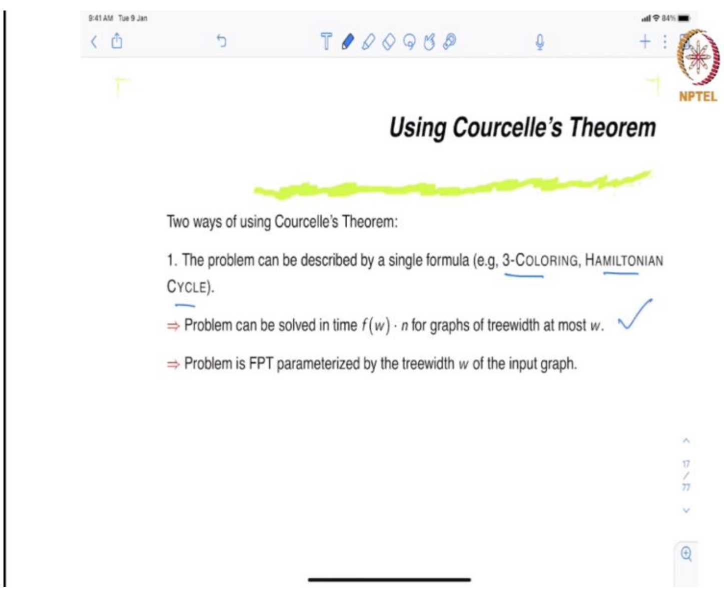
Which implies that any graph G which will satisfy this; property has a property that it is 3 colourable. Because what so we can find a three partition every vertex appear in seven it appears in one of the parts and each edge is going across. So, that is it.

(Refer Slide Time: 03:35)



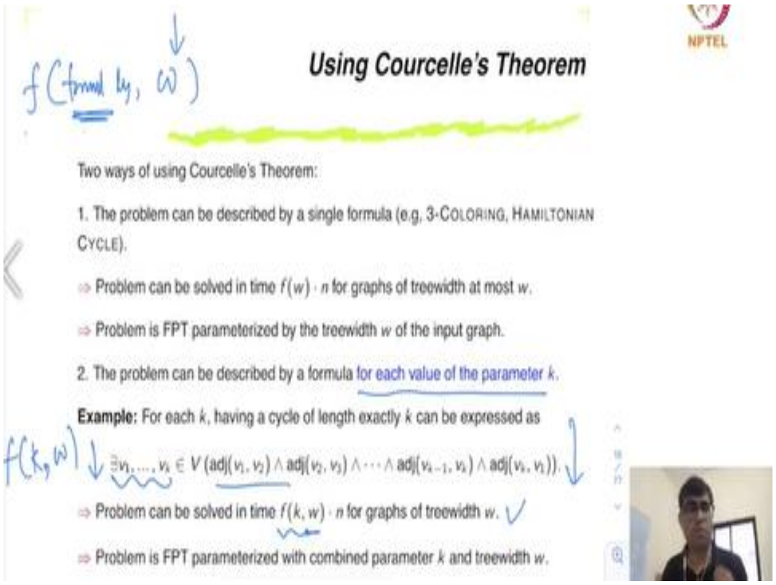
What about Hamiltonian cycle? So, I will not do it but basically all we are saying that there exists H subset of my edges, which is spanning and for all v in V degree of V in H is 2. So, I will provide this slide you can go through and this is a slightly bigger formula. But what is interesting is that all these formulas length is like is constant, like you can just count all these things which is constant which implies that all these problems like 3 colouring Hamiltonian cycles are FPT parameterized by tree width.

(Refer Slide Time: 04:21)



So, there are two ways of using Courcelle's theorem. The problem can be described by single formula example 3 colouring or Hamiltonian cycle. Our problem so for those problem can be solved in f of w times n for graphs of tree with at most w. So that implies the problem with FPT parameters by tree width of the input graph.

(Refer Slide Time: 04:42)



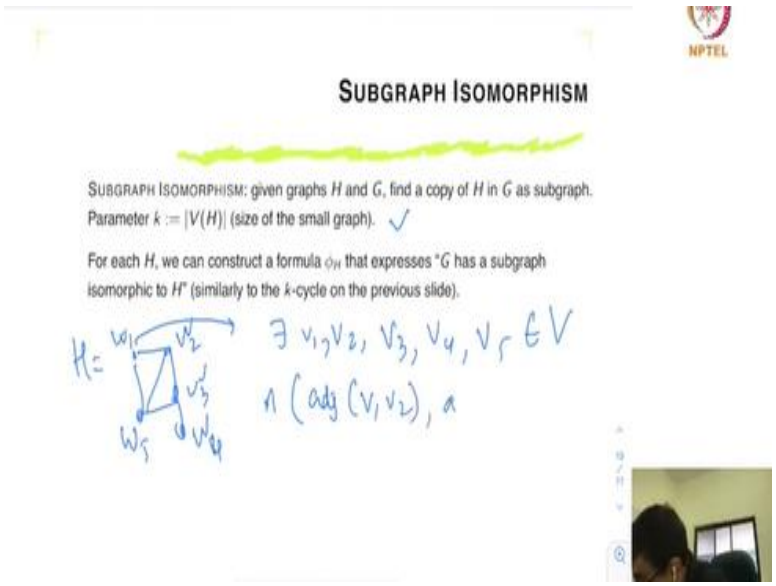
But sometime the problem can be described by a formula for each value of the parameter k. So, for example for each k having a side cycle of length exactly k can be expressed that exists v 1 to v k in V and these are adjacent to each other that is it. So, now notice that like if I am looking to find that k length exact cycle then this problem is FPT parameterized by f k the length of the formula, w. So, then the problem is FPT parameter like FPT running in time f k, w times n for a graph of treewidth.

So, problem with FPT; parameterized by combine parameter k and tribute w. So, there are several ways of using this, so sometimes in a problem the kind of problems we will generally people deal with. So, they are able to at least get formulas of this kind and secondly, they are able to reduce the w 2 as a function of k. So, everything will imply that it is a FPT parameterized by that function of k. So, there are several ways we could use it.

And whichever way we are using it we have to say that this is FPT parameterized by this. So, basically you should always remember that the Courcelle's theorem implies the problem with FPT parameterized by formula length, tree width. So, if the formula length is constant this becomes just w, if the formula length is also important then you have to represent both formula length and w. So, this is how of course now think about your any favourite problem.

Try to write a formula for that if you are able to write formula in the language described by this. Then you know that the problem with FPT parameterization by tree width. So, this provided a uniform answer to all the questions which are FPT two parameterization by treewidth. And this was proved by core cells and this is why it is called Courcelle's theorem.

(Refer Slide Time: 06:43)

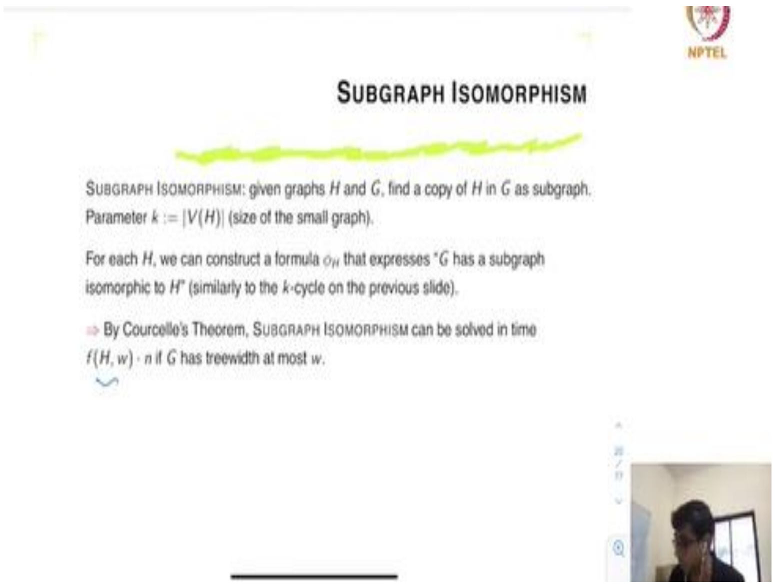


So, let us look at another example, Subgraph Isomorphism. So, you are given a graph H and G and we want to find a copy of H in G at sub graph and parameter is size of the subgraph. So, for each edge we can construct the formula phi H that expresses G has a subgraph isomorphic to H similar to k cycle in the previous. So, how we wrote the edges is we want to v k in V and v 1 and v 2 are adjacent v 2. So, for example so suppose H is this graph and this graph.

Then we will say and suppose I want to so there exists v 1 v 2 v 3. So, let us call it w 1 w 2 w 3 w 4 and w 5. So, we will say that is v 1 v 2 v 3 v 4 v 5 in V. So, we are thinking of w 1 as being v 1. So, then what is and what is this like this and what happens adjacent v 1 v 2. Now adjacent you just add you would put all the adjacent conditions and this is the way you can express that H there exists a sub graph isomorphic to H.

