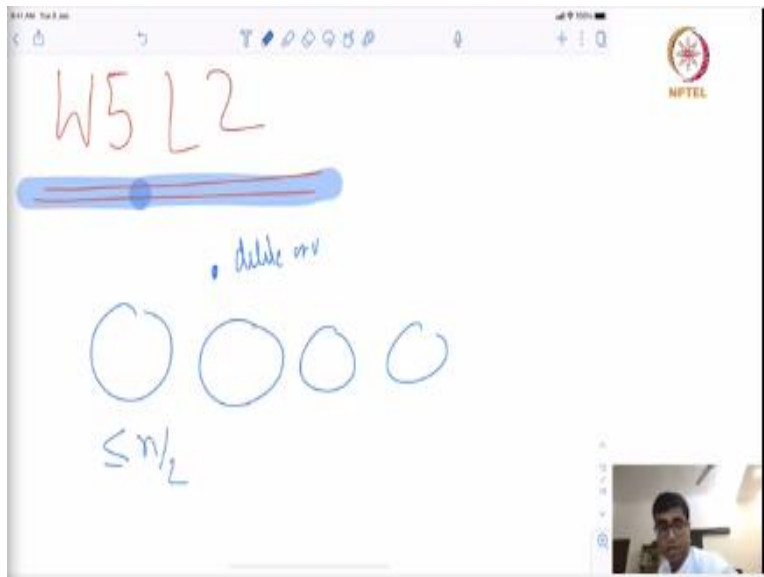


Parameterized Algorithms  
Prof. Neeldhara Misra, Saket Saurabh  
Department of Computer Science and Engineering  
Indian Institute of Technology-Gandhinagar

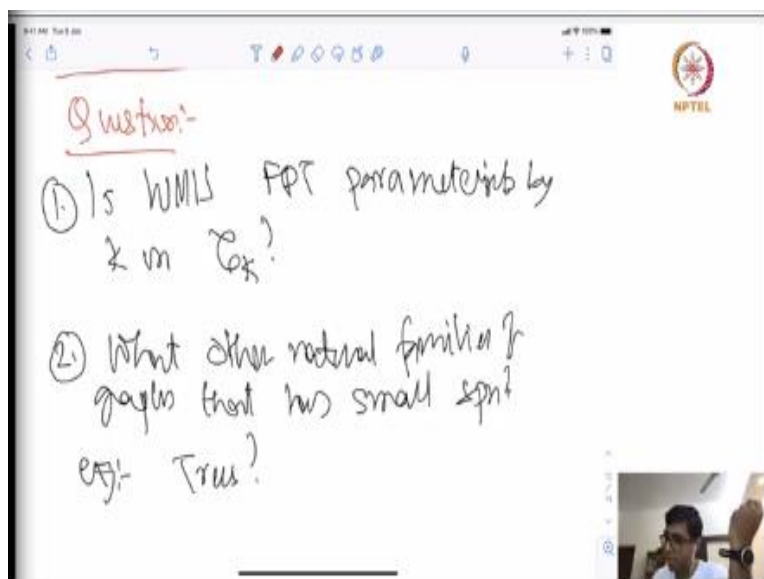
Lecture-22  
Towards Defining Treewidth

(Refer Slide Time: 00:16)



Welcome to the second lecture on treewidth.

(Refer Slide Time: 00:21)



If you recall we stopped last time, so we defined this notion of graph with small separation number which were basically let us recall. But basically what was that?

(Refer Slide Time: 00:30)

Exercise:-

$$\mathcal{H}_k = \{ \text{subgraphs of } K_{m,n} \text{ grids} \}$$

then  $\text{spn}(\mathcal{H}_k) \leq k$ .

WMIS (WMIS) can be solved in time  $n^{O(k)}$  on  $\mathcal{H}_k$ .

$$\mathcal{C}_k = \{ G \mid \text{spn}(G) \leq k \}$$

A graph had a separation number at most  $k$ , if look at any induced sub graph.

(Refer Slide Time: 00:37)

Defn.

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

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

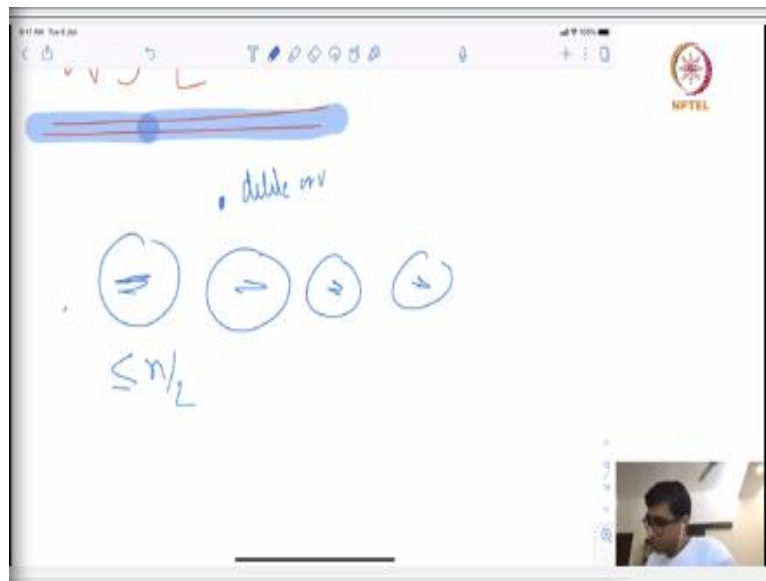
$\text{spn}(G) = \max_{V' \subseteq V(G)} \{ \text{bal\_spn}(G[V']) \}$

$\uparrow$   
separation # of  $G$

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

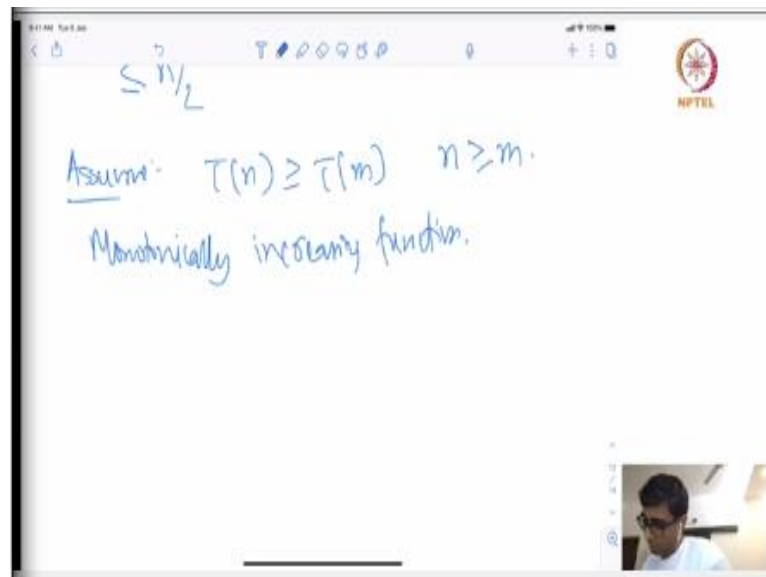
You can find at most  $k$  vertices whose division has the property that every connected component has small size. Now, notice that whenever I was trying to make this MIS algorithm, I try to write this recurrence, I mean which you may not be happy with is that. There are only 2 sub problems but in general if you notice when I delete one vertex I could get several components, maybe each of them has size at most  $n$  by 2.

(Refer Slide Time: 01:13)



But like I try to not get into these details of summing over all these components and everything. But like let us but let me go through slightly a little bit more about this how you handle if you have several components? To still say that even if you could delete a vertex and every component has size at most  $n$  by  $2$ , then we still have a good polynomial time algorithm, so let me try to do that for you.

(Refer Slide Time: 01:45)



So, first of all, let us I am going to make some assumption, our very basic assumption about algorithms that  $T$  of  $n$  is greater than equal to  $T$  of  $M$ , if  $n$  is greater than equal to  $n$ . So, like they are monotonically increasing functions that is it.

(Refer Slide Time: 02:30)

Lemma:  $n_1, n_2, \dots, n_l$  such that  
 $n_i \leq \frac{n}{2}$  &  $\sum_{i=1}^l n_i = n$ . Then there  
 exist a partition of  $n_1, n_2, \dots, n_l$  into  
 $A$  &  $B$  such that  
 $\sum$

So, let us try to understand what I will do or rather? So, imagine, so let us look at a lemma and this lemma is about numbers and there is on purpose I am talking about this key number. So, suppose this is about numbers, we have  $n_1, n_2, n_3, \dots, n_l$  such that  $n_i$  is at most half of  $n$  and summation of  $n_i$ ,  $i$  going from 1 to  $l$  is  $n$ . Then I am going to say then that exist of partition of  $n_1, n_2, \dots, n_l$  into  $A$  and  $B$  such that what is the property?

(Refer Slide Time: 03:26)

Lemma:  $n_1, n_2, \dots, n_l$  such that  
 $n_i \leq \frac{n}{2}$  &  $\sum_{i=1}^l n_i = n$ . Then there  
 exist a partition of  $S = \{1, 2, \dots, l\}$  into  
 $A$  &  $B$  such that  
 $\frac{n}{3} \leq \sum_{j \in A} n_j \leq \frac{2n}{3}$  /  $\frac{n}{3} \leq \sum_{j \in B} n_j \leq \frac{2n}{3}$ .

