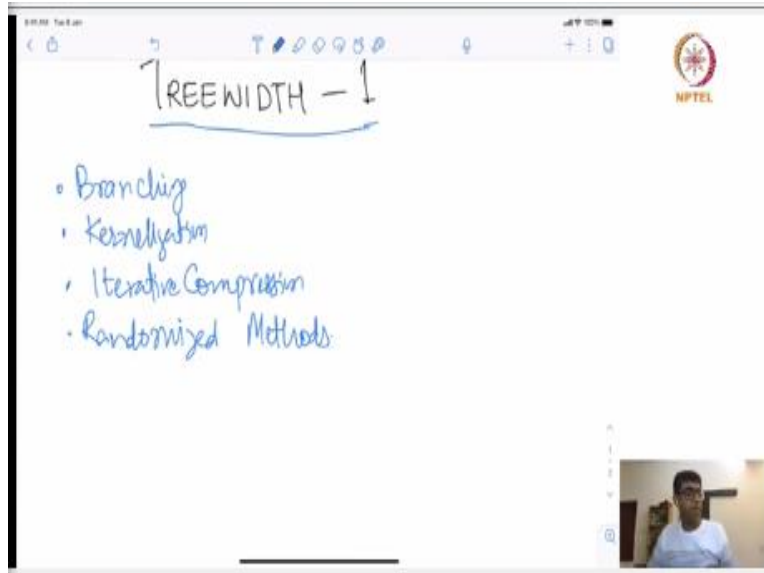


Parameterized Algorithms
Prof. Neeldhara Misra, Saket Saurabh
Department of Computer Science and Engineering
Indian Institute of Technology-Gandhinagar

Lecture-21
Divide and Conquer and Separator

(Refer Slide Time: 00:16)



Welcome to week 5 of the course, up until now we have seen 4 important techniques in the field of parameterized complexity including branching, kernelization, iterative compression and randomized methods. In the next coming 2 weeks we will be doing an another important tool techniques in parameterized complexity which is called treewidth. So, before we give even before we come to the definition of this new tool or new technique, we will go back and start looking at or design an algorithm for another problem.

(Refer Slide Time: 01:26)

Randomized Methods

MIS
 Maximum Independent Set [MIS]
Input:- G
Output:- An independent set of maximum size.

So, the problem which we are going to deal is called MIS or maximum independent set, MIS. So, input is going to be G and we need to output an independent set of maximum size. So, we are looking for an; we would like to output an maximum independent set of size as large as possible. So, it is basically a complement of minimum vertex cover problem where our objective is to find minimum or smallest size subset of vertices.

Such that every edge has an end point and in the outside on the complement of minimum vertex covered, it is a set of vertices that do not have any edge contained inside it, so it is a complement of minimum vertex coupled problem.

(Refer Slide Time: 02:50)

Output:- An independent set of maximum size.

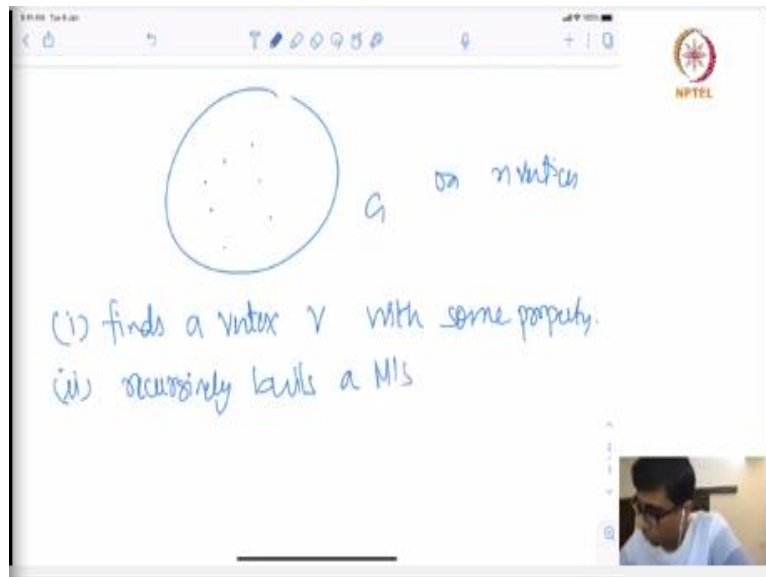
Parameterized Vertex Cover

MIS parameterized by $|V(G)|$.

- not the # of vertices
- exact exponential time algos. moderately ??

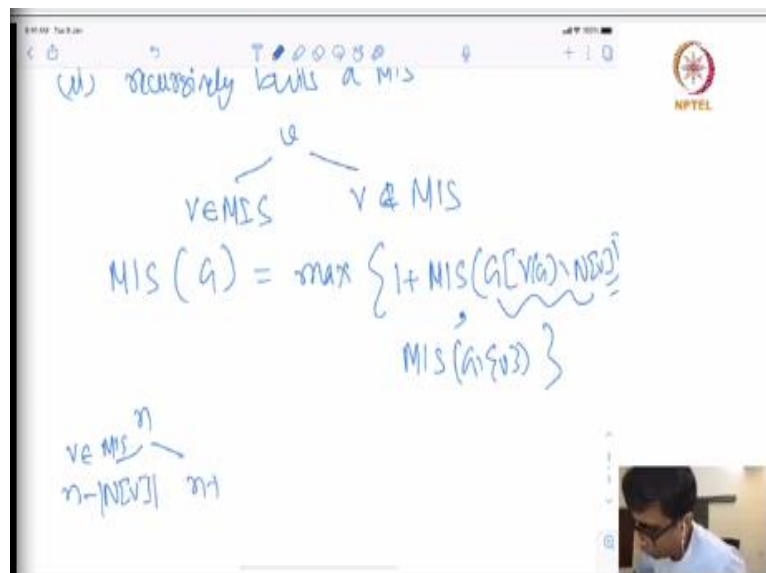
So, inspired by our minimum vertex covered algorithm or the parameterized vertex cover algorithm let us try to construct an algorithm for minimum independent for max independent set parameterized by size of the graph or either word parameterized by n the number of vertices. In literature these are also called exact exponential time algorithms or moderately exponential time algorithms. So, what is our generic algorithm will be? So, the generic algorithm for this problem will be the fact follows.

(Refer Slide Time: 04:01)



So, here is the graph G on n vertices, so the algorithm basically does is the following finds a vertex V with some property and recursively builds a MIS, what does that mean?