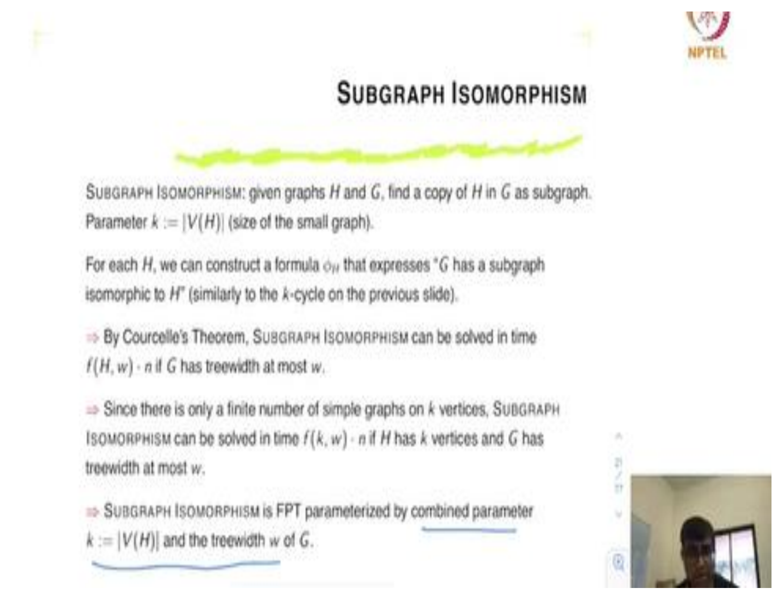
By just by writing down these expressions and for each of these guys you like think of some names in their head some symbol name. And you say look whatever the adjacency relation is you just write here, that is it. So, what does that imply?

(Refer Slide Time: 08:27)



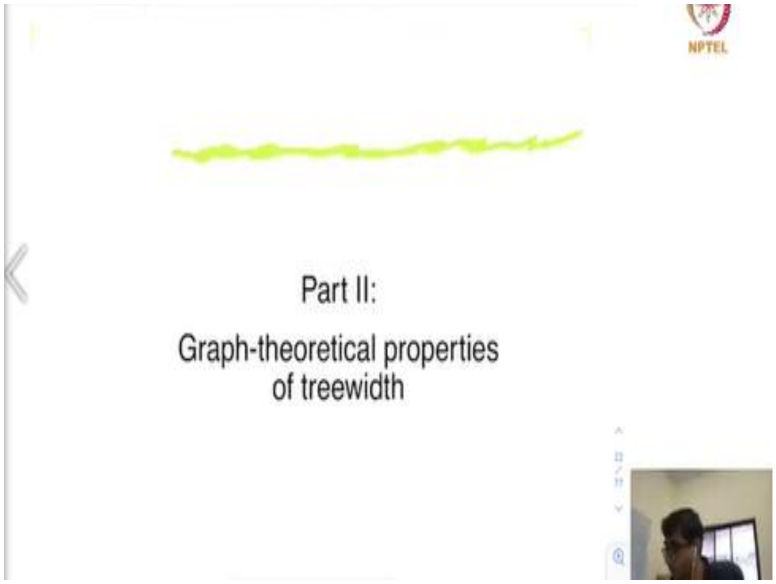
That implies that by Courcelle's theorem sub graph isomorphism can be solved in time f H, w times n if G has treewidth at most w.

(Refer Slide Time: 08:38)



And since there is only a sub finite number of subgraphs on k vertices subgraph isomorphism can be solved in time f k, w and if H if has k vertices and G has treewidth at most w. So, now you know that sub graph is a isomorphism is FPT parameterized by combined parameter k and the treewidth. Now look why we are think combine parameter K and H, because now the length of the formula here be whatever we are writing actually depends on the on k which is vertex set of H. That is the reason why we wrote down this.

(Refer Slide Time: 09:27)

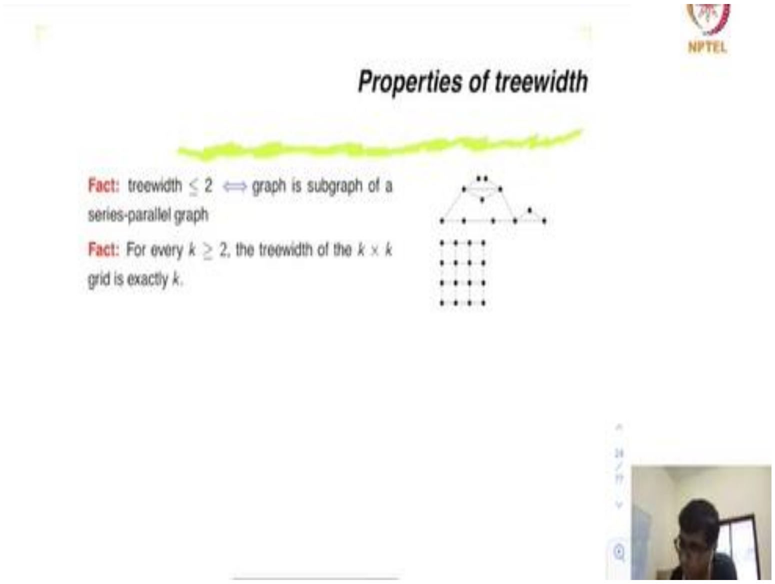


Now we have seen several graph theoretic properties of tree with we will see some more.

(Refer Slide Time: 09:32)

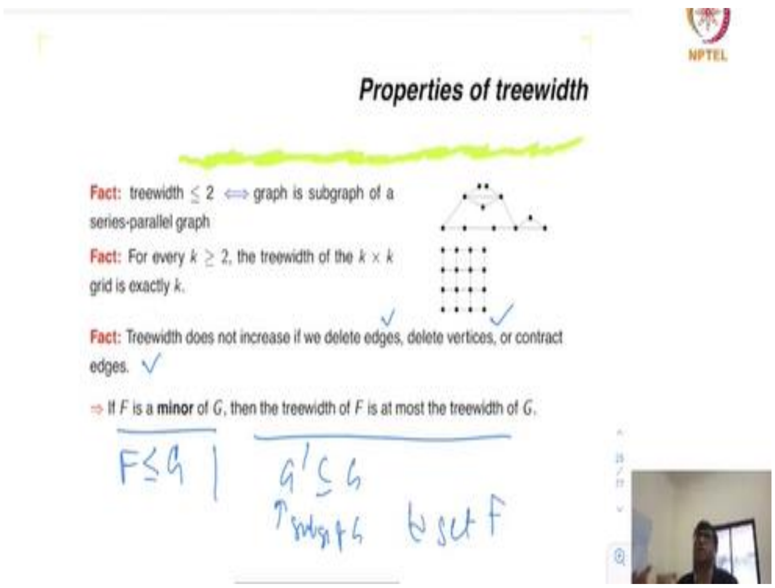
Now, so it is known that some fact so tree width at most 2 if and only if graph is a sub graph of something called series parallel graph.

(Refer Slide Time: 09:40)



And for every k greater than equal to 2 treewidth of k + k grid is exactly k. So, we have already talked about some of these things in our first week, so some more facts.

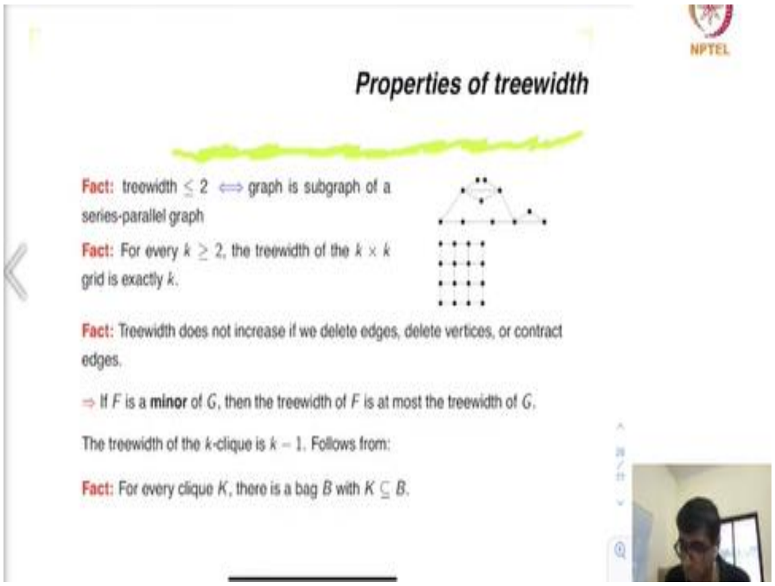
(Refer Slide Time: 09:53)



And we also know that previous does not increase if we delete edges, delete vertices or contract edges. And hence we have shown that if F is a minor of G, then the tree width of F is at most the treewidth of G. So, remember what we said we have also redefined what is an F is a minor of G, if you can get how can we get F from G? You start with G prime which is a sub graph of G which is basically means you have deleted some edges and vertices.

And now you only apply some contraction operation to get F. So, any graph that can be obtained from a particular graph G using this operation is called minor. And we have already proved that if a graph has a minor H, then the treewidth of H is at most the tree width of G. And this is why this is also called minor close properties.

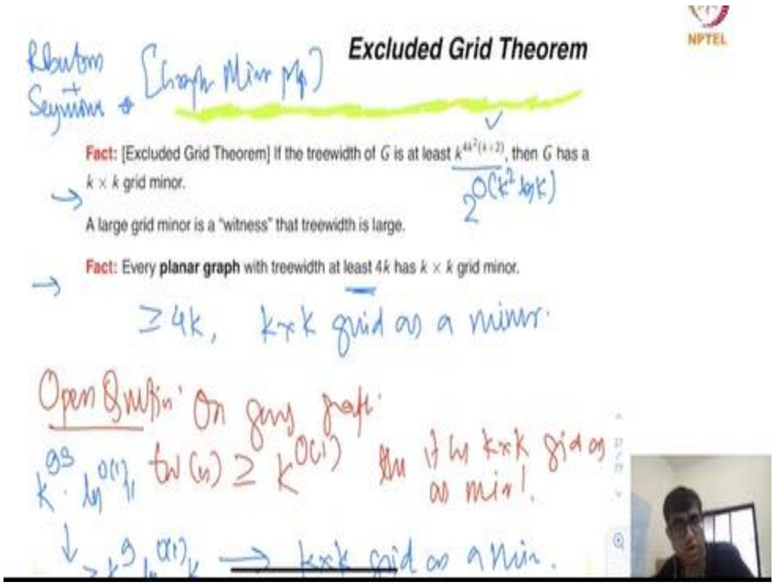
(Refer Slide Time: 10:46)



So, what does this imply? The treewidth of k clique is k -1 and it follows from the fact that for every clique k there is a bag B with K which is contained inside B. So, if a graph contains a clique of size k, then you know the treewidth has to be have at least k-1 because you will contain a bag containing all the vertices of a clique and this also, we have seen in our first lecture. So, but there is another so what do this provides? So, existence of big clique implies treewidth is B.