Such that summation rather there exist a partition of 1 to  $l$  into  $A$  and  $B$  such that summation  $j$  into  $i \in A$   $n_j$  has a property that it is at most  $n$  over 3 and at most  $2n$  over 3. And similarly,  $j$  and  $B$  has a property that  $\sum n_j \leq 2n/3$ . So, what I am saying that if you give me  $l$  numbers such that each

of them is at most  $n$  over 2, there is summation in  $n$  then I can partition these numbers into 2 parts. Such that each part has some threshold and there is a like threshold from both lower bound and upper bound and we will see how we can use this. So, let us prove this first.

(Refer Slide Time: 04:27)

Let  $A$  &  $B$  such that  $\frac{n}{3} \leq \sum_{i \in A} n_i \leq \frac{2n}{3}$  and  $\frac{n}{3} \leq \sum_{i \in B} n_i \leq \frac{2n}{3}$ .

proof:-  
arrange these integers in increasing order.  
 $n_1 \leq n_2 \leq \dots \leq n_k$

(i) Case 1:-  
 $n_1 \leq n/3$

It is a very simple fact and proof is also fairly easy. So, let us arrange these integers in increasing order. So, suppose we have you like assume that we have  $n_1$  at least  $n_2$  at least  $n_1$ . So, this is what it is? So, now we have first case, case 1,  $n_1$  is less than equal to  $n$  over 3, then.

(Refer Slide Time: 05:37)

proof:-  
arrange these integers in increasing order.  
 $n_1 \leq n_2 \leq \dots \leq n_k$

(i) Case 1:-  
 $n_1 \leq n/3$

$A = \{1, 3\}$   
 $B = \{2, 1, 3\}$

$\frac{n}{3} \leq n_1 \leq \frac{n}{2} \leq \frac{2n}{3}$

Or rather let us say different is  $n_1$  is greater than equal to  $n$  by 3, this is our first case. If  $n_1$  is greater than equal to  $n$  by 3, then I am going to take  $A = 1$  and  $B = 2$  to 1. Let us see why they

satisfy the whole thing? Now notice, so summation, so what is a property in this case what is the sum of  $n_1$  is at least  $n/3$ ? Because of our assumption and because every number has is at most half, it is at most  $n/2$ , which is at most  $2n/3$ .

(Refer Slide Time: 06:28)

(i) Case 1:-  
 $n_1 \geq \frac{n}{3}$   
 $A = \{1, 3\}$   
 $B = \{2, 1, 3\}$   
 $\frac{n}{3} \leq n_1 \leq \frac{n}{2} \leq \frac{2n}{3}$   
 $n - n_1 \leq \sum_{j \in B} n_j \leq n - n_1 \leq \frac{2n}{3}$   
 $n - \frac{n}{2}$   
 $\frac{n}{2}$

Now look at B, summation  $j$  in B, what is  $n_j$ ? Now, notice that  $n_j$  because  $n_1$  has size at least  $n/3$  over 3 summation of the  $n$ , so it is definitely at most  $n - n/3$ , which is at most  $n/2$  is at least  $n/3$ , so this is  $2n/3$ . Now, what about this, what is summation  $j$  and  $B$   $n_j$ ? Well, you are talking about lower bound. Now, you know that anyone has size at most  $n/2$ , so this is definitely, again I can say at least  $n/3$ .

So, I know that summation  $n_j$  is, so I am going to use the following lower bound. It is  $n - n/2$ . So, here I use the fact that  $n_1$  is at least  $n/3$  to get an upper bound, I am using here  $n$  is at most  $n/2$  to get a lower bound on this, which is at least  $n/2$ . So, in this case you notice that we have been able to find a partition A and B; such that the property is that if I look at the sum of these numbers, there at least  $n/3$  and at most  $n/2$ . Now what about the case 2?

Case 2, so what is the complement of case?

(Refer Slide Time: 07:59)

Handwritten notes on a whiteboard:

- $n - n/2$
- $\sum_{j=1}^k n_j$
- (ii) Case 2:  $n_1 < n/3$
- $k$  is the first integer such that
- $\sum_{j=1}^k n_j \geq n/2$
- $\left[ \sum_{j=1}^{k-1} n_j < n/2 \right]$

The NPTEL logo is in the top right corner.

So, if  $n_1$  is strictly less than  $n$  by 3, now let us say  $k$  is the first integer such that, so you are counting, like I am just counting the what is the first integer  $k$  such that this number is greater than half.  $k$  is the first integer such that summation  $j$  going from 1 to  $k$   $n_1$  is greater than or equal to  $n$  over 2. Like what is the first time what is the first, like so  $n_1$  is the largest number,  $n_2$  a second largest number so on and so forth. And say I am going to sum these guys and the first item  $k$  were find.

That if I look at the summation of first  $k - 1$  it is strictly less than  $n$  by 2 and the moment I add the last item the  $k$ th item, I cross the threshold of this. So, meaning the first just which basically mean that if I look at  $j$  to 1  $k - 1$  and  $i$   $n_j$ , this is strictly less than  $n$  by 2. Now let us try to understand what happens then?

**(Refer Slide Time: 09:31)**

$$n_k \leq n/2$$

$$n_k \leq n_1 \leq n/3$$

$$A = \{1, \dots, k\}$$

$$B = \{k+1, \dots, l\}$$

$$\sum_{j=1}^{k-1} n_j + n_k$$

So, what can we say about  $n_k$ ?  $n_k$ , I know that  $n_k$  is upper bounded by  $n/2$ . Sure, but this lower upper bound is not useful for my purpose, why? We will see them. But we can prove something better. Look  $n_1$  which is the largest integer is at most  $n/3$ , then the  $n_k$  which is a smaller than the  $n_1$  is going to be at most  $n/3$ . So, rather than using this we will use the fact that in fact look at  $n_k$  is less than  $n_1$  which is less than  $n/3$ . So, if that is a case, then I think I made a mistake, so let us not say  $n/2$  let us say  $n/3$ .

**(Refer Slide Time: 10:44)**

$$V1 \quad n/2$$

(ii) Case 2:  $n_1 < n/3$

$k$  is the first integer such that

$$\sum_{j=1}^k n_j \geq n/3$$

$$\left[ \sum_{j=1}^{k-1} n_j < n/3 \right]$$

$$n < n_1$$

So, now let us look at my A, so what is A? is 1 up to K and B is 1 up to say  $k+1$  to  $l$ . Now let us look at summation  $i$  going from 1 to  $j$  or rather. So, summation  $n_i$  or  $n_j$ ,  $j$  going from 1 to  $k-1$  +  $n_k$ . Now what is this? I know it is strictly less than  $n/3$ . And what is  $n_k$ ?



(Refer Slide Time: 11:33)

$$n_k = \frac{1}{3}$$

$$A = \{1, \dots, k\}$$

$$B = \{k+1, \dots, l\}$$

$$\frac{n}{3} \leq \sum_{j=1}^{k-1} n_j + n_k \leq \frac{n}{3} + \frac{n}{3} \leq \frac{2n}{3}$$

$n_k$  is also upper bounded by  $n$  over 3, so this is at most  $2n$  by 3. And by choice of  $k$  what do we know? This is definitely this whole sum is at least  $n$  over 3. So, again we are able to prove that the sum of these here is the integers in index set  $A$  is at least  $n$  by 3 and at most  $2n$  by 3. What about  $B$ ?

(Refer Slide Time: 12:00)

$$\frac{n}{3} \leq \sum_{j=1}^{k-1} n_j + n_k \leq \frac{n}{3} + \frac{n}{3} \leq \frac{2n}{3}$$

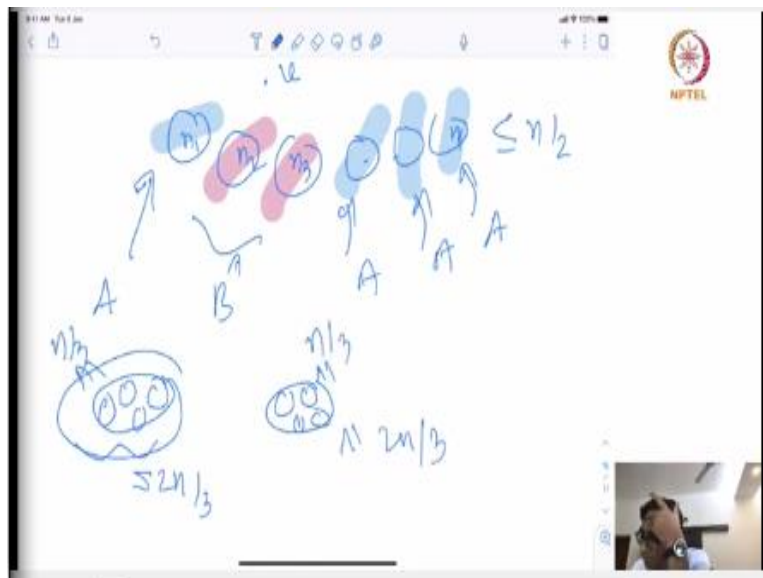
$$n - \left( \sum_{j=1}^k n_j \right) \leq \sum_{j=k+1}^l n_j \leq n \cdot \left( \frac{\sum_{j=1}^k n_j}{n} \right) \leq \frac{2n}{3}$$

Well, let us look at this, first is summation  $j$  going from  $k+1$  to  $l$   $n_j$ , what is this? Well, since the sum in  $A$  is at least  $n$  by 3, so it is  $n - \sum_{j=1}^k n_j$  which is at least  $n$  by 3, so this is at most  $2n$  by 3. Because this is at least  $n$  by 3, it could only be smaller; it cannot be

bigger than this. And now for the lower bound we will use the same thing, I mean again I can write down this is  $n - \sum_{j=1}^k n_j$ .