(Refer Slide Time: 04:36)



So, I will take at this vertex v and we say or we branch now, we say either v is part of maximum independent set or v is not part of the maximum independent set. If v is a part of the maximum independent set then what can we say? Well, so maximum independent set, so if I have to write a small recurrence it will be going to be max independent set of $G = \max$ of v is in the maximum independent set of G .

So, if I am the kind of maximum independence set I am looking for either that contains v or it does not contain. If it contains v then the maximum independent set cannot contain any of its neighbour. So, the problem reduces to finding 1 plus maximum independent set of graph minus neighbourhood of V or let us say let us write it better $VG - \text{closed neighbourhood of } V$. Or it does not contain V in the maximum independent set.

Then the maximum independent set of G is same as maximum independent set of $G - V$, so this becomes maximum independent set of $G - V$, so this is it. Now if I look at what does it reduces? So, notice that in one branch there are 2 branches n then v is in the maximum independent set, the size of this is going to be at least n minus this and in this size it is $n - 1$.

(Refer Slide Time: 06:24)

$$T(n) \leq T(n-1) + T(n - |N(v)|)$$

- select v such that the size of $|V(v) - N(v)|$ is minimized

\Rightarrow by selecting a vertex of highest degree

So, the recurrence running time is governed by T of n is T of $n - 1$ because of this and T of $n - \text{neighborhood of } V$. Now if we want to get good running time, we want to select V , so this is exactly what we do here right, what we do? We select V such that G size of such that the size of

is minimized. And that precisely happened by selecting a vertex of highest degree. And so by coming up with some simple reduction rules and everything it can actually.

(Refer Slide Time: 07:34)

$T(n) \leq T(n-1) + T(n-1)$
 • select v such that the size of $|V(v) - N(v)|$ is minimized
 \Rightarrow by selecting a vertex of highest degree

select v such that $\deg(v) \geq 3$
 $T(n) \leq T(n-1) + T(n-4)$
 $v = 1.3803^n$ algorithm.

You can show that we can always select v such that degree of v is at least 3 because if the graph has degree at most 2 you can solve the problem in poly time. So, this recurrence will lead to T of n being T of $n - 1$ and T of $n - 4$ and it should lead to some good number like 1 point let us see how much is 1, 4? So, 1, 4 solves to 1.3803, so this will be like 1.3803 to the power n algorithm, excellent. Now I want to understand that what happens that rather than find a vertex V with small property recursively build a MIS, we do as follows. And suppose we are lucky or whatever we will come to that.

(Refer Slide Time: 08:58)

• find a vertex v such that

- the size of a connected component in $G-v$ is minimized.

Size of these components as small as possible

I want to find a vertex v such that the size of a connected component in $G - v$ is minimized. So, what I want? I want to find a vertex v such that if you delete v there could be several components, size of these components are as small as possible. Now why am I asking us to do this? Because then the running time of these algorithms if you notice is going to be, so what will you do?

(Refer Slide Time: 10:09)

possible

$\leq n/2$ $\leq n/2$

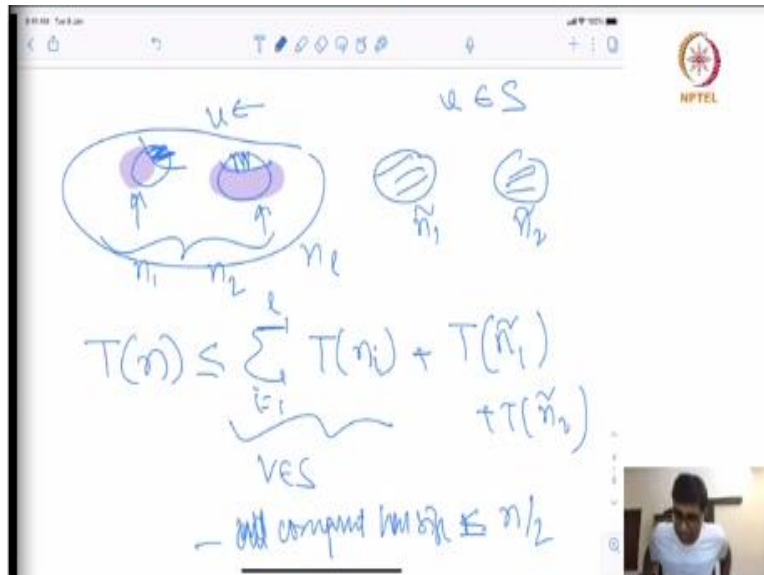
$v \in S$

$G-NB]$

So, again we will say suppose v has 2 components, so $G - v$ at least gets 2 components. And say both has side say n by 2, n by 2, it is a great like. Now notice that the running time of this algorithm is going to be, what will you do? So, in one side when v is in the solution, what will

you do? Well, then the running time of this algorithm is going to be G minus you will delete neighborhood of V .

(Refer Slide Time: 10:47)



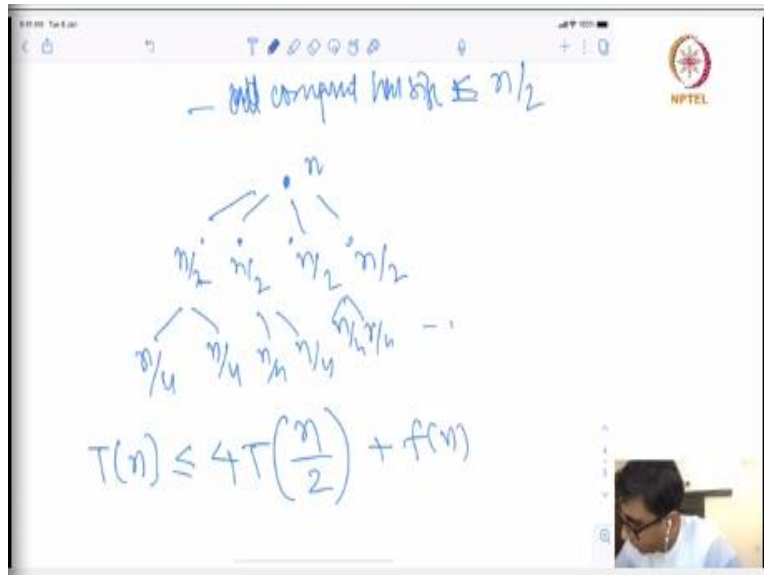
Then it is possible that what are you going to get is that like you will delete v , you might also delete some of its neighbours and here, you will get smaller, smaller components or suppose in other branch when v is not in the solution you are still guaranteed to have 2 components of size at most $n/2$. And in this case you might have many components but each component has size at most $n/2$.