But there is another structure which is not as dense as say click but that existence also provides a existence of what is that called big grid.

(Refer Slide Time: 11:40)



So, what is this? So, this is the excluded grid theorem which is proved by Seymour plus Robertson and Seymour in graph minor papers. What do they plore if a treewidth is large then G has a k cross k grid as a minor. So, a large grid minor is a witness that treewidth is large. So, if you do not have a small if you do not have some big grid as a minor then what you know its treewidth is small. So, notice that this function, so this let us spend some time on this function.

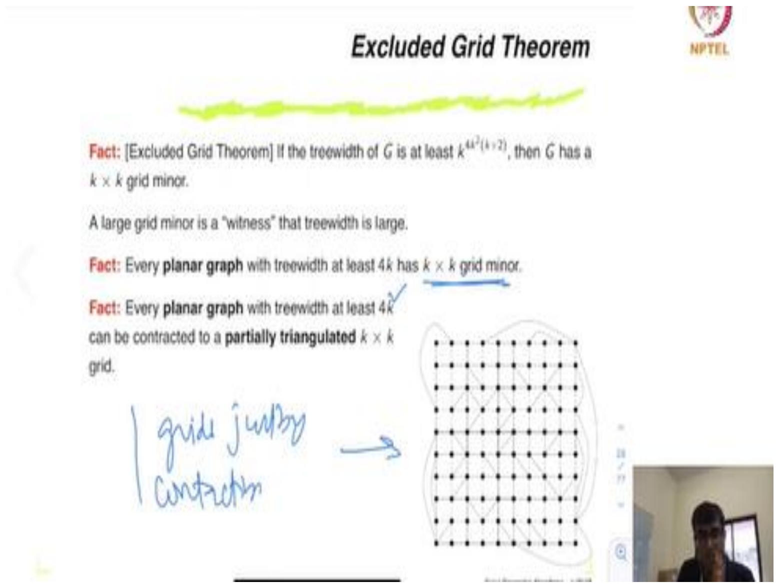
So, this basically was the function if you look 2 to the power k square log k. So, they proved that if the if the treewidth is like 2 to the power big of k square log k. Then a graph contains k times k great as a minor. But whatever further but however they proved is that if I have a planar graph then this is not true. What is if a planar graph has treewidth at least 4k then it contains k times k grid as a minor.

So, while if on a general graph existence of like you need an exponential lower bound on the treewidth to prove an existence of k times k greater than minor. On planar graph, like if the tree width is like treewidth and grade are linearly related means like. If you have a treewidth just 4 power k it has k times k grid as a minor. So, it was a big big open question in the literature, on general graph can we say that if the treewidth of G is more than some k to the power big of 1.

Then it has k times k grid as a minor. Surprisingly it was open for long time before Chandra Choukri and Julia Chuje showed it an upper bound of k to the power some 99 and some log. But after successive paper now it has been shown that the treewidth even just k poly log in K. Then that implies existence of k times k grid as a minor. So, it is one of the very important big theorem in graph theory is that this excluded.

So, now notice if I can somehow prove that look at the graph does not have a k times k grade as a minor. What does it imply? The control positive this implies the treewidth of the planar graph is at most k, 4k. Similarly on the general graph you say there is no k times k grade as a minor which will immediately imply that the treewidth is at most k to the power 10 for example.

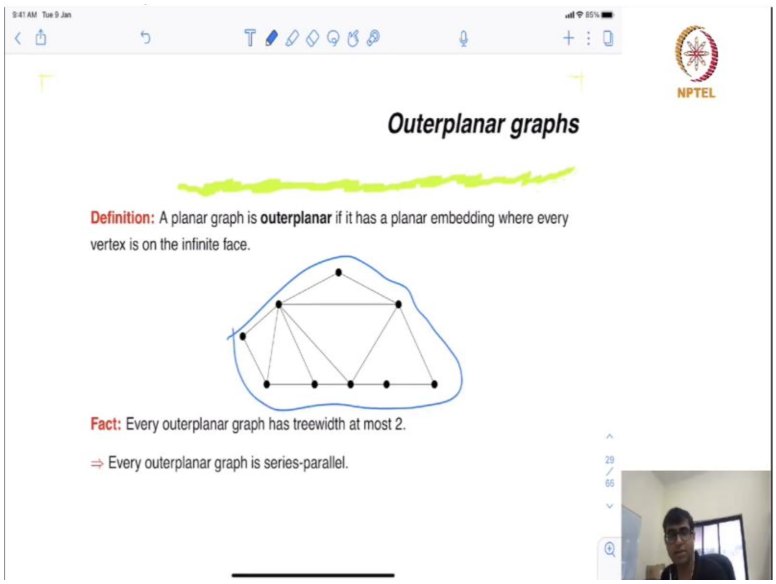
(Refer Slide Time: 14:53)



So, in fact you can prove something more that every planar graph with a treewidth at least 4k can be contracted. So, not only you can get contain k times k grid as a minor. But in fact, you can get following kind of grid just by contraction. So, look when we say that grid as a minor, what you say that? Look if a treewidth is larger than 4k then by deleting some edges and contracting I can get k times k grade does not mind.

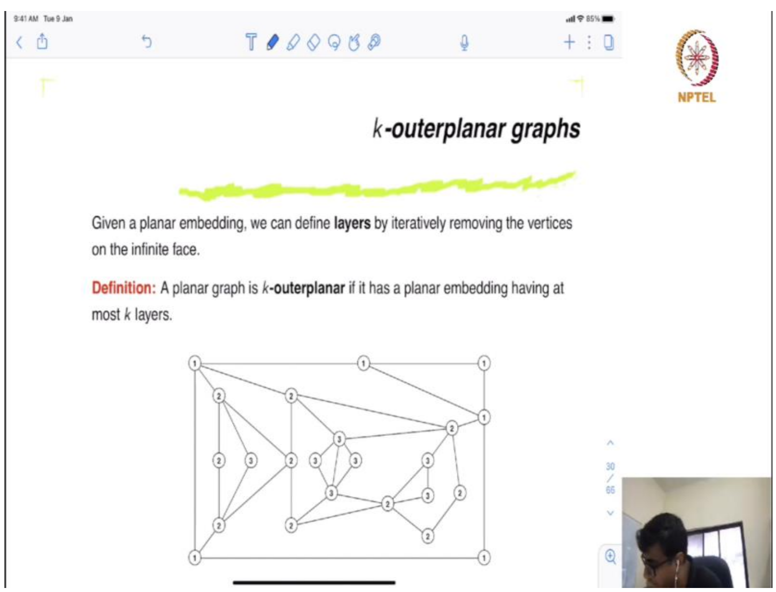
But imagine that I do not allow you to delete vertices or delete edges and only thing which are allowed is contract. Even then we can show that we may not be able to get a grid but we can get a grid like structure which you can see here and it is also called partiality triangulated k times k grid. So, not only 4k treewidth 4k implies k times k grade as a minor, in fact it also implies something which we can obtain just from contraction.

(Refer Slide Time: 16:07)



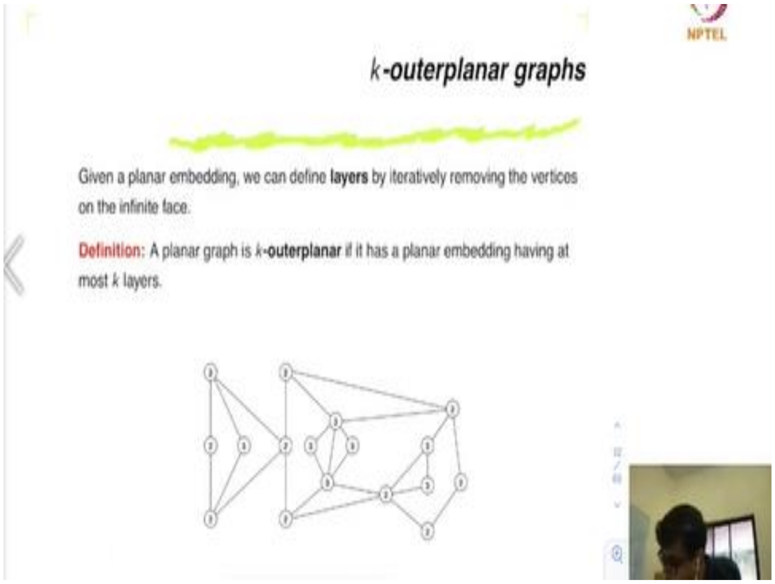
So, now we saw there is another relation of treewidth on planar graph that exists that is exploited quite a lot load. So, there is a notion of outer planar graphs, a planar graph is called outer planar. If it is a planar embedding where every vertex in the infinite phase. So, just look at this infinite phase and it is known that every planar graph has treewidth at most two of course every outer planar graph is series parallel.

(Refer Slide Time: 16:39)



So, in fact what is interesting is that you can come up with a notion of what is called k outer planar where you first a graph is called k outer planar if it is a planar embedding having at most k layers. What do you mean by at most k layers? So, look at the vertices on the top layer that is k. So, look at all the vertices which on the outer layer give them a number k that is one.