But what is  $\sum n_j$  upper bounded by is  $2n/3$ , so this will imply  $n/3$ . So, once you have the upper bound of  $n/3$ ,  $2n/3$  on A it automatically actually implies an upper bound for this. So, now with this we are done because what will I do? So, now in the branching step suppose let me do it for single vertex and then you will understand for others.

**(Refer Slide Time: 13:18)**

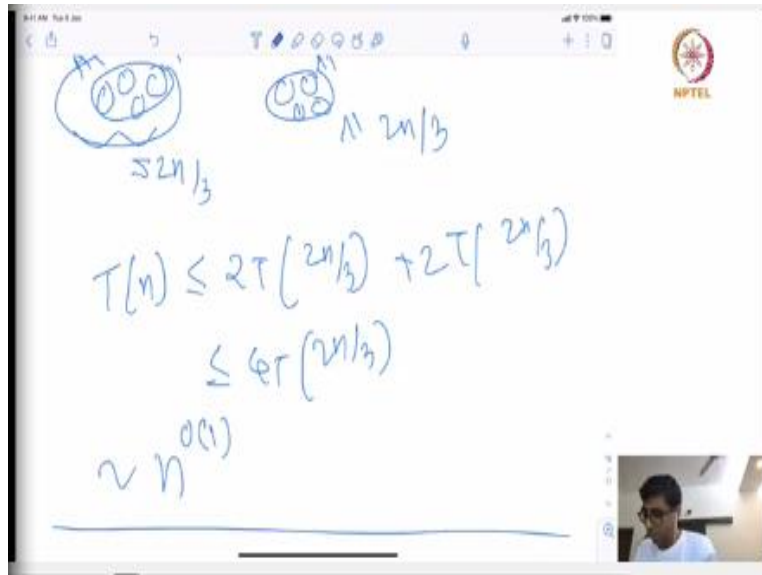


So, suppose I have a vertex  $v$ , so that every component has size  $n/2$ , so what will you do? You say ok fine, let us partition these components into say sum this is A part and say this is B part. So, this is like A, A, A based on the sizes and this is B. So, I will apply my previous lemma on these numbers. So, like suppose this has size  $n_1, n_2, n_3, \dots, n_k$ . So, based on the previous lemma, I know that now I can apply the previous lemma with sum being  $n - 1$  and I can partition this into 2 parts like there are this is some set of components B some set of component.

Such that what is the property? That here the size of the total size of the components is at most  $2n/3$  and at least  $n/3$ . So, this is at most  $2n/3$  and at least  $n/3$  at most  $2n/3$ , so this is a property. So, now rather than branching on like rather than making our algorithm work on each component separately which is the best way to do just for our analysis we will say ok let us combine these components in A let us call that left graph. Let us combine the components in the

index set of B, call them a B part. So, now and I am going to call my algorithm. So, if I delete V, if V is going to part of the solution, you have to also delete their neighbours. So, whatever more components you are going to get after deleting the neighbours of V into A, you are going to call them into 1 set of graph 1 graph and similarly in the B set.

**(Refer Slide Time: 15:20)**



The whiteboard shows a handwritten derivation of a recurrence relation. At the top, there are two small diagrams of a graph with three nodes and two edges, labeled  $5n/3$  and  $2n/3$ . Below these, the recurrence relation is written as:

$$T(n) \leq 2T\left(\frac{2n}{3}\right) + 2T\left(\frac{2n}{3}\right)$$

$$\leq 4T\left(\frac{2n}{3}\right)$$

Below the recurrence relation, the asymptotic complexity is written as:

$$\sim n^{O(1)}$$

So, now what will happen is that, you can show that T of n is like you are going to create 2 problems in each sides. So, it is going to be 2 times at most  $2n/3$  + 2 times at most  $2n/3$ , so this is like at most 4 times  $2T(n/3)$  and you can show that this is  $n$  to the power  $O(1)$ . So, now this time I have not hand weighed I have just given you an algorithm with a like upper bound running time which is still polynomial.

**(Refer Slide Time: 15:55)**

$$T(n) \leq 2T(n/2) + 1$$

$$\leq 4T(n/3)$$

$$\sim n^{O(1)}$$


---


$$T(n) \leq 2^{k+1} T(2n/3)$$

$$\sim n^{O(k)}$$

And, so if you do happen say for example, for the case I separated then you will get something like  $2k + 1, 2Tn \text{ over } 3$  and in my opinion this also should be something like  $n$  to the power  $O$  of  $k$ . So, we are not going to lose anything, so this is something which we should know and should utilize the fact. Now taught you all this, so remember from the last time what we said ourselves that our question like 2 question we asked.

Is weighted max independence at FPT parametrized by  $k$  on class  $k$ ? On what are the natural families of graphs that has a small separation number? So, let us try to answer the question number 2 first.

**(Refer Slide Time: 16:46)**

QUESTION:  
What is the  $\text{sep}_n(G)$  if  $G$  is a tree or union?

— (if we wish to prove  $\text{sep}_n(G) \leq k$   
→ for any induced subgraph  $G[X]$   
on  $X \subseteq V(G)$  has a balanced  
separator of size  $\leq k$ .

So, so, here is the question we would like to address. What is the separation number of  $G$  if  $G$  is a tree on  $n$  vertices? So, notice which basically if I say, so if we wish to prove that separation number of  $G$  is at most  $k$ , we should so that for any induced sub graph  $G$  of  $X$  on  $X$  subset of  $V$   $G$  has a balanced separator of size at most  $k$ , this is what we need to do.

(Refer Slide Time: 17:59)

— If we wish to prove  $sn(k) \leq k$   
 → for any induced subgraph  $G[X]$   
 on  $X \subseteq V(G)$  has a balanced  
 separator of size  $\leq k$

$\bigcirc \times \bigcirc$   $\nabla n \leq k$

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

Or rather have to find a set  $X$  of size at most  $k$  or in other words say that every component had size at most half, so this is what we had to show this.

(Refer Slide Time: 18:19)

$T, w: V(T) \rightarrow \mathbb{R}^+$

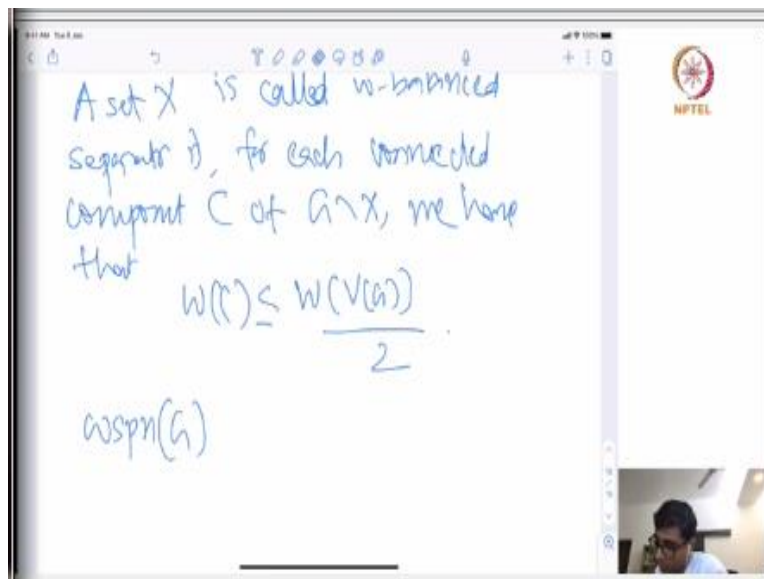
A set  $X$  is called  $w$ -balanced  
 separator if, for each connected  
 component  $C$  of  $G \setminus X$ , we have  
 that

$$w(C) \leq \frac{w(V(G))}{2}$$

So, let us we will try to show that separation number of a  $G$  and if  $G$  is a tree is at most 1. So, to prove this, what I am going to show to you is that rather than proving that separation number of

$G$  is at most 1, I am going to prove something more general. So, what I am going to show suppose I have not given a tree  $T$  and let us say we have a weights, some positive weights and I am going to find a notion of  $W$  balance separator. So, what is the  $w$  balanced separator? A set  $X$  is called  $w$  balanced separator if for each connected component  $C$  of  $G - X$ , we have that weight of  $C$  is at most weight of vertex set of  $G$  divided by 2. Now I am going to show, so this is like a let us call it a weighted separation number of  $G$ .

**(Refer Slide Time: 19:50)**



So, this is the definition of weighted separation number of  $C$ . So, we will rather let us leave it for now. So, I am going to prove to you that for any weight function  $w$ , I can find  $\alpha$  for if, so what I am going to show to you?

**(Refer Slide Time: 20:20)**

For any weight function  $w$ , a tree  $T$  has  $w$  balanced separator of size at most 1.  
 Then co-balanced separator of size at most 1.

$G$  has a balanced separator of size at most 1.

$w: V(T) \rightarrow \mathbb{R}^+$   
 $\forall v \in X, w(v) = 1$

For any weight function  $w$ , a tree  $T$  has  $w$  balanced separator of size at most 1. Now notice if we prove this why are we done? Well, that is very easy, because look at suppose you wanted to show that  $X$  has a balanced separator of size at most like graph induced on  $X$  at the balanced separator size 1, then what we need to show? What weight function do we choose? Then we will assign the following weight function we will take for  $V$  in  $X$  weight of  $V$  is 1.

**(Refer Slide Time: 21:24)**

$\forall v \in X, w(v) = 1$

$G$  has a balanced separator of size at most 1.