Because we only had 2 components of size at most $n/2$ and now if you delete more vertices from these components you can only get smaller components but not more than $n/2$. So, the running time of this algorithm if you notice is going to be, T of n is going to be, so running time for this is nothing but it I can write it down summation T of n_i suppose it leads to several components of say n_1, n_2, n_l .

And so i going from 1 to l , this is one side, this is when v belongs to the solution. And look at this when v does not belong to the solution then you have summation $i = 1$ to T say T of n_1 tilde because they only have 2 components. So, let us say let us just write them down explicitly T of say n_1 tilde and say this is n_2 tilde n total. So, this is what the recurrence says.

And now if we can ensure that in every iteration every recursion we are able to find a vertex V such that it has 2 components and no component has size more than n by 2. And for now let us just believe that like even here no component has size.

(Refer Slide Time: 13:28)



We also assume the no components like or all components has size less than n by 2. Now what happens in this case? So, now if you are able to do this and able to do this recurrence then let us look at the picture by what is happening? I have my graph n and suppose I create 2 sub problem here of size n over to each. And then I create another because every iteration I am able to for every component of the graph I am able to find a vertex.

So, that if I delete it both has a smaller component of size at most half of what it is, then you will get n over 4, n over 4, n over 4 so on and so forth. So, the running time of this algorithm is T of n and suppose you are able to do. So, this is going to be right 4 times T of n over 2 plus say some time to find f of n is time to find the vertex v of desired kind.

(Refer Slide Time: 14:53)

The whiteboard contains the following handwritten content:

- At the top, a sequence of terms: $\frac{n}{4}, \frac{n}{4}, \frac{n}{4}, \frac{n}{4}, \frac{n}{4}$.
- The recurrence relation:
$$T(n) \leq 4T\left(\frac{n}{2}\right) + f(n)$$
- An arrow points from $f(n)$ to the text: "time to find the vertex v of desired kind."

The NPTEL logo is visible in the top right corner of the whiteboard interface.

And you can show that this is like some polynomial, this is like some fixed polynomial it is n to the power O of order 1. And so what I am trying to tell you that it is a very nice that this algorithm which is exponential in general but suppose you are guaranteed that in each time you are getting a connected components which are at most half of my original instance. Then the running time of this here every time you want to branch you are able to find a vertex.

Such that you are able to decompose, you are able to like delete one vertex and get smaller connected components. Then the running time of this whole algorithm is actually polynomial and it is not an exponential. Now I will construct such an example soon for you. But just to keep in mind what property am I using one property that I try to use in this algorithm is that, this vertex had a very interesting property that this vertex was able to separate my graph or my partition my graph into 2 parts, no part is very big.

Like what is because now the running time of the algorithm is you are summing the running time of these algorithms. Like and sum of 2 smaller instances is in the terms of exponential will be much, much smaller compared to say $n - 1$ size instance.

(Refer Slide Time: 16:50)

$\sim n^{O(1)}$
MIS on Paths
 $w: V(n) \rightarrow \mathbb{R}^+$
 find max weight ind set of G
 $w(S) = \sum_{v \in S} w(v)$

So, if you would like to exploit this kind of properties, now hey I did this and it worked out very well. But maybe you might think that I am talking about some sort of imaginary graph classes. So, where are such graph classes where I could find such kind of object or such kind of recurrence? So, let us just try to do maximum independent set on paths and just to make things interesting let us add a positive weights on the vertex set of G .

And what we want is to find max weight independent set of G and like what is the weight of a set? So, weight of a set S is summation v in f weight of v , so you like to find an independent set such that. And if I sum the weights of the vertices in this independent set that is maximum.

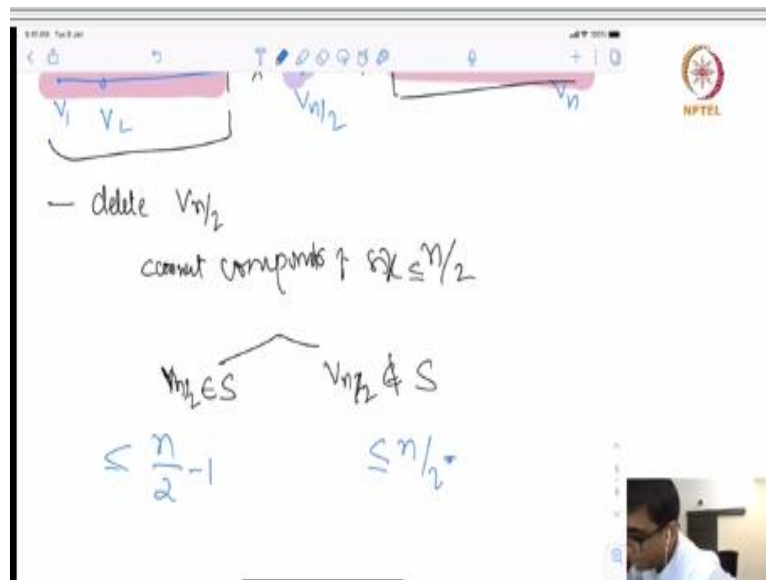
(Refer Slide Time: 17:55)

$w: V(n) \rightarrow \mathbb{R}^+$
 find max weight ind set of G
 $w(S) = \sum_{v \in S} w(v)$

\rightarrow delete $v_{n/2}$
 convert components to size $\leq n/2$

So, now what is given to me what is given to me is some path $v_1, v_2, \dots, v_{n/2}$ and $v_{n/2+1}, \dots, v_n$ and there are some weights and just to be a let us write them $w_{n/2+1}, \dots, w_n$. So, now let us look at the middle vertex, this is the middle vertex, what is the property? The property is what is the property; delete $v_{n/2}$ has a property that you get connected components of size at most $n/2$. Now if I branch on this what will be my branching algorithm?

(Refer Slide Time: 18:57)



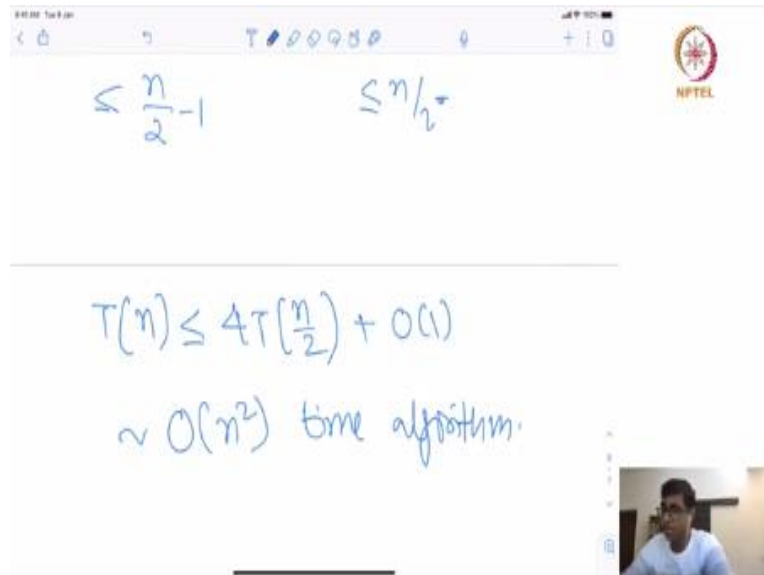
We will say like here it is $v_{n/2}$ is in the solution or $v_{n/2}$ is not in the solution. So, now what happens in this case? If $v_{n/2}$ is in solution then you are looking to find a maximum weight independent set without like, so suppose these are its neighbours on the path. Then you are trying to find a maximum weight independent set here like up to here and up to here in this case in these 2 component.