(Refer Slide Time: 17:06)



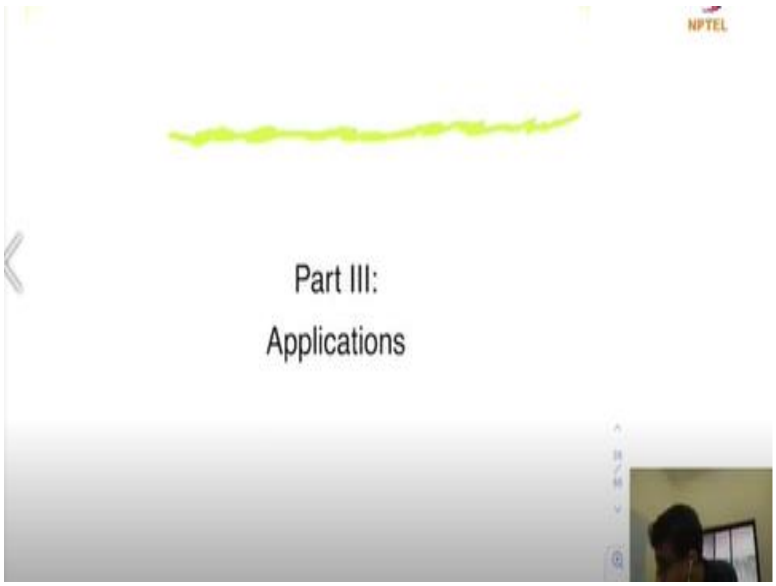
Now if you delete them what is left is some other vertices give them layer 2. So, you can what is the minimum number of layer that you can do this to achieve.

(Refer Slide Time: 17:17)

So, such that what you are left with everything is outer planar. So, the minimum number of layers you need to get to an outer planar graph is called k outer planer. So, if you need k layer so for example it is a what we had was a three outer planar graph because we could reach this. So, we had first so you delete all the vertices which on the outer planar layer or infinite layer you get after you delete them you get another set of vertices which are on infinite or which are the outer face.

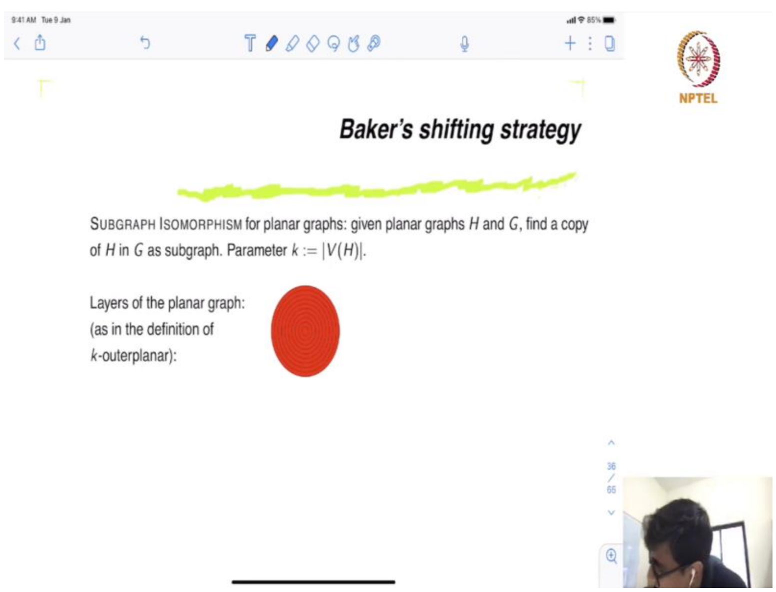
You delete them you get another set of vertices which on the outer phase, if that is the only one then number of times you have to do this is called outer planarity of the graph. And what is known is that what is known is a very nice relation between outer planar and treewidth who we will exploit it for some of the examples or which we will see in a minute.

(Refer Slide Time: 18:12)



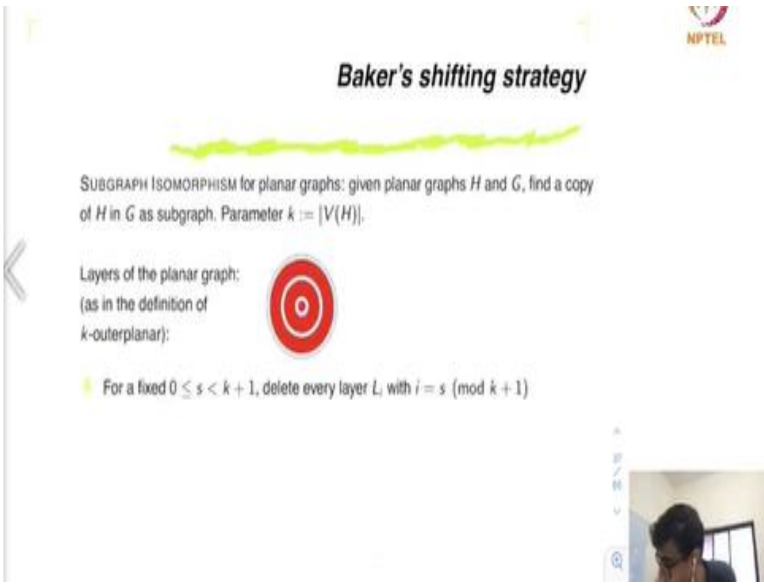
So, look at some of the applications we have.

(Refer Slide Time: 18:15)



So, now let us want to do let us try to prove that subgraph isomorphism for planar graph. So, you are given a graph H and G and parameter is k. So, look at the layers of the planar graph as in the definition of k outer planar.

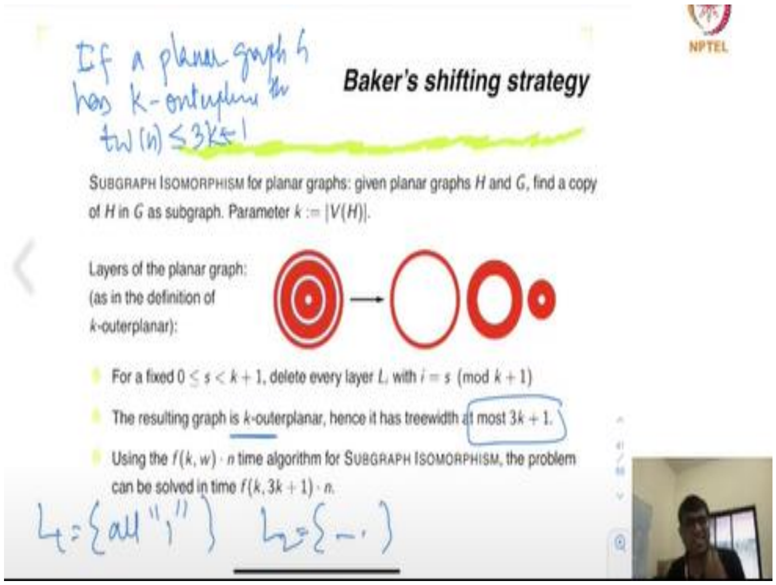
(Refer Slide Time: 18:28)



And so, you fix some integer s and 0 to k + 1, now let us delete every layer L i with i = s mod k + 1. What is the meaning of this? So, you have layer so you give them layer number 1 2 3 k +1 then again you call layer 1 2 3 k + 1 again you call 1 2 3 k +1. So, now let us delete all the layers with number 1 1 1 1. Now after you have deleted it what every component of the graph is completely contained inside this and every component has at most some k layer.

So, this is what you fix integer 0 and you delete all layer with some mod k + 1. What do you know about this? So, whatever you see that you are deleting different things.

(Refer Slide Time: 19:47)



So, you basically you delete some of these things and you will get all these pieces. Now what you know? That every component is a k outer planer and it is known that if a planar graph is k outer planar then this tree with at most 3k + 1. So, if a planar graph G has k outer planarity, then treewidth of G is at most 3k +1. Now that is great for us, why? Because why this is important? So, suppose my H look at this planar graph H it has only k vertices.

Now if it has at k vertices then by pigeon hole principle one of the layer 1 2 k + 1 does not contain its vertices, Am I right? So, if it does not contain say it does not contain either it does not contain vertices from layer one or it does not contain vertices from layer two and so on like it does not contain vertices with say. So, we what did we did 1 2 3 k + 1, 1 2 3 k + 1 and this is why we call, so we made like.

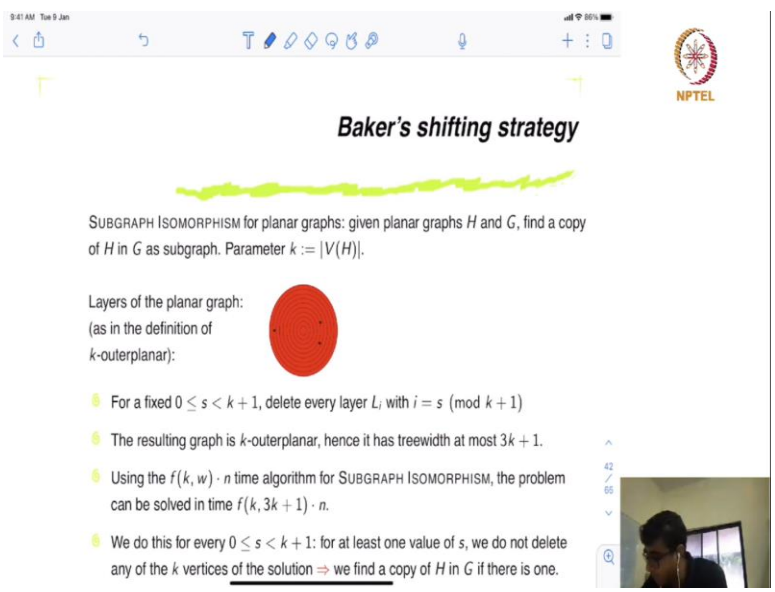
So, what was this L 1 one is like all one layers like one from each 1 2 k +1 the first one then next 1 2 k + 1 first layer. And similarly, l 2 now what do you know that because our graph only had k vertices one of the layers it does not contain any vertices. So, if we have guessed the correct layer then what you know after I have deleted that guest layer then every connected component has at most k layers and hence this treewidth is at most 3k.