$w: V(T) \rightarrow \mathbb{R}^+$   
 $\forall v \in X, w(v) = 1$

$w(C) = |C \cap X|$   
 $w(C) = |C \cap X| \leq \frac{w(V(T))}{2}$   
 $\leq \frac{|X|}{2}$

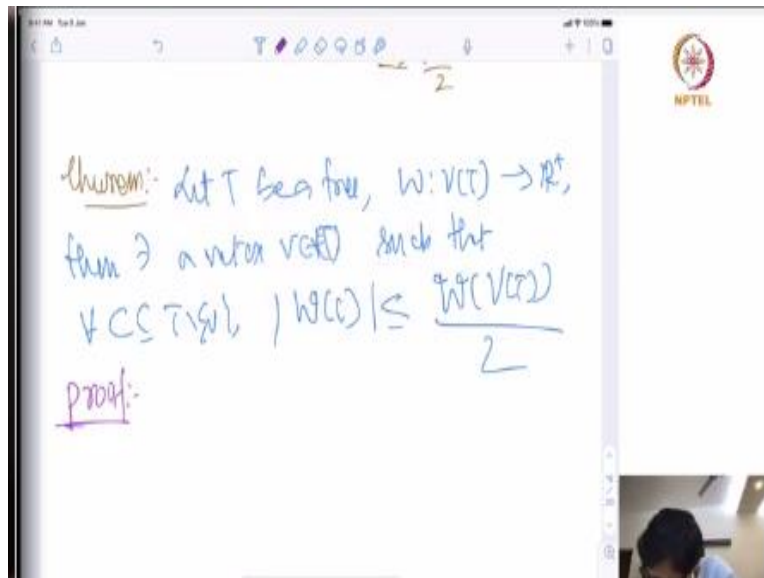
And  $V$  not in  $X$ , weight  $v$  is 0. So, notice what happens? So, then if we were able to find a vertex  $v$  and look at the component what is the property of this component? Well, what is the weight of this component? Weight of this component is nothing but  $C \cap X$  suppose some  $Y$  is there, so what is the component? It is  $C \cap X$ .

Because why it is  $C \cap X$ ? Because those are the only vertices which has some weight. And what is the property of this? Weight of  $C \cap X$  is at most weight of  $C$  which is at most weight of vertex set of  $G$  divided by 2 which is at most  $\frac{W(G)}{2}$ . Now if you delete this set of vertices, now, so if I delete this single vertex whatever the single vertex is that single vertex could belong to  $X$  could not belong to  $X$ , it does not matter.

If it does not belong to  $X$  then anyway if you look at the connected component and look at the intersection with  $X$  those are the connected components of  $X$  itself and they all have size at most half. It means basically what was there that the  $X$  which you gave me already had every connected component has size at most graph induced on  $X$  every connected component has size at most half  $X$  by 2.

Because if it connected then you have found a vertex on this such that this property holds. So, if we prove this property for any arbitrary weight function that given a tree arbitrary weight function that exist a weighted balance separator of size 1, then we have shown that the separation number of this graph is at most 1, so let us prove this.

**(Refer Slide Time: 23:46)**



$$\frac{W(G)}{2}$$

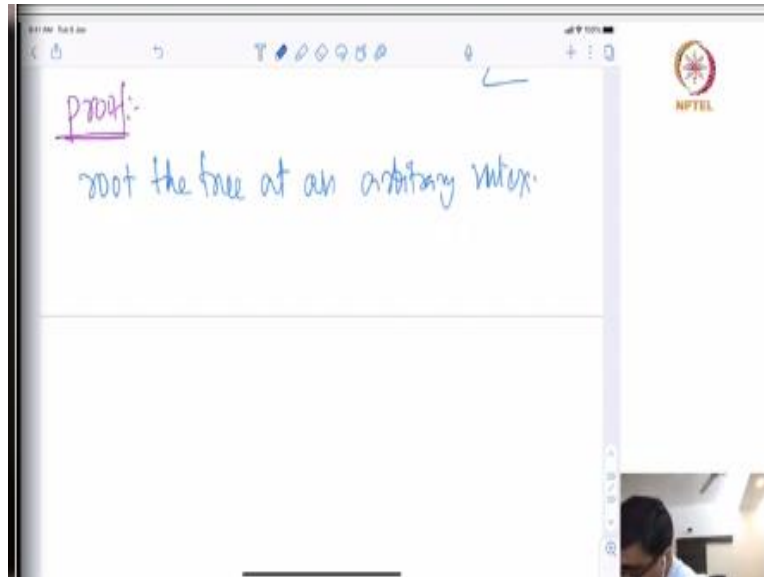
Theorem: Let  $T$  be a tree,  $w: V(T) \rightarrow \mathbb{R}^+$ ,  
 then  $\exists$  a vertex  $v \in V(T)$  such that  
 $\forall C \subseteq T, v \in C, |w(C)| \leq \frac{w(V(T))}{2}$

Proof:



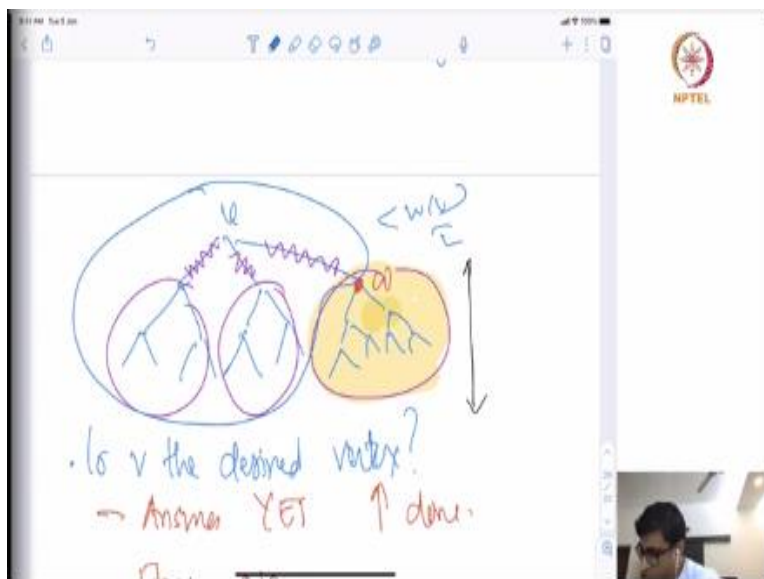
So, here is our theorem, let  $T$  be a tree  $w$  be a weight function from vertex set to  $A$ . Then there exist a vertex  $V$  in  $T$  such that for all  $C$  of  $T - V$ , weight of  $C$  is at most weight of vertex set of  $T$  divided by 2. Proof, what is our proof?

(Refer Slide Time: 24:35)



Just root the tree at an arbitrary vertex.

(Refer Slide Time: 24:56)



So, suppose here is a  $v$ , here is my tree and so I am going to ask at each step is  $V$  the desired vertex? Answer could be yes then we are done. Else we are going to else, no, it means what is the meaning of this?

(Refer Slide Time: 25:39)

Is  $v$  the desired vertex?

→ Answer YES ↑ done.

— Else NO

There is a component such that

$$w(C) > \frac{w(V(T))}{2}$$

There exist a component such that weight of that component is strictly greater than weight of vertex set of  $T$  divided by 2. So, if I delete  $V$  what are the components? Well, the components are going to be, what are the components if you delete? This is going to be one component. So, now as you have asked yourself is that, well, if this is not the case then there is a component here, say this is my component which is heavy. So, now I am going to ask is ok fine, let us call that. So, then it is like a think of it i, now what I will say?

**(Refer Slide Time: 26:50)**

Let  $w$  be the neighbour of  $v$  in  $C$  such that

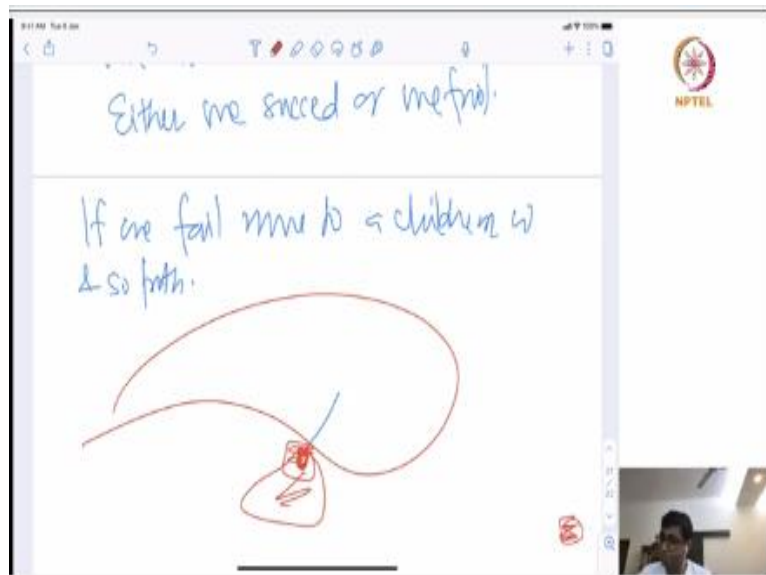
$$w(C) > \frac{w(V(T))}{2}$$

— Ask the same question but with  $w$ .

Either we succeed or we fail.

Let  $w$  be the neighbour of  $V$  in  $C$  such that weight of  $C$  is greater than equal to the weight of vertex. So, now I am going to ask,  $V$  is not a such a candidate, fine. Is here is my  $w$ . Now I ask the same question but with  $w$ . Now either we succeed or fail.