So, in this case what is the size of this component? The size of this component is at most these 2 components has the size of the components are in this case is at most $n/2$ minus maybe 1. But in the other case I am saying that look I am not looking to contain $v_{n/2}$ as my max weight independent set. Then what is my component size? This is at most again sum $n/2$. So, basically and notice that recursively what is the property?

So, now if I give you some small piece of sub path and I want to find a vertex such that both parts are equal that is again very simple. You look at the sub path and find the middle vertex of

that sub path and that will satisfy the property that every component has size at most half of whatever we started with.

(Refer Slide Time: 20:40)



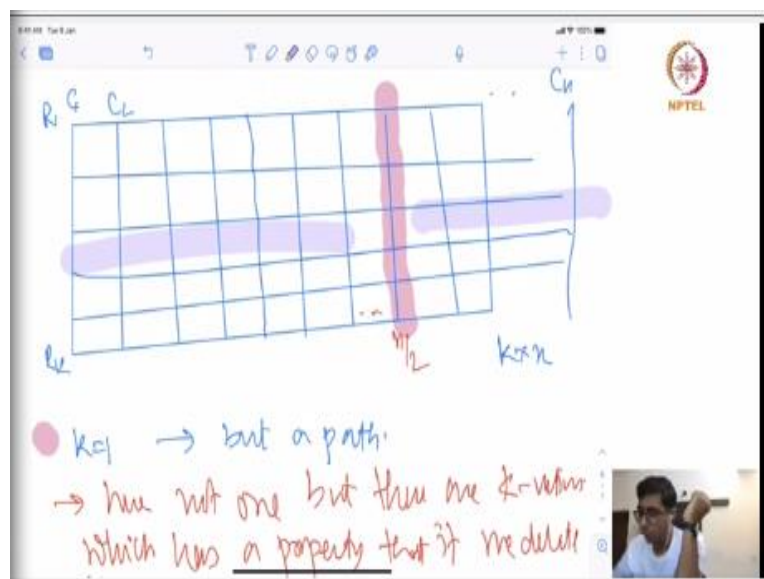
$$\leq \frac{n}{2} - 1 \quad \leq \frac{n}{2}$$

$$T(n) \leq 4T\left(\frac{n}{2}\right) + O(1)$$

$$\sim O(n^2) \text{ time algorithm.}$$

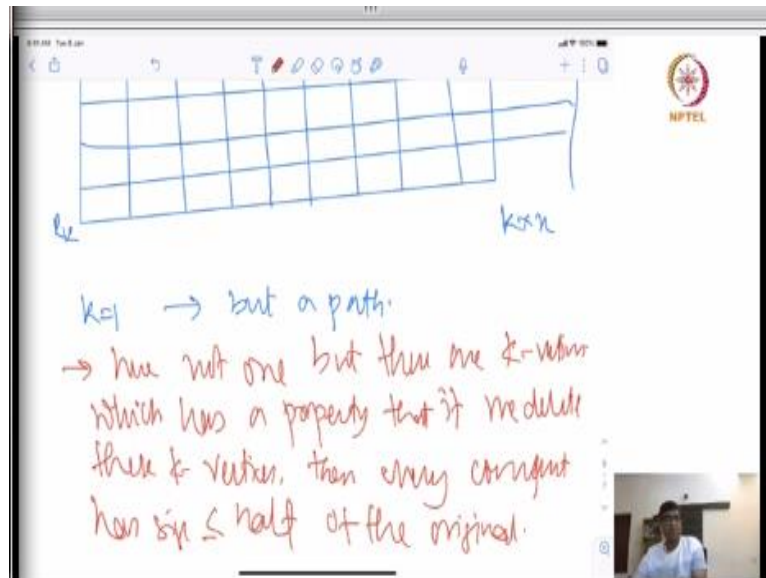
Now it implies that for this family the running time is T of n is 4 times T of n over 2 and plus some constant to find the middle vertex. And we can show that this is like at most order n square time algorithm. So, yeah I mean the algorithm which I talked about is like a single vertex branching that is great. But so path was very simple example path was very simple family of graphs. What other graphs are there where we should be able to find such kind of algorithms?

(Refer Slide Time: 21:27)



So, let us say let me write down another family of graph, let us I call it let us say wide path or fat path whatever you want to call it, does not matter. So, basically it is, so let us look at the grid, so this is like some k cross n grid meaning there are k rows R_1, R_2, \dots, R_k and there are n columns, some n columns. So, this is why I am going to call it a k cross n grid.

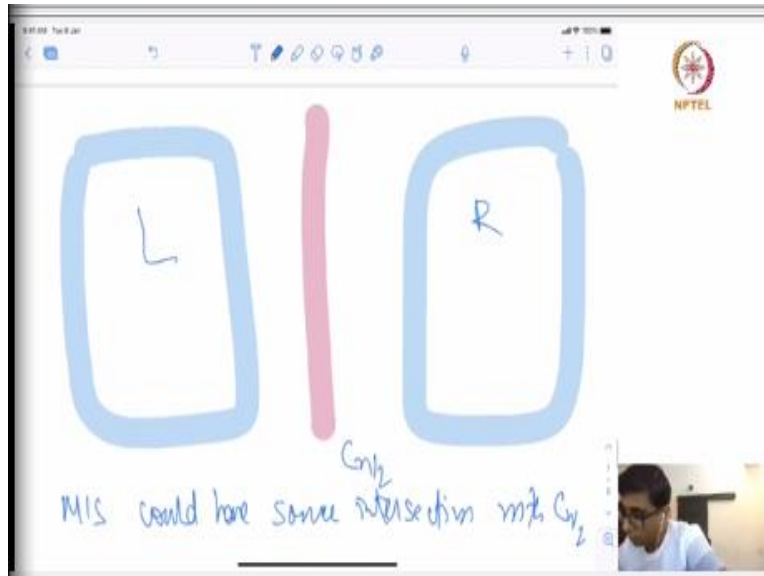
(Refer Slide Time: 23:19)



So, notice $k = 1$ is nothing but a path. And now notice that here we may not have a one vertex here I cannot say I look, here not one but there are k vertices which has a property that if we delete these k vertices then every component has size at most half of the original grid. So, this is a very nice, here I do not know how to look because if I just delete one vertex you will this grid like you cannot even disconnect them.