So, now we can apply our f of k 3k + 1 algorithm of sub graph isomorphism and solve my problem. Now if any of the like if for any of the call of subgroup isomorphism we succeed in finding this sub graph then we are very happy. Otherwise what we know about this? We know that there is no subgraph because if there was a sub graph of this edge then at least deleting one of these layers will preserve that sub graph and we would have found this using this dynamic program.

So, this is what is called in some sense. This is a very famous strategy which is used in polynomial time approximation scheme. We will see its application in a few minutes also but you notice this is like you divided your layers into 1 2 3k +1, 1 2 3k +1. So, it is like 1 2 3k + 1 then 1 2 3 and then you say look imagine that it does not contain vertices from layer one. So, then it will contain only here.

And I know and the treewidth of a graph is basically maximum of treewidth of connected components, and each connector component has layers at most k. So, the treewidth of each connected component is at most 3k + 1which implies the treewidth of the whole graph is at most 3k + 1 and you will be able to find this pattern, I hope this is clear.

(Refer Slide Time: 23:03)



So, we do this for every s at most between this and for at least one value of s we do not delete any of the k vertical solution which will imply that we will find a copy of H in G if that is one.

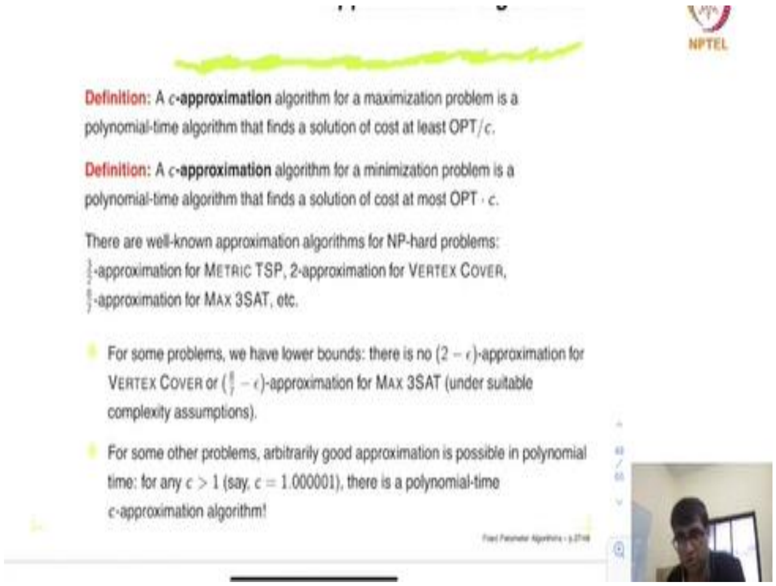
(Refer Slide Time: 23:29)

Now let us Detour to the approximation.

(Refer Slide Time: 23:32)

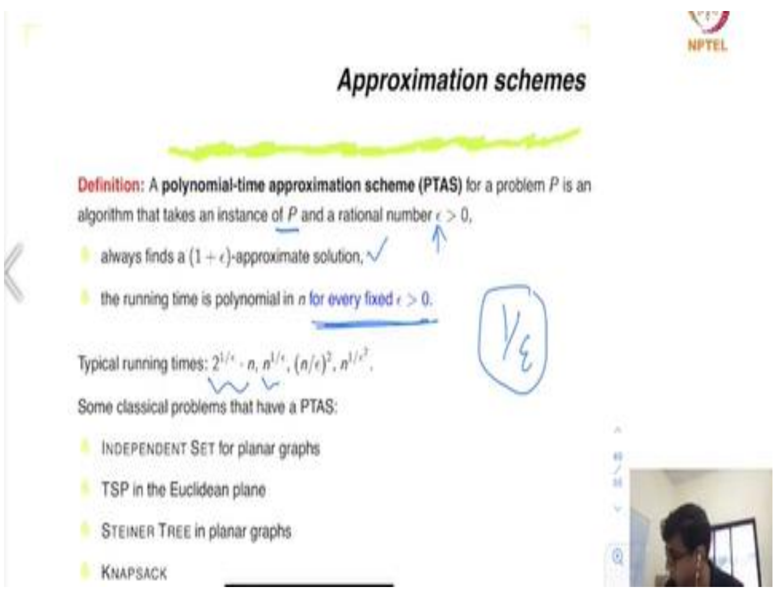
So, what is the c approximation algorithm for a maximization problem is polynomial time algorithm that finds a solution of cost at least OPT divided by c. And a c approximation of a minimization problem with a polynomial time algorithm that will find you a solution of cost at most OPT time c. And there are several approximations of NP-hard problems like for METRIC, 2-approximation for VERTEX COVER, Max 3SAT 8 by 7 approximation metric TSP 3 by 2 approximation and so on and so forth.

(Refer Slide Time: 24:03)



For some problems we have a lower bound that is there is no two minus epsilon approximations for vertex cover or 8 by 7 minus epsilon approximation from max 3SAT of course under suitable complexity assumption. For some other problems arbitrarily, good approximation is possible so for any c greater than equal to 1 say c 1.001 there is a polynomial time approximation of polynomial time c approximation algorithm.

(Refer Slide Time: 24:29)

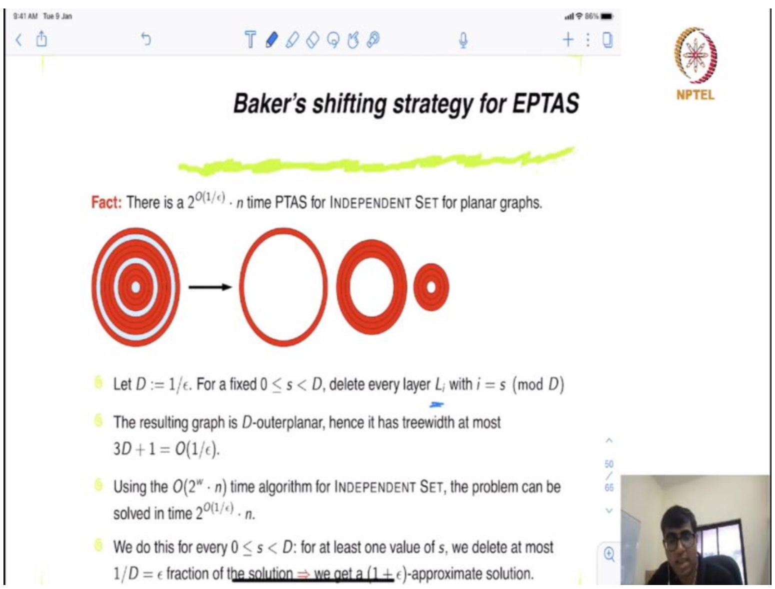


And the notion of what is called polynomial time approximation scheme. So, what happens in polynomial time approximation scheme for a problem P? So, it takes an instance of problem P and it gives takes a parameter epsilon and it will always find you a one plus epsilon approximate solution but the running time is polynomial in n for every fixed epsilon greater than equal to 0. So, what could be the running time?

The running time could be 2 to the power 1 over epsilon times n or maybe I mean some n to the power 1 over epsilon or maybe n over epsilon to the power whole square and n to the power of 1 up 1 by epsilon square. So, if you think of this is like a we are trying to think of in beta as whether it is an FPT or XP algorithm in the parameter one over epsilon. So, you can do all kind of things here.

Some classical problems that have PTAS like independent set for planar graphs, TSP in the Euclidean planes, Strainer Tree in planar graph Knapsack and so on and so forth.

(Refer Slide Time: 25:32)



So, let us look at this Bakers shifting strategy for EPTAS. So, there is a 2 to the power one number epsilon n times PTAS for independent set for planar graphs. So, you set D = 1 over epsilon and now as we did it for this sub graph isomorphism, you look at your planar graph and you draw them outer planarity layer. And as always what is this L i those i who leaves s when you divide by D. So, all those layers who's like mod D.

So, now so the result so now if you delete any layer, what is the property of this graph? It is a D outer planar and hence its treewidth is at most 3D + 1 order. So, using the time 2 to the power W times n the existing independence for independent set, we can solve this problem in time 2 to the power big of 1 over epsilon times n. So, what we did? So, based on one over epsilon we fixed a number D and we made we deleted some layers like.

And on each we solve the problem and what are we going to output. So, we do this for every s between 0 and 2 and we will output the layer with respect to which the weight of the independent set is maximized. What do you know about? So, look at the Pigeonhole Principle What is pigeonhole principle tells us? That look this is some layer this is some layer this is some layer this is and these layers are disjoined.

Now your independent set vertices are distributed among them. Now when I say layers this is like some set of layers L 1 L 2 L s or L d. Now look at the independent side vertices but pigeonhole principle at least one layer will contain at most one over D fraction of this. So, at most you would have returned suppose the solution had size weight W. So, what will you return? W minus at least solution of size you will return W minus epsilon.