(Refer Slide Time: 27:50)

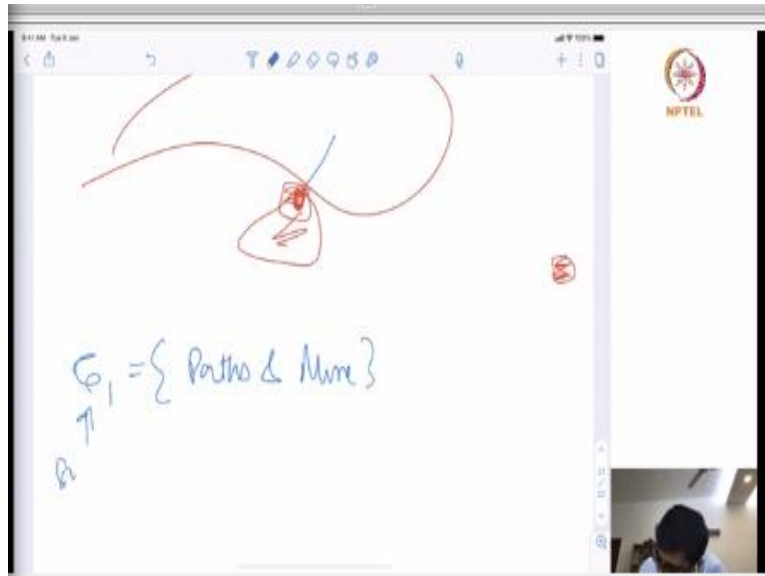


If we fail move to a children of  $w$  and so forth. So, you will keep updating, this is an algorithm actually, like it is an algorithmic proof. But what changes at each time the vertex which are considering it is distance from  $V$  increases, so it is a finite graph. So, after a finite  $(\infty)$  (28:18) this step you will find this vertex. Now, why? Because look at some, is it possible that you reach a leaf and both sides are you understand the question, because what happens?

So, the point is look at this component, look at the weight of this guy, because this is more than  $V$  over 2 weight of this is strictly less than weight of  $V$  over 2. So, if we go further look at the leaf maybe you reach a leaf. Why did you reach to this leaf? You reach to this leaf, because well, this is the guy such that whose weight is more than  $V$  over 2. Because there is no component here, which implies that if you delete this the component which is left has weight at most  $V$  over 2.

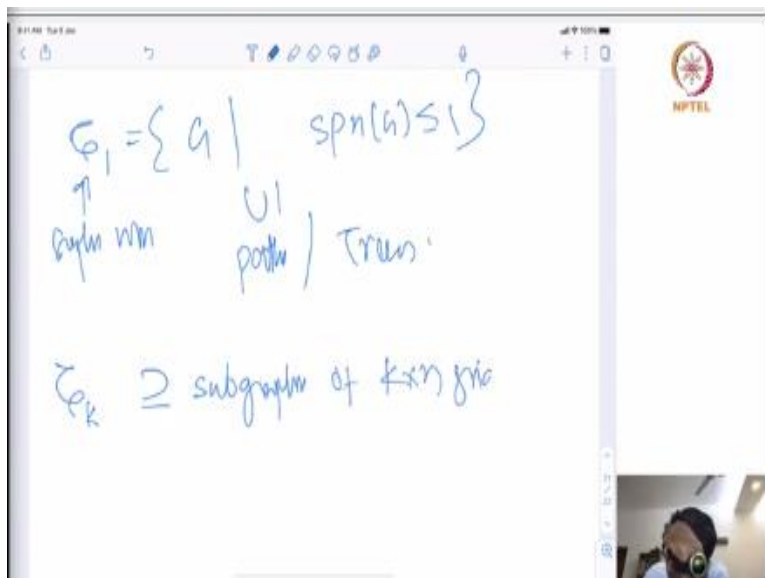
So, basically you can show yourself that if you move inductively in this way, you will find a vertex all the time in a tree. Such that if I delete this every component has size the total weight of the every component has is at most the total weight divided by. So, it is not very difficult proof you can also follow it yourself, if I was not very clear in the class. So, it is great that even for trees we can find a vertex such that every component has size at most half of this.

(Refer Slide Time: 30:34)



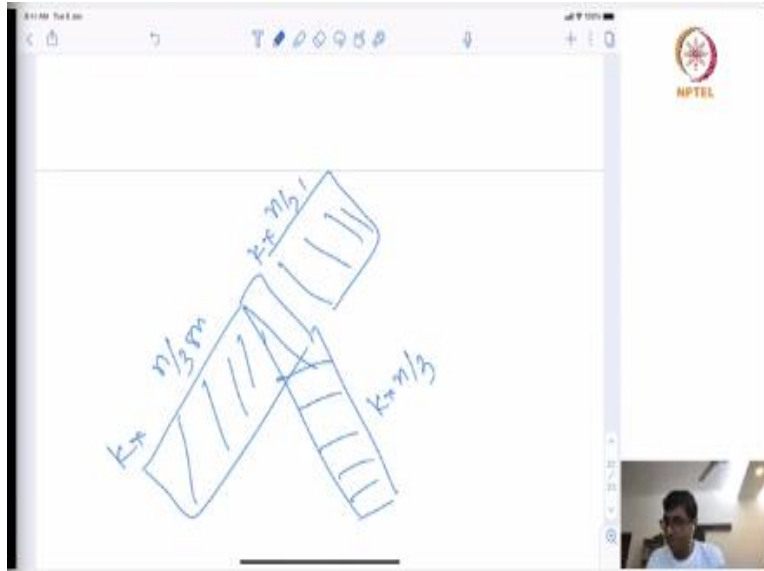
Now which implies that the up until now notice  $H_1$  was basically up until now contained paths and more or sorry let us call it  $Z_1$  graphs with rather.

(Refer Slide Time: 30:52)



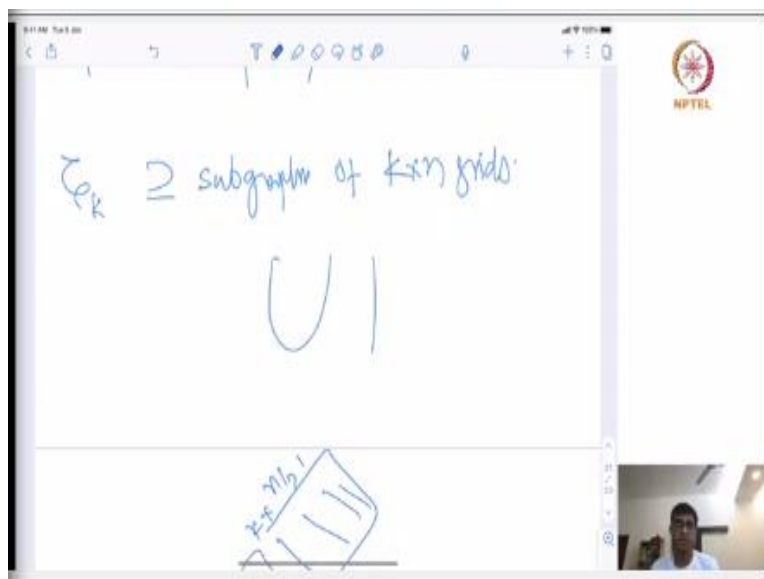
So,  $G$  and vertex separation  $V$  separation number of  $G$  is at most 1. So, now what we know? I know that this contains paths. Now, what does it contains? It contains trees at least this much  $v_1$ . And if you recall  $Z_k$  we know that it contained sub graphs of  $k$  times  $n$  grids.

(Refer Slide Time: 31:35)



In fact notice what else can it contain? It does not have to be, so for example look at this is my right and this is suppose this is  $k$  cross  $n$  by  $3$  grid. And like this part and this is another  $k$  cross  $n$  by  $3$  grid and this is another  $k$  cross  $n$  by  $3$  grid. So, this is not a grid like any you notice that any sub family sub graph of even such structure has at most  $k$ , so this is like a tree. So, if you think of this  $k$  cross  $n$  grid as like a fat path, I have tried to make a fat trees in some sense.

**(Refer Slide Time: 32:22)**

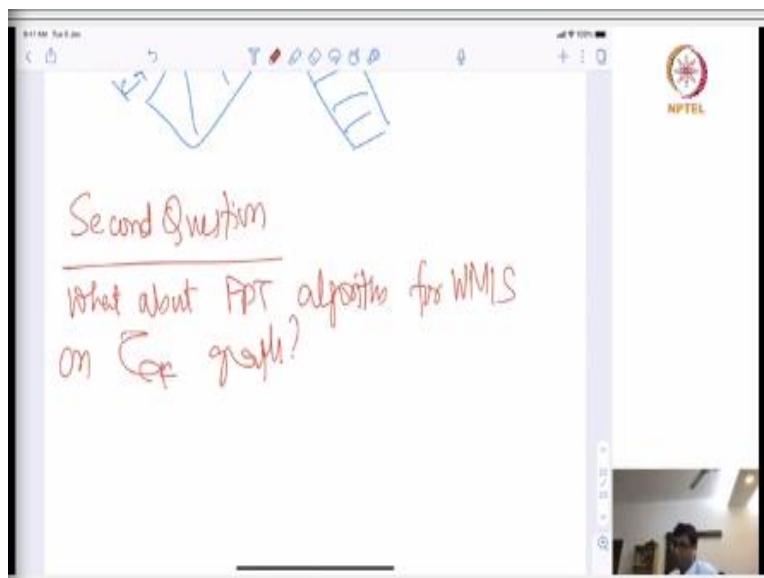


So, even for such objects these guys which will start belonging to the graph  $Z_k$ . So, this is what exactly wanted. So, now, because by proving this nice theorem about trees, we have been able to say to you or show to you that separation in a graph. And separation is very simple property of a graph. What is a simple property? That a graph has a separation number at most  $k$  if for every

induced sub graph, I can delete  $k$  vertices such that every component has size half of what I started with.