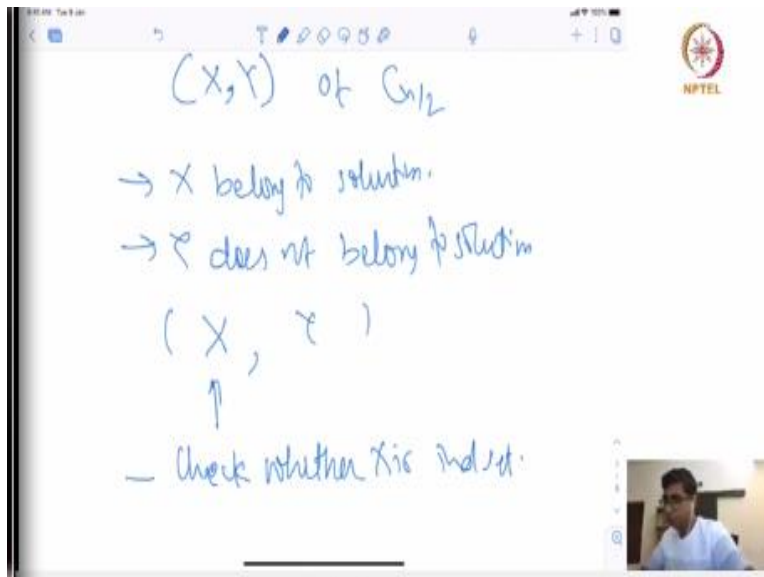
I mean, but if you look at the middle suppose this is like somewhere n by 2 , suppose this is n by 2 . Now look at these k vertices if I delete these k vertices then there is a well defined left side and the well defined right side and each side contains half the vertices. And each side contains half the vertices of my graph. So, now I would ask myself is that can I exploit these k vertices on this like n by 2 th column? And the answer is fine let us try to do it.

(Refer Slide Time: 25:19)



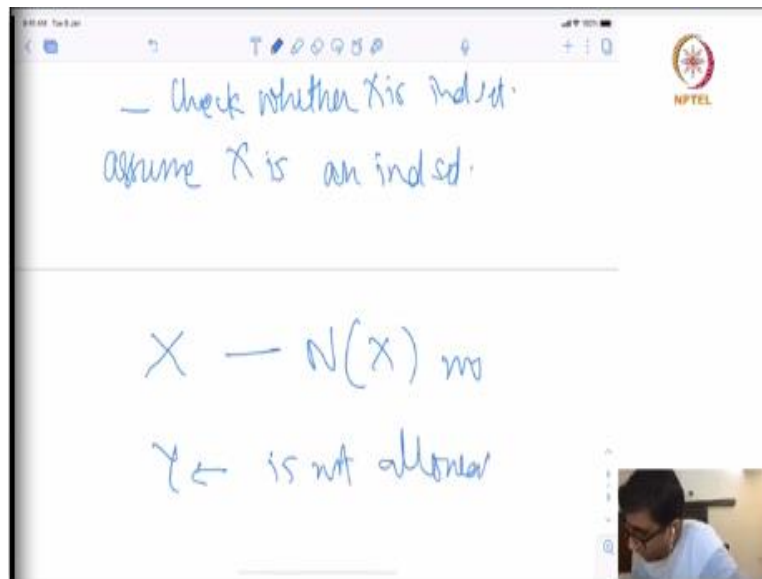
So, but now there are cavities I do not know, so now let us look at the picture, picture is that here is our grid and here is our left side and here is our right side, left, right. Now I want to find a maximum independent set, so the maximum independent set could have some intersection with C_n by 2 columns. We do not know what that intersection is, so what is this?

(Refer Slide Time: 26:17)



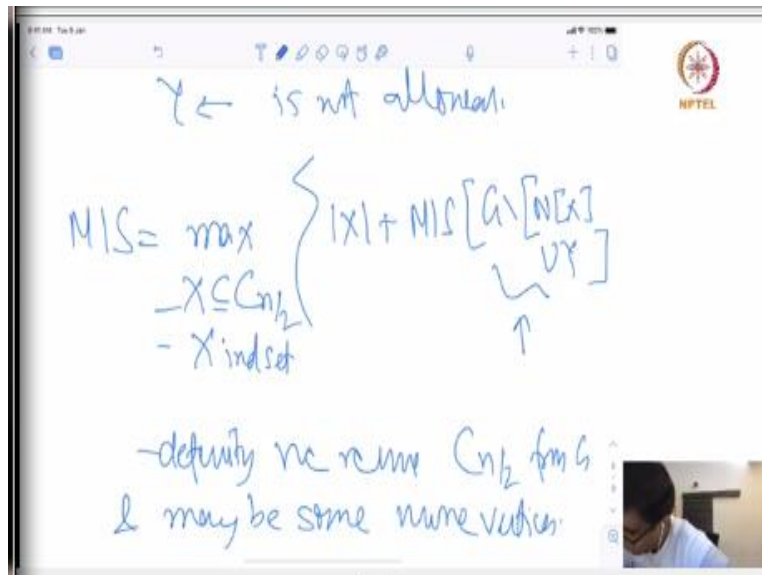
So, suppose, so basically it is like a partition x, y of column n by 2 and what is this? X belongs to solution that we are seeking; y does not belong to the solution. Now notice what can we say about it ok, fine. Here it is X is there and Y is there, so first of all if I am looking for a solution that contains X , check whether X is an independent set or not. If it is not then this branching of x , into x, y is not a valid branching.