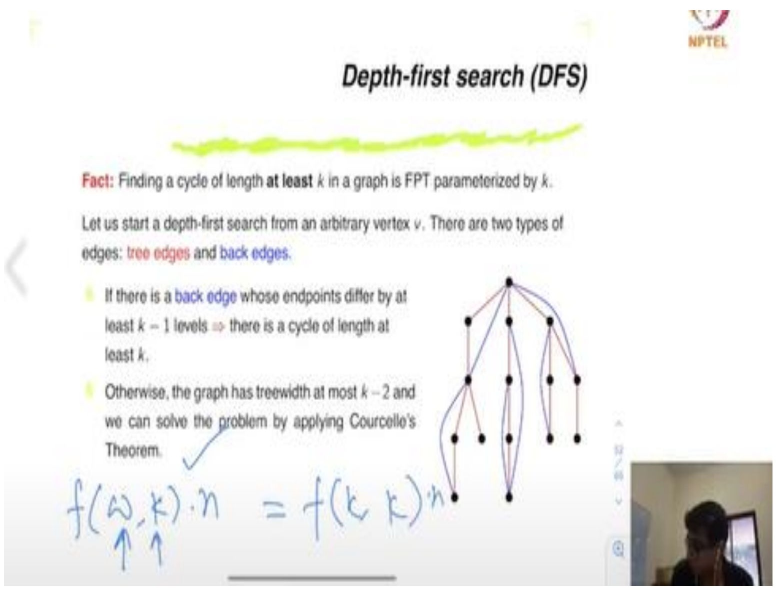
So, if you do this by pigeonhole principle you will be able to show that you only lose one but one over epsilon fraction or not even one, one over epsilon fraction of the total weight and you are able and that if you do maths, you will get 1 + epsilon approximation solution. So, I mean I leave this math to you, but the idea is that if the total weight was W for max weight independent set, there is you will lose how much W.

There is a there is a layer which contains at most W by D. So, you will definitely be returning a solution of weight this and if you do math this will come one plus epsilon approximate solution.

(Refer Slide Time: 28:55)

So, now let us go back to FPT algorithm.

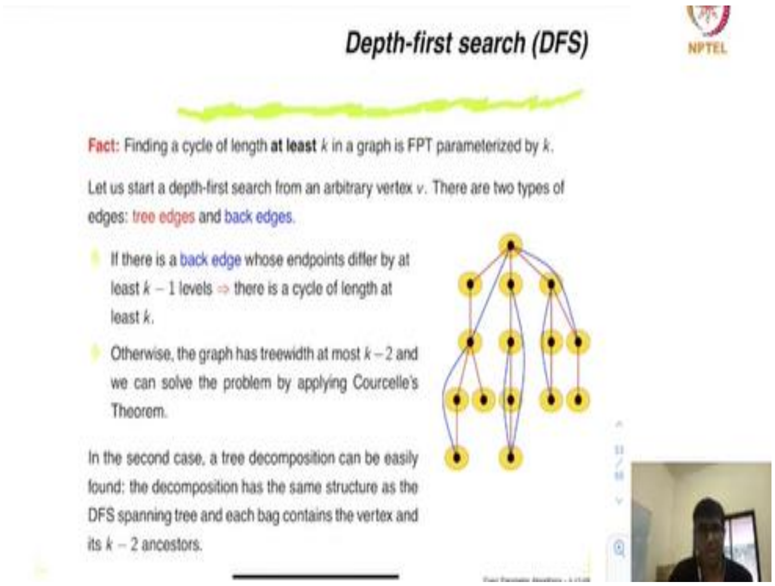
(Refer Slide Time: 28:58)



So, this we have already discussed when we were talking about the applications why we care about tree width in parameterized algorithms. So, I told you that look using this DFS either we can find a long path or long cycle or we can compute a tree decomposition. Now if we can compute our tree decomposition, we can apply again courses theorem and like. Otherwise, the graph at treewidth at most k - 2 we have already done this.

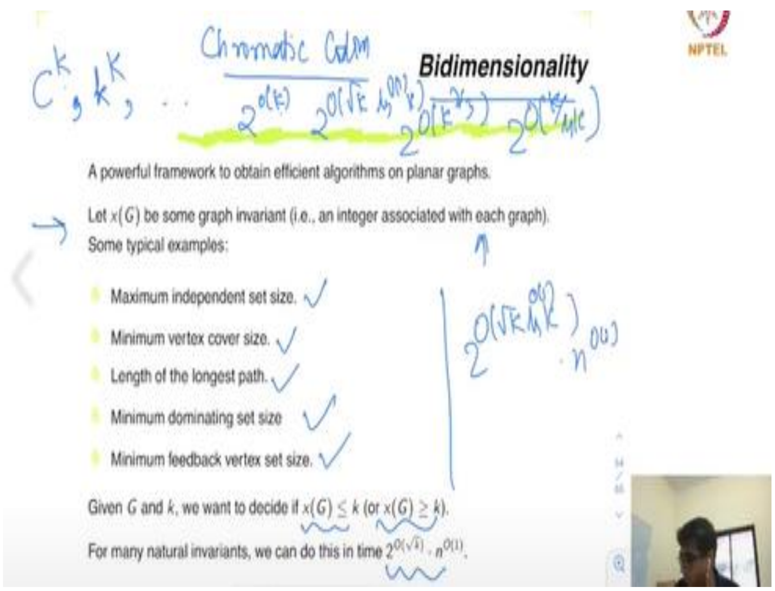
And then once the graph has boundary treewidth we did not do anything in our first big lecture. But now we know Courcelle's theorem we can write down this formula and using this we can show that finding a cycle of length at least k in a graph is FPT parameterized by k. Now because why? Because the running time will be f of w, k times n. But w is here like where k is a length of the formula which is k and w is also become k. So, you are done.

(Refer Slide Time: 30:01)



So, we have already seen this so we will I will leave this out.

(Refer Slide Time: 30:08)



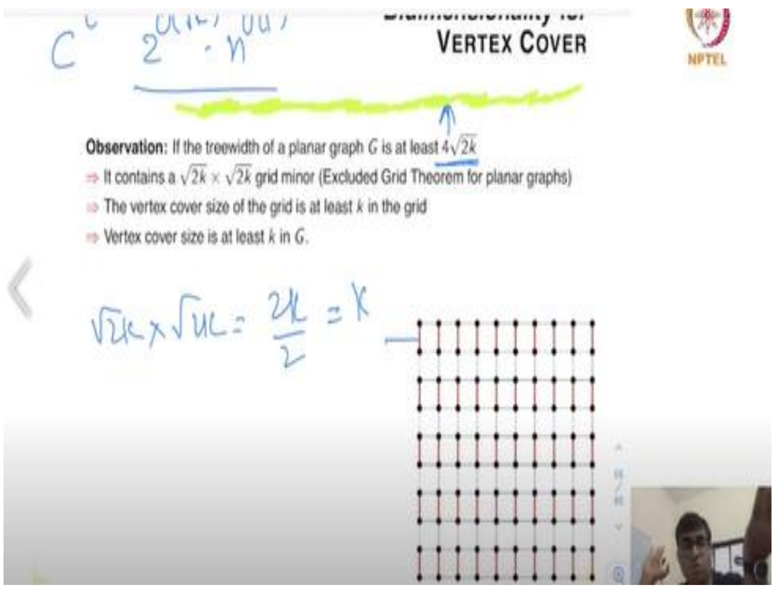
But this is the one thing which we have not talked about unless spent next 10 to 15 minutes on what is called bidimensionality. So, up until now we have been designing an algorithm with running times either C power k or k power k. But you have seen an algorithm via chromatic coding where we design an algorithm with running time 2 to the power little of k. What is the meaning of little of k is basically something like 2 to the power big of root k, log 1 over k or maybe 2 to the power big of k two third.

But not like something little of k. So, for example 2 to the power big of K by log k is also 2 to the power little of k. And it is a very powerful framework to obtain efficient algorithm on planar graphs for this. So, let x G be some graph invariant, that is an integer associated with each graph and some typical example maximum independent set size, minimum vertex cover size, length of longest path, minimum dominating set, minimum feedback vertex set.

And our question is given G and k we want to decide whether x of G is less than equal to k or x of G is greater than equal to k meaning I want to say is a vertex cover at most k is feedback vertex at most k is a dominating set at most k is longest path is the is there just a path of length at least k is the independent set of size at least k. And for many of these natural graph problems or graph in variance.

We can actually design our algorithm with running time 2 to the power big of root k polygon or if nothing then we can definitely get this and we will see how we will be able to do this.

(Refer Slide Time: 31:50)



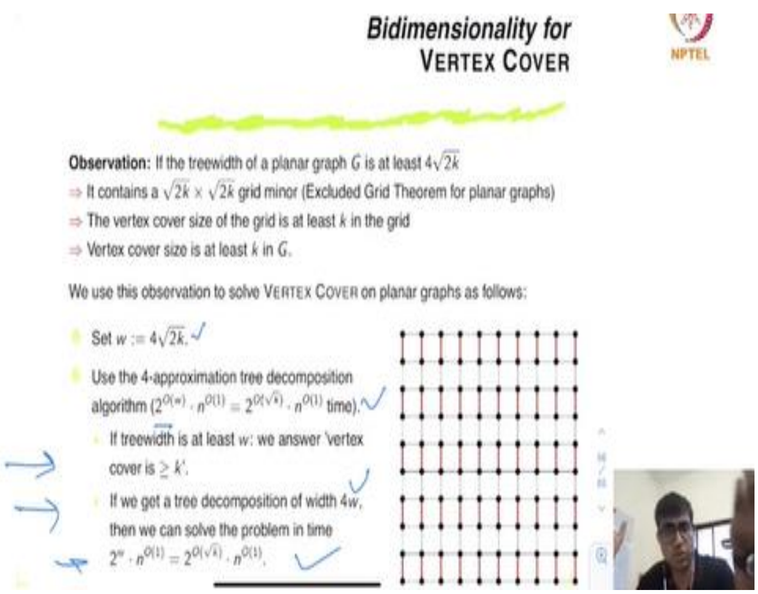
So, our observation is very simple let us look at the vertex cover, I say if the treewidth of planar graph G is at least 4 root k, fine if it is at least 4 root k and how can we check this? We already design an algorithm that given C an integer k it can it runs in time some C power t and test whether the treewidth is at least this much or not. So, for even a planar graph G and we want to test whether the tribute is at least 4 root k.