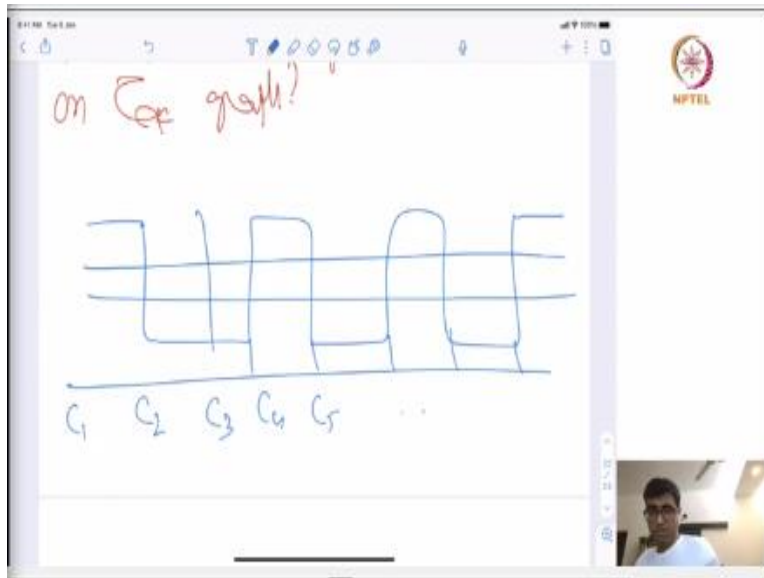
If this is the case then the graph had separation number at most  $k$  and for those families of graphs, for those classes of graph, maximum weight independent set can be solved in  $n$  to the power  $k$  time. So, there are lots and lots of graphs where such properties would hold. You should try to make some examples for yourself. But the second question that we wanted to resolve or solve last time was of following.

**(Refer Slide Time: 33:22)**



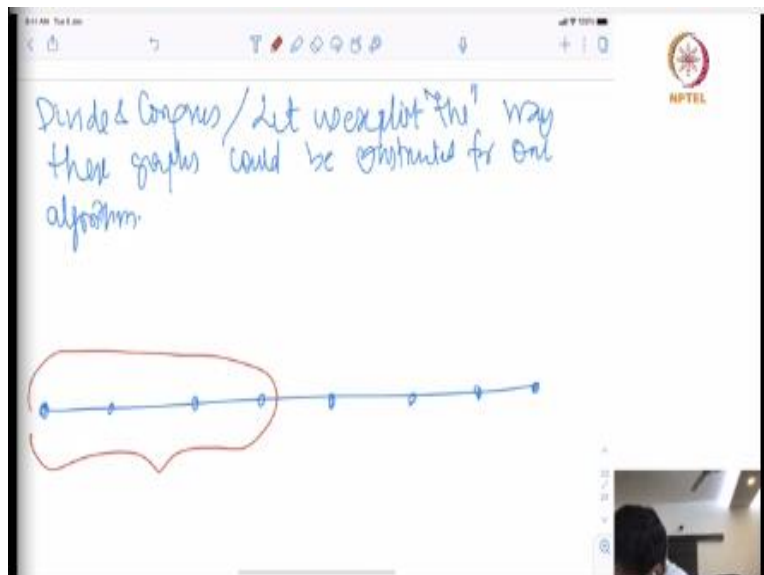
Second question. What about FPT algorithm for say weighted max independent set on graphs? This is a very basic question here. So, before we answer this question let us try to do the how can we do this things if for say grid graph?

**(Refer Slide Time: 34:09)**



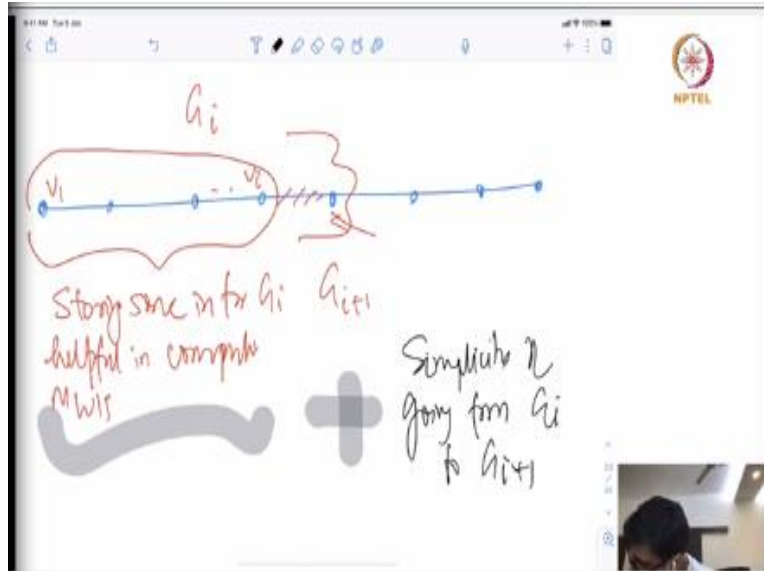
So, suppose we have this time I will try to draw, this is also some subclasses of grids. Say, this is column 1, column 2, column 3, column 4, column 5 so on and so forth maybe let us try to do it for path itself. So, we will try to do this a little bit later, let us do it for path itself.

**(Refer Slide Time: 34:45)**



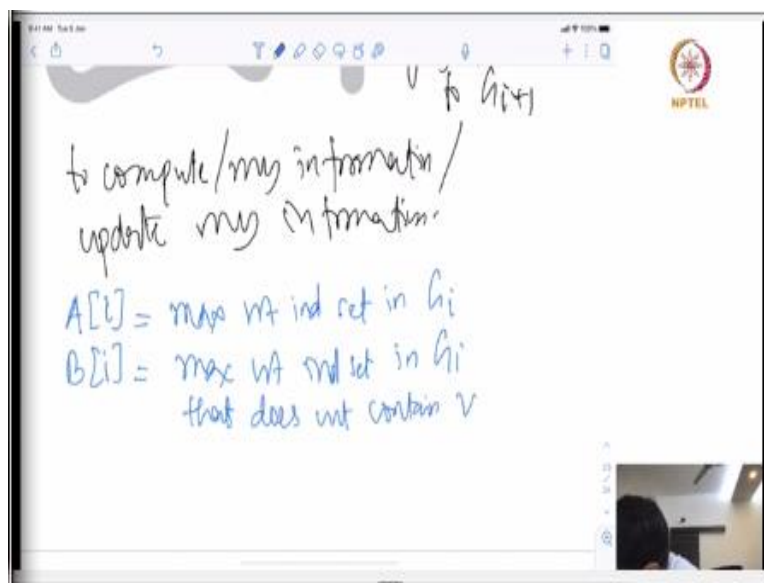
So, rather than doing divide and conquer, let us exploit more algorithmic. Let us exploit the V these graphs could be constructed for our algorithm; it will become very clear soon. So, the idea will be is that look suppose somehow I knew or I am storing some information storing some information for this graph.

**(Refer Slide Time: 35:53)**



Say  $G_i$  which consists of say first  $i$  vertices. Then can I construct like store some information for  $G_i$  helpful in computing maximum weight independent set. So, from looking at this I can tell you this is the max weight independent set. So, I want to store some information and then the next question fine, can exploit the fact that look at the graph between  $G_i$  and  $G_{i+1}$ . This is like not much first of all this is the vertex which only connects to just 1 vertex of my previous graph  $G_i$  and that also with some edge. Can I use this information? Plus simplicity of going from  $G_i$  to  $G_{i+1}$  to compute my information, rather update my information.

**(Refer Slide Time: 37:11)**



So, if I can do this then from  $G_{i+1}$ , I will go to  $G_{i+2}$ , which will use the information stored and the simplicity to get the information. So, there are 2 things first of all what does it mean to be

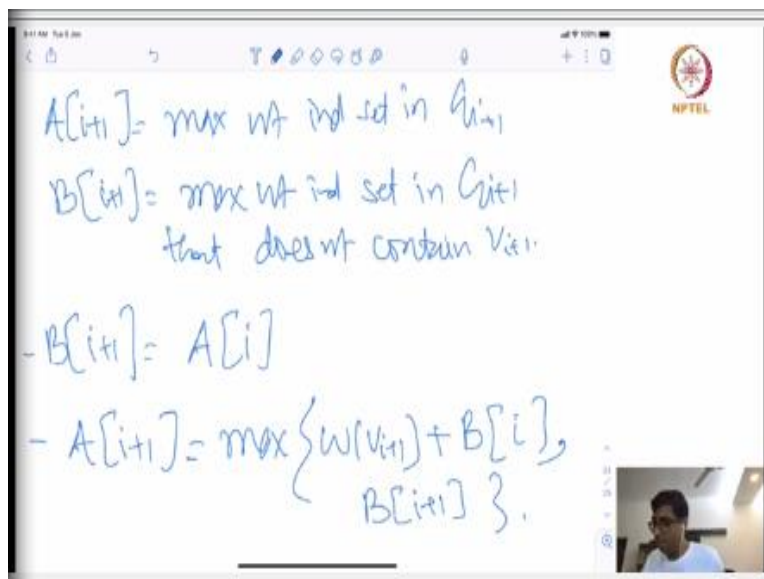


simply sitting of this graph how do we construct the simplicity of this graph? That is one thing or define such simplicity of graph? And secondly is what information do I need to store?