(Refer Slide Time: 27:08)



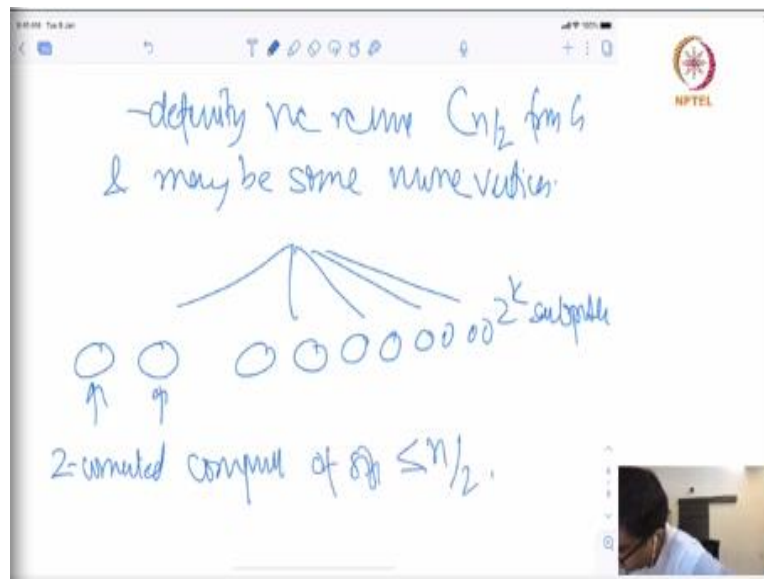
So, assume X is an independent set, now what can we say ok, fine if you are going to contain X then 2 things you cannot contain, neighborhood of X , no way like then you cannot contain neighborhood like open neighborhood of X and you cannot and by definition Y is not allowed.

(Refer Slide Time: 27:37)



Then basically what happens? So, maximum independent set becomes, notice max over X subset of say column n by 2, that is one thing, X is an independent set, right and it is size of X because that you are going to contain. And this is nothing but max IS of which graph $G - \text{close neighborhood of } X \text{ union } Y$. But notice this graph; definitely we remove column n by 2 from G and maybe some more vertices. So, what happens?

(Refer Slide Time: 28:33)



So, we are creating 2^k sub problems, so when remember in path we had 2^k sub problems. And in each we are going to solve problem on 2 connected components of size actually at most $n/2$.

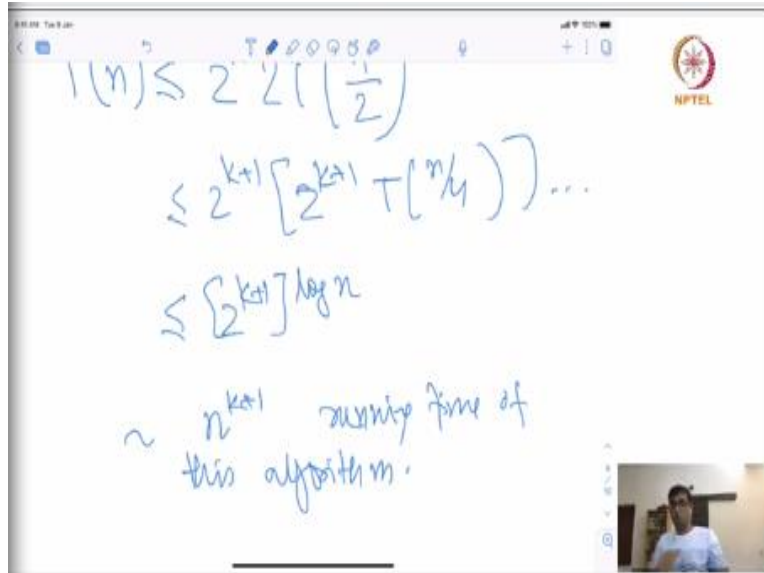
(Refer Slide Time: 29:04)

$$\begin{aligned}
 T(n) &\leq 2^k \cdot 2T\left(\frac{n}{2}\right) \\
 &\leq 2^{k+1} \left[2^{k+1} T\left(\frac{n}{4}\right) \right] \dots \\
 &\leq [2^{k+1}] \log n
 \end{aligned}$$

It means our recurrence in this time is going to be T of n is 2^k sub problems, in each there is one more sub problem, so that is like 2. And the size is like $n/2$, this is exactly what we are going to solve it. Now so basically if you extend this recurrence you are going to get 2^{k+1} , then again you are going to get again you will get some 2^{k+1} times T of $n/4$ and so on and so forth.

So, this is going to come $2k + 1$ but how many times? $\log n$ times, because n by 2, n by 4, n by 8, n by 16 so on and so forth and in $\log n$ terms this is. So, here and so this is why this is upper bounded by this.

(Refer Slide Time: 30:09)



$$\begin{aligned}
 T(n) &\leq 2T\left(\frac{n}{2}\right) \\
 &\leq 2^{k+1} \left[2^{k+1} T\left(\frac{n}{4}\right) \right] \dots \\
 &\leq 2^{k+1} \log n \\
 &\sim n^{k+1} \text{ running time of this algorithm.}
 \end{aligned}$$

And if you do this it is going to come to you roughly n to the power $2^{\log n}$ is n to the power $k + 1$, roughly this is the running time of this algorithm. So, notice that this is a great strategy that if a family of graph has a property that for any connected component what is the property? That for any sub graph there exist a vertex of some size whose deletion has a property that every connected component has small number.

So, this tells us a following family of graphs, so let us define this following family of graph. So, let us to do that let me I have now I will take a path backward and let me define some simple notion.

(Refer Slide Time: 31:10)

Defn. A set X is called a balanced separator of G if in $G - X$ every connected component has size $\leq \frac{|V(G)|}{2}$.

$\text{balspn}(G) = \text{size of a minimum sized balanced separator}$

$\text{spn}(G) = \max_{V' \subseteq V(G)} \left\{ \text{balspn}(G[V']) \right\}$

\uparrow
separation # of G

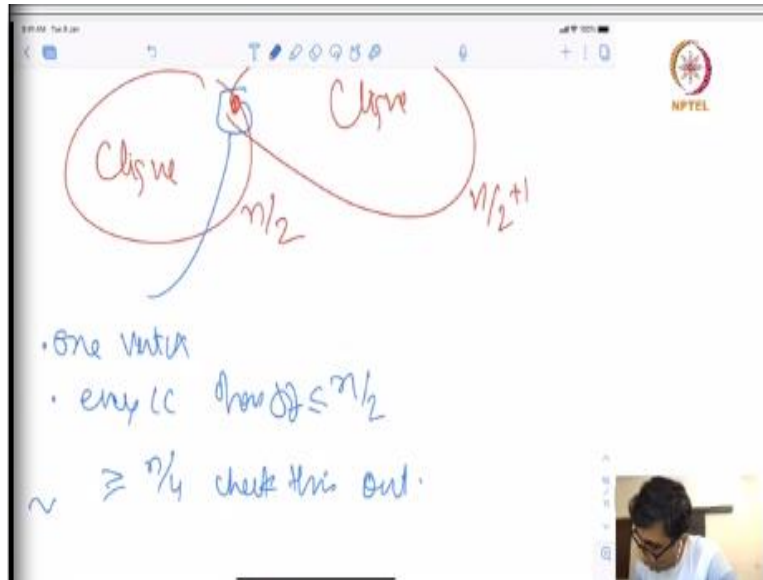
\uparrow
minimum sized balanced separator of $G[V']$

So, now let me define a some simple notion. So, let me define, a set x is called a balance separator of G if in $G - X$ every connected component has size at most $\frac{|V(G)|}{2}$, so this is my first definition. Now let me define this is like a separation number of G , this is max over V prime subset of $V(G)$, balance separation number of graph induced on V prime. Now what is this balance separation number of graph induced on this?