So, this we can do in 2 to the power big O of root k times some n to the power big O we can check. If the treewidth is large, what can we say? Well, it contains of course you need to be a little clearer you want to say treewidth at least 4 root 2k or something or not. So, what happens? So, it contains a root 2k times root 2k grid minor. Now if look at this is a grid as a minor, so now look at these red edges what are they, they are matching edges.

So, any vertex cover of this grid must contain at least one vertex of this matching. So, if the number of matching edges is strictly larger than k. Then what is this? So, look at this this is like you will at least need root 2k vertices from here and the root 2k times root 2k which is divided by 2 because root 2k times root 2k is like 2k divided by 2, which is like k. So, if you have a slightly larger grade then you know that this minor itself needs k vertices.

So, of course the whole original graph needs more than k vertices 2 in the vertex cover. So, you can say no.

(Refer Slide Time: 33:54)

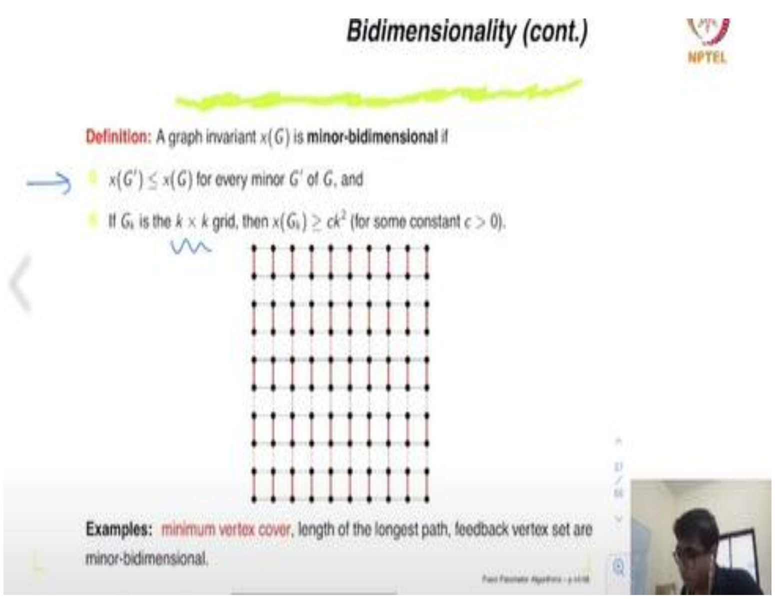


So, or the vertex cover of size grid is at least k times k at least k in the grid which implies vertex for size. At least so how do we use this observation to design, you set w = 4 root k. We use the four-approximation tree decomposition four approximate tree decomposition 2 to the power big of w. So, it will run in this line, so if treewidth is at least w we answer vertex cover is greater than equal to k, done.

Otherwise, you get a tree decomposition of width 4 double o then you solve your problem in time 2 to the power big W so you are done. So, basically what is this problem all about like I check the treewidth this graph. If the treewidth is large I say no there is no vertex or we have found a small tree decomposition now I apply my dynamic programming algorithm and we have solved the problem.

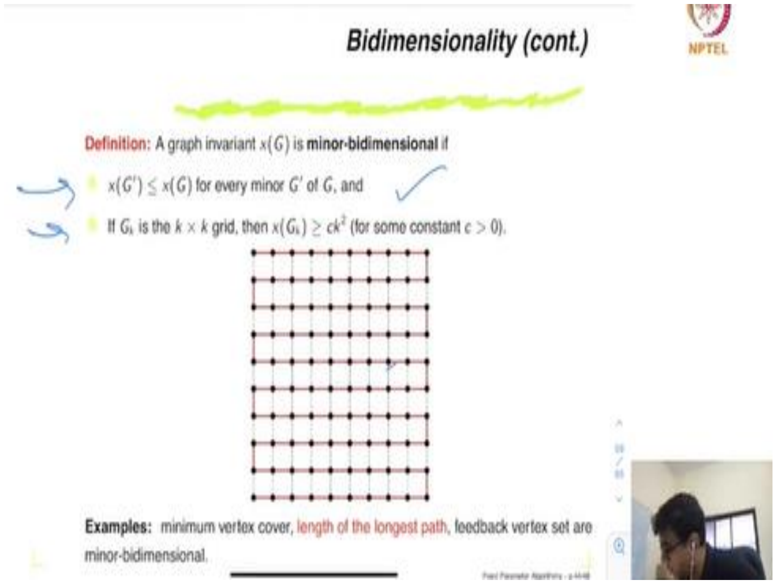
But this is nothing about vertex cover that we use, only think we use is that o. What is the solution size on the grid?

(Refer Slide Time: 34:51)



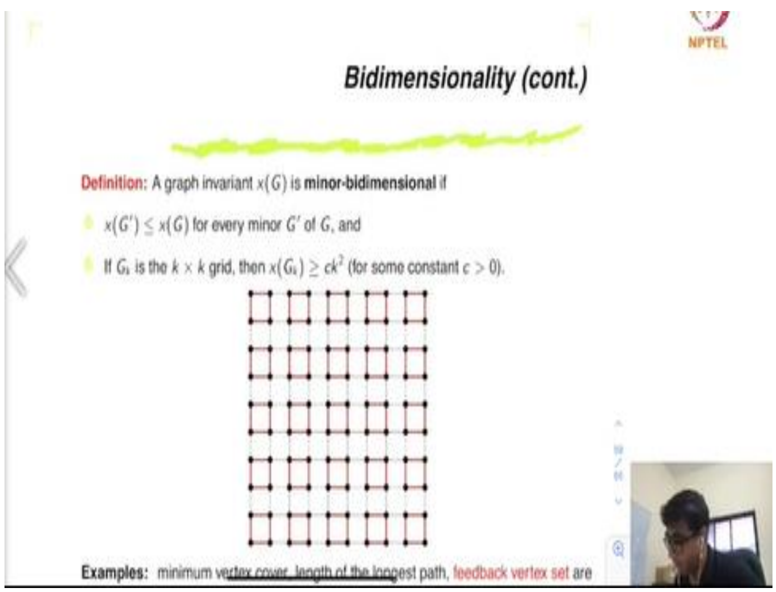
So, we that defines a notion of what is called minor bidimensional if what is a minor bidimensional? So, the parameter like the solution size does not increase on the minor and if you have a times k grid then the solution is at least C times k square. We already saw for vertex cover, what about feedback vertex set? For feedback vertex set so this is for the vertex cover.

(Refer Slide Time: 35:22)



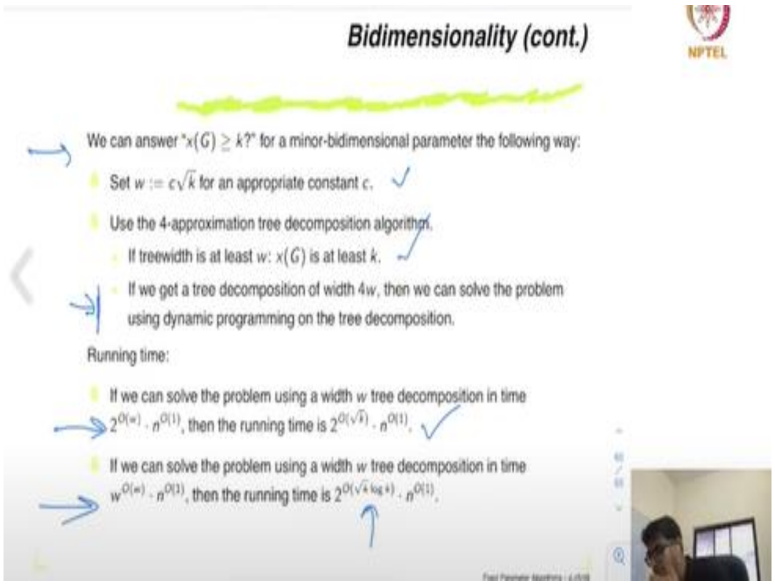
Now look at the longest path, so look at this this is like a big big path. So, if you have a more than some, so it means if you have a k times k grade then the length of the longest path is of order k square, the problem is bidimensional. And on minor and length of the longest path cannot increase like of the on the minors. So, it is minor bidimensional and it is like step first property holds. And secondly solution size on the grid is like order k square so this is a minor by dimensional property.

(Refer Slide Time: 36:00)



What about feedback vertex set? Again, feedback vertex set does not increase on minors and if you look at k times k grid then you can get these vertex joint cycles of like order k square vertex the joint cycle which implies you at least need order k square vertices to intersect all this cycle which implies feedback vertex set is minor bidimensional.

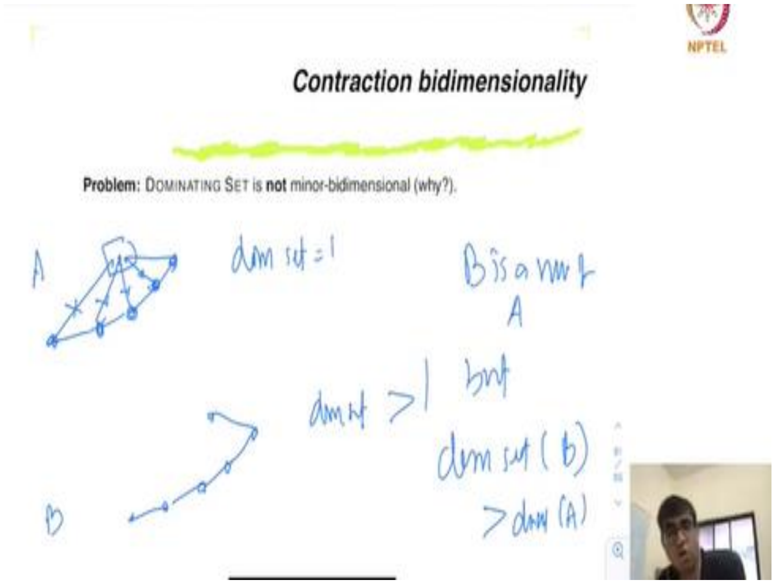
(Refer Slide Time: 36:23)



So, then what does this tells us so for minor bi-dimensional this is the algorithm we can get. So, we can answer x G greater than equal to k for minor by following way. You set appropriate c root k, as w you apply a factor four approximation and if the tree width is large, you say answer you know that x G is at least k, so for maximization problem might be you might be able to say this is any guess instant for a minimization problem you might be able to say it is a no instance.