So, for example, I am going to store the following information. So, the for max independent set I am going to store information say  $A_i$  and  $B_i$ . So what is this? This is going to store, this is nothing but max weight independent set in  $G_i$ . And  $B_i$  is max weight independent set in  $G_i$  that does not contain  $V_i$ . So, basically I am saying I am going to store 2 information because and from where does the information come.

So, if you look at your divide and conquer algorithm what are we branching on? We were branching on either my solution contains this independent set, this vertex in the independence or it does not. So, in some sense, that is what rather than doing branching like or divide and conquer and then getting those solutions, we are going to like store these solutions into a table and going to update this table as like we go forward. So, let us try to understand. So, now let us look at suppose I have this information for  $A_i$ .

**(Refer Slide Time: 39:32)**



The image shows a digital whiteboard with handwritten notes in blue ink. The notes define two arrays,  $A$  and  $B$ , for a dynamic programming algorithm. The top part defines  $A[i+1]$  and  $B[i+1]$ . The bottom part shows the recurrence relations for  $B[i+1]$  and  $A[i+1]$ . The NPTEL logo is visible in the top right corner. A small video inset of a person is in the bottom right corner.

$$A[i+1] = \text{max wt ind set in } G_{i+1}$$
$$B[i+1] = \text{max wt ind set in } G_{i+1} \text{ that does not contain } V_{i+1}$$
$$- B[i+1] = A[i]$$
$$- A[i+1] = \max \{ w(V_{i+1}) + B[i], B[i+1] \}$$

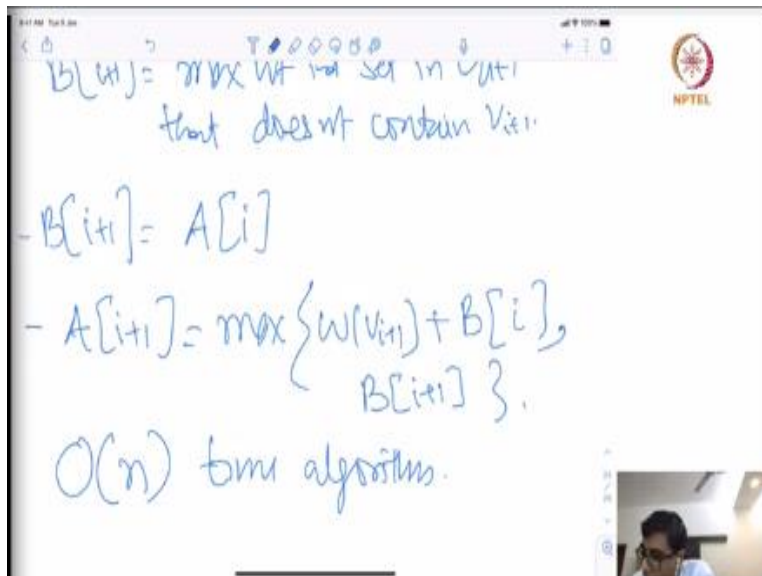
So, now how we are going to do  $A_{i+1}$ ? What is  $A_{i+1}$ ? It is a max weight independent set in, so this is what it needs max weight independent set in  $G_{i+1}$ , this is what you would like to update. And what is  $B_{i+1}$ ? It is max weight independent set in  $G_{i+1}$ , that does not contain  $V$

$i + 1$ . Now what is  $B_{i+1}$ ? Notice, so, if this is going to be the maximum weight independent of  $G_{i+1}$  that does not contain  $V_{i+1}$ .

Then that must be we have max weight independent set of  $G_i$  and that we have already stored in  $A_i$ . So  $B_{i+1}$  is nothing but  $A_i$ , good. So, now what is  $A_{i+1}$ ? Either it contains  $V_{i+1}$  or it does not contain  $V_{i+1}$ . If it contains  $V_{i+1}$ , then it cannot contain  $V_i$  also, which implies that information is nothing but it is a max independent set of  $G_i$ , that does not contain  $V_i$  which is stored in  $B_i$ . So,  $A_{i+1}$  is max.

So, if it is going to contain  $V_{i+1}$ , then it is weight of  $V_{i+1}$ , + say  $B_i$ , so this is one. Or it going to contain or it does not contain  $V_{i+1}$  and then we already have updated  $V_{i+1}$ , so it is there. So, we are done, so we have been able to update this information using this.

**(Refer Slide Time: 41:37)**



$B[i] = \text{max wt ind set in } G[i] \text{ that does not contain } V[i]$

$B[i+1] = A[i]$

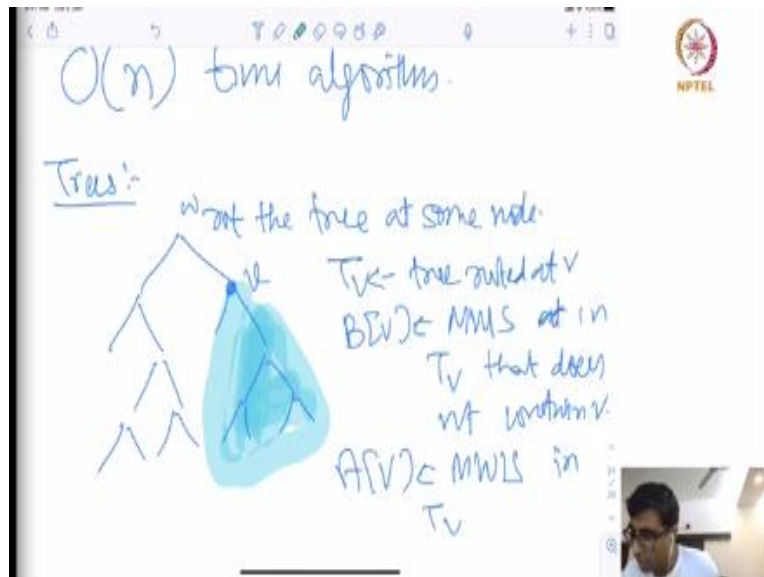
$A[i+1] = \max \{ w(V_{i+1}) + B[i], B[i+1] \}$

$O(n)$  time algorithm.

And now notice this is like a linear time algorithm. So, we are just maintaining 2 cells and go from left to right and you are able to construct such objects. Let us ask ourselves, can we do the same thing for the trees? Because here there was a path, there was a notion of left to right,. But, so we were able to exploit that, how will we exploit such kind of like left to right? So, what are we left with?

So, this is some way we are constructing this graph and we are saying ok, follow this construction of this graph and update your information. So, now for trees, what can we do? So, let us look at the tree, so let us do the same thing with the trees.

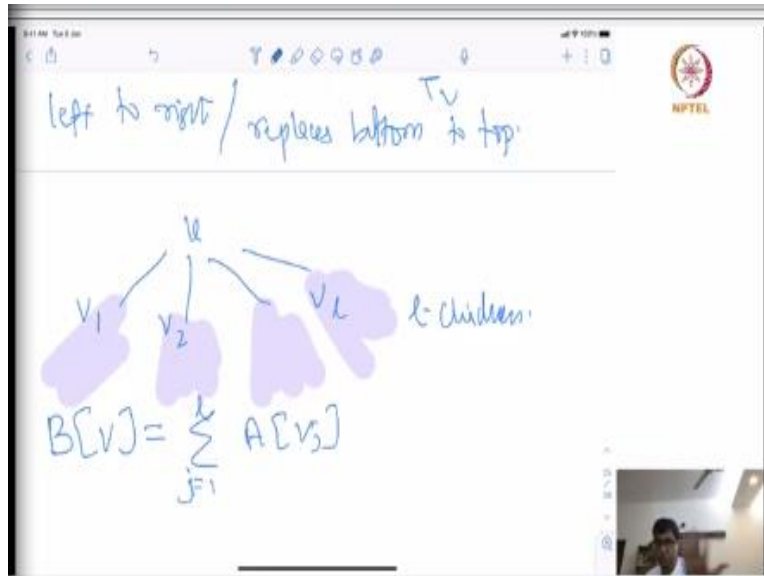
**(Refer Slide Time: 42:34)**



Let us root the tree at someplace, at some node, let us say  $w$ . And now let us look at a vertex  $v$  by  $T_v$ , that is mean tree rotate at  $v$ . And  $B_v$  will be again max weight independent set in  $T_v$  that does not contain  $v$ . And what is  $A_v$  max weight independent set in  $T_v$ ? So, same to this like, so basically, this is what. So, notice that because we have rooted this tree, now we do have a notion of where to look?

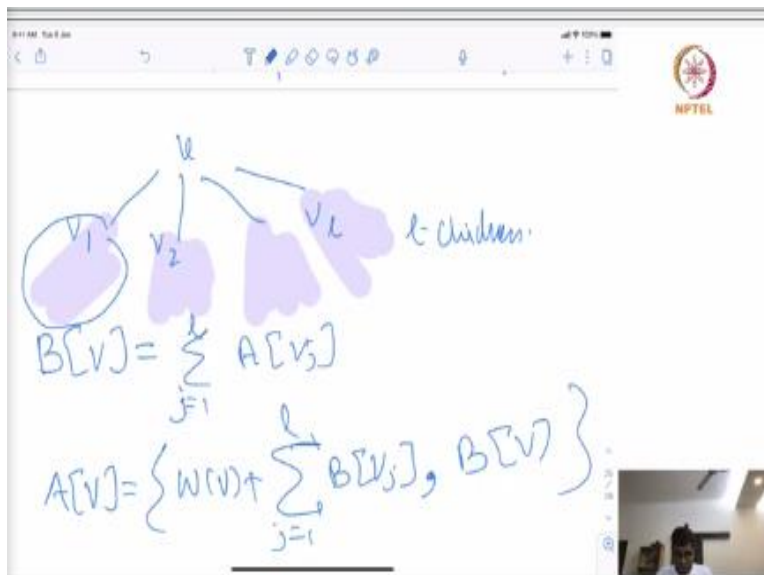
I said fine, so this is. And we are what will we do? So, we were doing left to right. But now it is a tree and say root a tree. So, we will do from bottom to top, so first we will take care of leaf then their parent and then their parent and their parent, so.