This is basically what is this minimum sized balanced separator of G of graph induced on V prime. So, what is a balance separation? A set of vertices such that if you delete it every component has size at most the half of the graph and so basically what happens is the balance separator number of G is minimum size is a basically let us say size of a minimum sized balanced separator.

So, like what is the smallest size subset whose deletion has the property that every component has size at most half of the original is the balance separation. And what is the separation number of graph? You go over every vertex set of my graph and say, what is the balance separation number of this meaning? For any induced sub graph what is the minimum sized vertices which does this job and I take maximum over this, so for example look at this example of very nice example.

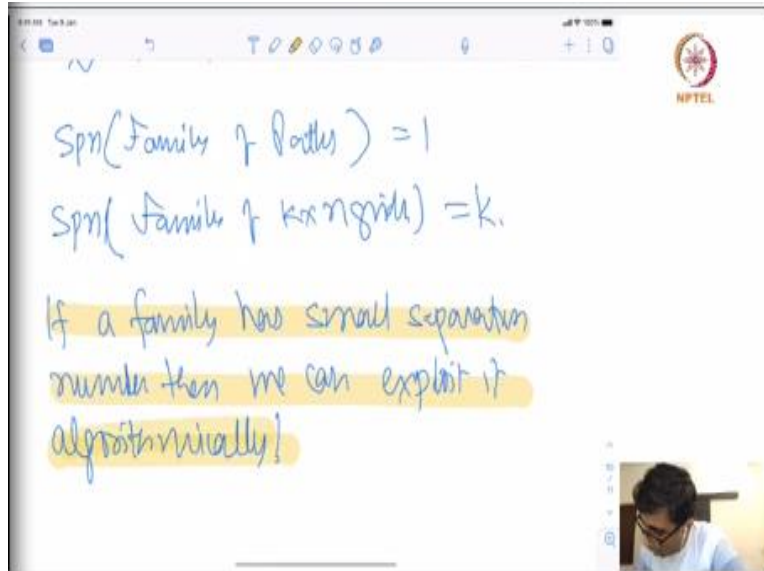
(Refer Slide Time: 34:57)



Say here is a clique on say n by 2 vertices and here is another clique on say n by $2 + 1$ vertices. Now notice that if we delete this vertex which is one vertex, if we delete this one vertex then you get that every connected component has size at most n over 2 . But now look at a clique, you cannot delete less than n by 2 vertices and have a component of size at most half of what you started with.

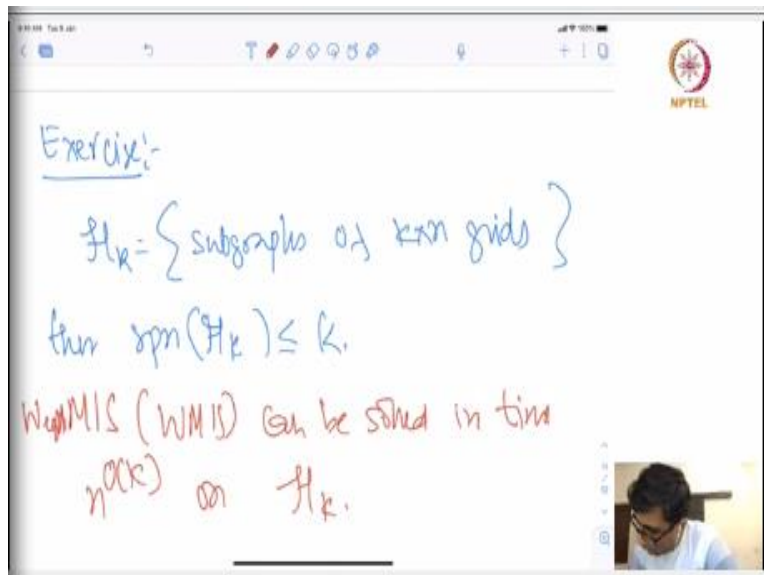
So, if you wanted to get a component of size n by 4 from this clique you at least need to delete n by 4 vertices. So, if you notice that although the balance separation number of the G itself is 1 , balance separation of induced sub graph could be much larger. So, you can show here that probably what is the balance separation number of this graph? I claim that it is at least n over 4 and you check this out. But what is a vertex? What is the separation number of family of paths?

(Refer Slide Time: 37:09)



So, but separation number of family of paths is 1, what is your separation number of family of k cross n grids? It is k . So, it seems like that if a family has small separation number then we can exploit it algorithmically. So, this is our mantra that if a family has a small separation number then we can exploit it algorithmically.

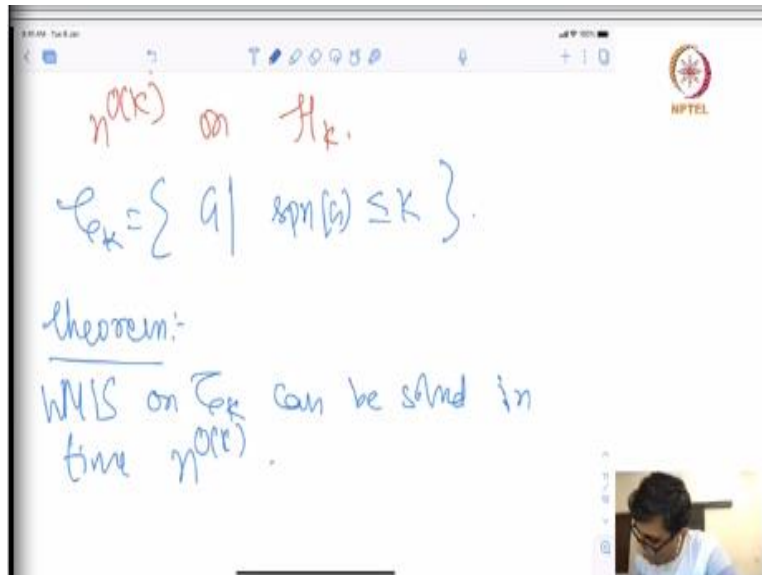
(Refer Slide Time: 38:21)



Here is an exercise of convince yourself. Let H_k be sub graphs of k times n grids. Then separation number of H_k is at most k . So, not only look and then what we have shown? We have shown that maximum independent set, what we have shown is that maximum independent set in fact weighted MIS let us call it WMIS can be solved in time $n^{O(k)}$ on H_k . Because there was nothing that we used about the grid because all we used in the algorithm for

this is that find a k sized balance separator, you branch on possible base and then I use solving problem recursively. So, the theorem which we have proved.