Otherwise, you have got a small tree decomposition and then either this running time, then you will get such an algorithm if you had w to the power big of w time algorithm then you will get such kind of algorithm. So, very simple strategy on 4 planar glass, what is this? I check what whether my problem is minor bidimensional or not. Then I just need to check whether it has some appropriate c power w or w to the power w algorithm depending on that you know that it has such algorithm and that is it.

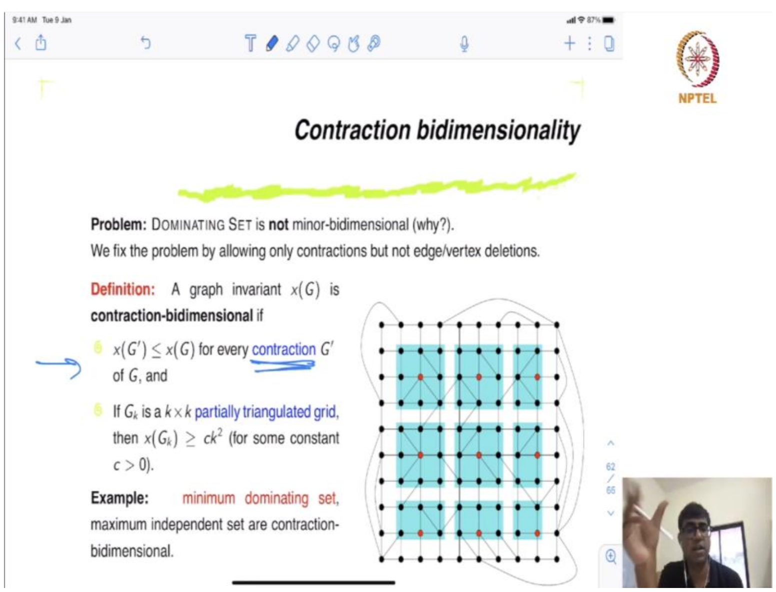
(Refer Slide Time: 37:27)



But there are problems which are not minor financially like for example dominating set. Now I will give you an example look at this, it is a dominating set of size one. Now let us delete all these edges, then what you will get? You will get a path and now you can check the dominating set is strictly more than one, one vertex cannot help you or is not sufficient. So, this is graph A this is B, B is a minor of a but dominating set of B is strictly more than dominating set.

So, dominating set is not bidimensional because when I take minors the dominating set size can increase.

(Refer Slide Time: 38:37)

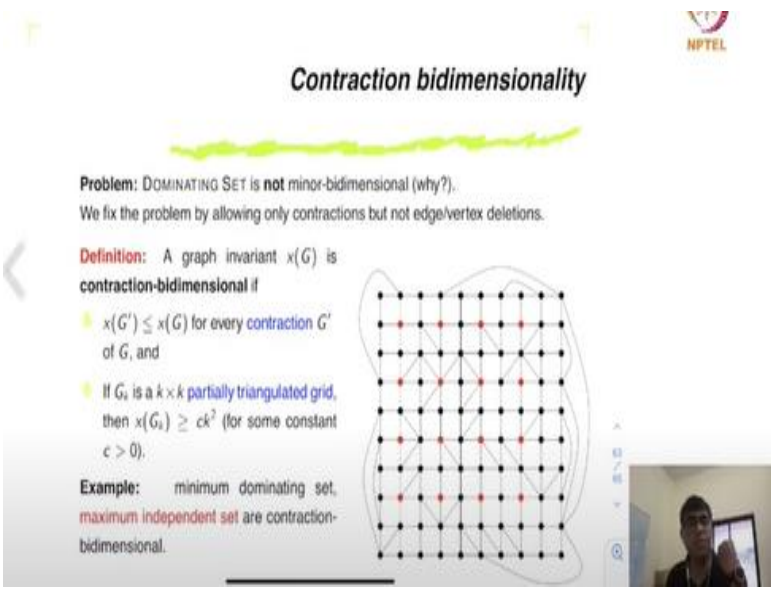


So, we will fix this problem by allowing only contraction but not edge or vertex relation. So, we say look fine dominating set but whatever property of dominating set if you contract a dominating set will not increase and these are the problems which are called contraction bidimensional where the parameter does not change it, where G prime is only obtained by contraction not by vertex or edge deletion.

And but now we cannot talk about k times k grid but we will talk about k times k partially triangulated grid. On this partially triangulated grid the size of your solution it should be order k square, example minimum dominating set or maximum independent set because you at this now look at the central vertex where are his neighbours, his neighbours are here here here where are you here like.

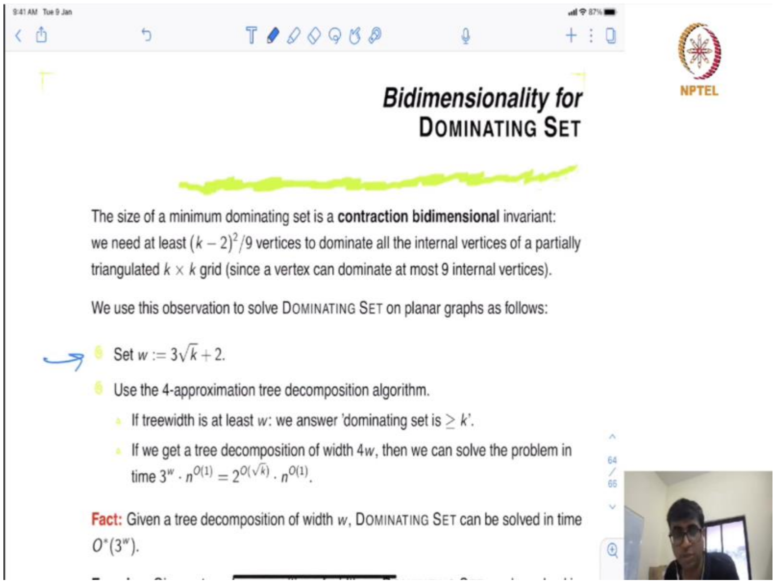
So, you have found set of vertices whose close neighbourhoods are paired by this joint and if you have found the set of vertices with like pairwise disjoint in closed neighbourhood that is a lower bound on the dominating set. So, what we know that minimum dominating set maximum independence set in fact contraction bidimensional but they are not minor bidimensional.

(Refer Slide Time: 39:58)



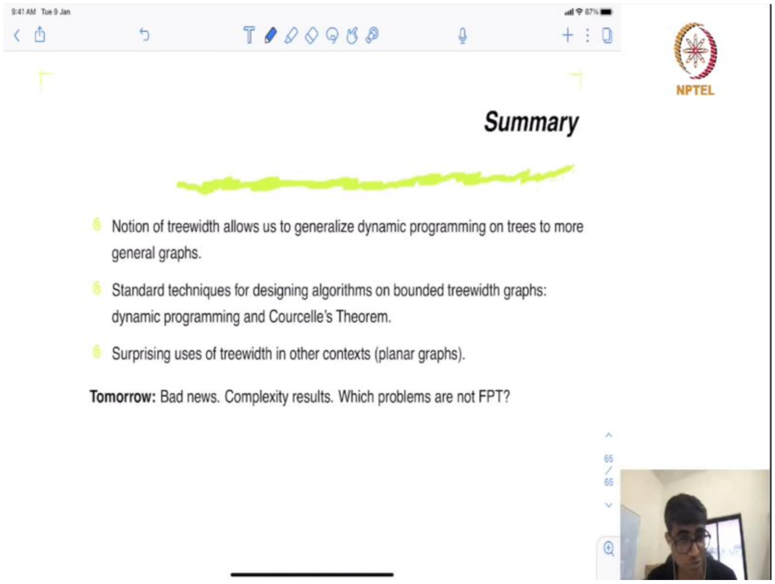
And look at the maximum independent set. Again, you look at a vertices which are like, so that the neighbourhoods are disjoint then that is a lower bound on the solution side. So, these problems which are close in the contracts in the sense that the solution size does not increases when we take the contraction and the solution is quadratic on partially triangulated grid. Then these kinds of problems are called contraction bidimensional. But then the story is can be repeated similar to what we did for minor bidimensional.

(Refer Slide Time: 40:32)



So, now you set up again appropriate w you compute four approximate solution if treewidth is large you know this is more than k. Otherwise you have a small treewidth and you can get you can get dominating set can be actually solving 3 to the power w so you can get you can show that dominating set is sub exponential. So, even for contraction bidimensional problem it all like if it is a contraction bidimensional problem then you can immediately get a sub exponential algorithm.

(Refer Slide Time: 41:08)



So, as a summary our notion of treewidth allows us to generalize dp on trees to more general graphs. Standard techniques for designing algorithms and boundary tree width graphs we thought dynamic programming and Courcelles theoram. And we saw surprising use of treewitdh in other contexts such as planar graphs and in designing FPT. So, I think with this I will close our session on treewidth.

There are we can talk a lot more things about treewidth. But for a course which is as basic as this I think amount of information amount of knowledge we have learnt or we have talked or discussed is enough. And if you need more then please let me know I will give you some more pointers as well as I will give you some more pointers. Thank you.