(Refer Slide Time: 39:44)



$n^{O(k)}$ on H_k .

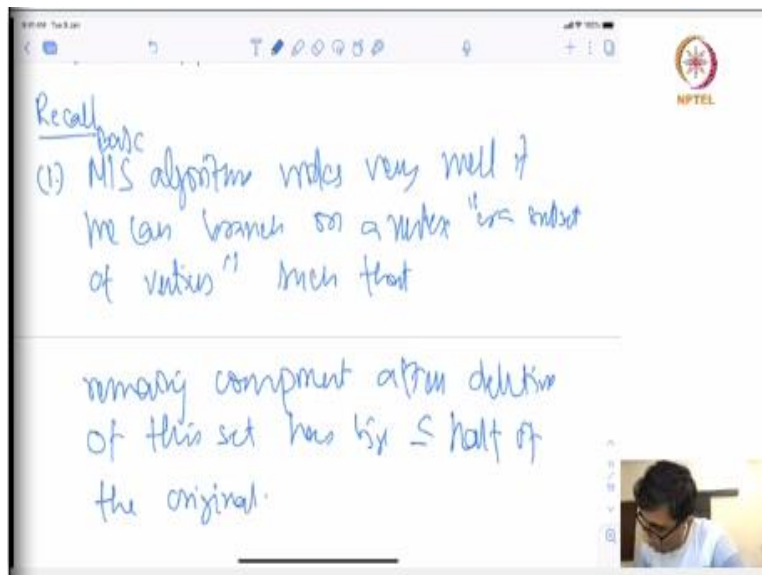
$C_k = \{ G \mid \text{spn}(G) \leq k \}$.

theorem:

WIS on C_k can be solved in time $n^{O(k)}$.

Let us call it Z_k if family of graphs or rather say what is C_k contains? All those graph G such as a separation number of G is at most k . So, what are we theorem we proved? Theorem we proved is that weighted maximum independent set on this can be solved in time n to the power O of k .

(Refer Slide Time: 40:26)



Recall base

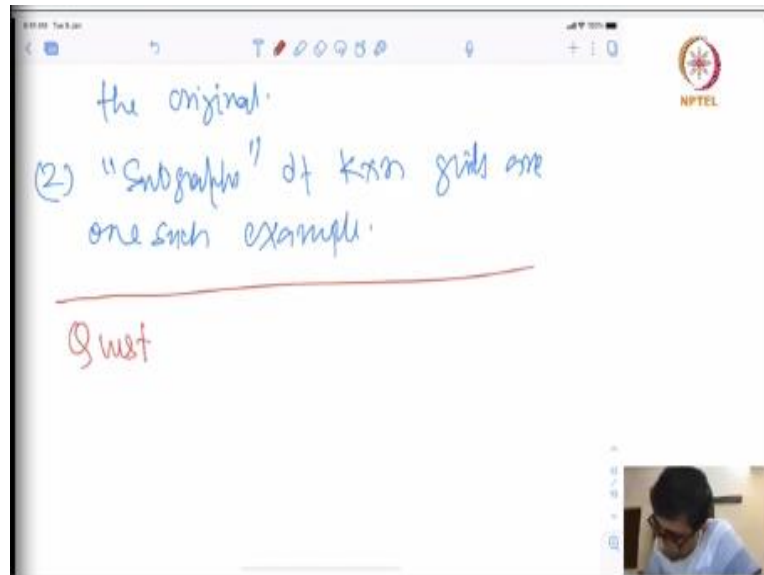
(1) MIS algorithm works very well if we can branch on a vertex "is in set of vertices" such that

remaining component after deletion of this set has size \leq half of the original.

Now notice that up until now we looked at the family of power, so let us just try to recall. So, the point what we made is that MIS algorithm works very well if or the basic MIS algorithm works

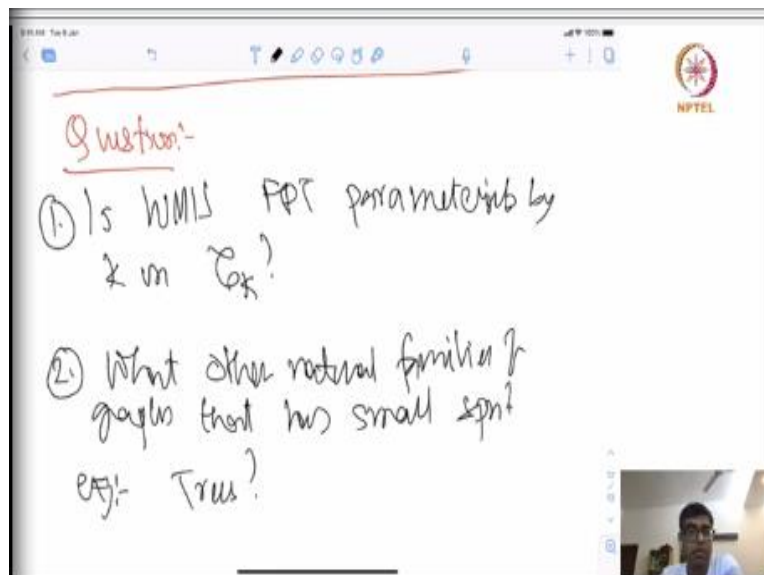
very well if we can branch on a vertex or a subset of vertices. Such that remaining component after deletion of the set has size at most half of the original, so this is what we saw first.

(Refer Slide Time: 41:16)



And secondly we saw that sub graphs of k cross n grids are one such example. So, let me end this class with 2 questions and we will try to find answer to these 2 question, in the next lecture. So, what is my question?

(Refer Slide Time: 42:20)



My first question is; Is weighted MIS fixed parameter tractability tractable parameterized by k on this? That is a question number first. Second, what other natural families of graphs that have small separation number, small spn ? Example may be trees. So, that will form answer to these 2

questions will form the main genesis of next lecture. And hopefully from there we will be able to get a definition which will be useful for our purposes. So, that is all for the first lecture on treewidth, we will connect back later.