**(Refer Slide Time: 43:43)**



Left to right replaces bottom to top, great. So, suppose we are at some vertex  $v$ . And here the  $v_1, v_2, \dots, v_l$ , there are  $l$  children. And now, again we will do what is  $B$  of  $v$ ? It is a maximum weight independent set that does not contain  $v$ , it means look at then what is the maximum weight independent set? That does not contain  $v$ , it means It should be maximum in the sub tree here, it should be the whatever it is. So,  $B$  of  $v$  is very simple, it is nothing but summation  $j$  going from 1 to  $l$   $A[v_j]$  which we have already stored, so that is great. And now what is  $A$  of  $V$ ?

**(Refer Slide Time: 45:03)**

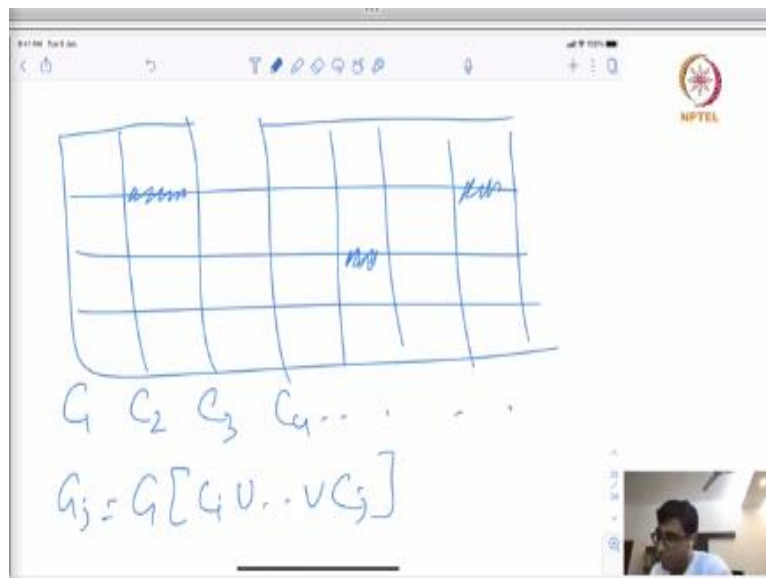


Ok fine, you are probably going to contain  $V$  in your independent set, if you are going to contain  $v$  in the independent set, then you cannot contain  $v$  want to be  $l$ , it means in this sub part of the tree you are looking for and a max weight independent set, that does not contain the  $v_1$  that we

have already computed. So, what is  $A$  of  $v$  is nothing but weight of  $v$  + summation  $j$  going from 1 to  $l B v j$  that is correct. Or you do not contain  $v$ , if you are not going to contain  $v$ , then that is like  $B$  of  $v$  which we have already computed. So, this is how we can update this information. So, it does not have to have a left to right but even top to bottom.

I notice that, top to bottom generalizes the idea of left to right, because I can think of path also the tree rooted at  $V_n$  and then go from bottom to top. So, the moment you make  $V_n$  as your root left to right replaces from bottom to top. And so this idea gets generalized. Now, so leaves and trees are very beautiful but now what about this grids or sub classes of grids.

**(Refer Slide Time: 46:29)**

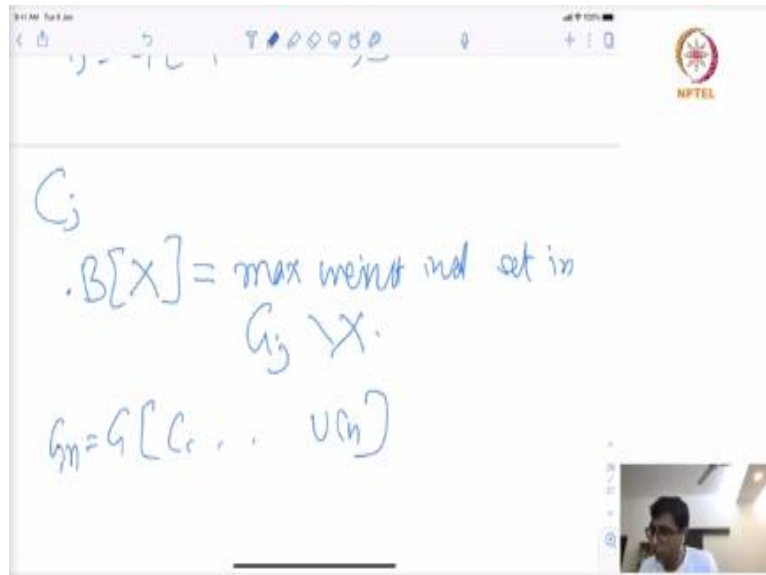


So, let us say never mind if my drawings are not that beautiful. The way we have constructed path we could construct the grid at such like column by column. So, first we construct column 1, column 2, column 3, column 4 dot, dot, dot, dot, dot. And what will be the graph  $G_j$  is basically graph induced on column 1 union dot, dot, dot union column  $j$ , so first  $j$  graph induced on this.

So, I mean if you really want to make life nicer, maybe like difficult or something think of them as a sub graph like some edges are missing. For example, just it does not matter for our purposes. But like this is missing, they are like some. And we can go from again as before we could go from left to right and whatever we deciding.

So, notice what is  $B[V_j]$  storing?  $B[V_j]$  are storing, what is the max weight independent set that does not contain some vertex. So, now we will generalize so look at the column  $C_j$ .

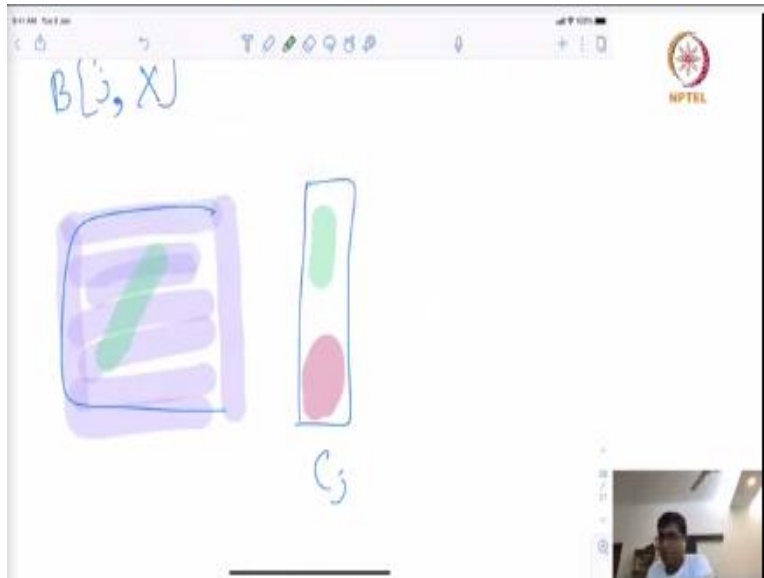
**(Refer Slide Time: 47:48)**



And we are going to say  $B$  of  $X$ , what is going to be  $B$  of  $X$ ? It is going to store max weight independent set or maximum weight of an independent set rather not. We are going to store max weight independent set in, so you are not going to contain. So, in the graph  $G_j - X$ , this is what you are going to store. So, what is a maximum weight independent set in  $G_j - X$  **ok** and that is it.

So, look at this, I can find the maximum weight independent set of my whole graph  $G$  by looking at  $G_n$ , which is nothing but  $G[C_1 \cup C_n]$  column. And what will I do here So, then I know that either the max weight independent set has empty intersection or non empty intersection and based on that we can do everything. Now the question is, how are we so rather than writing this maybe we can write  $j, X$  is like, what is the max weight possible independent set in  $G_j - X$ ?

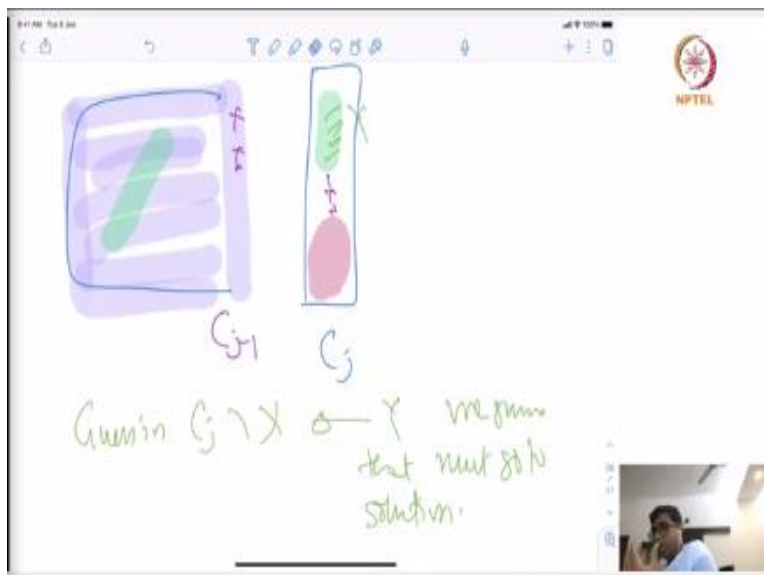
**(Refer Slide Time: 49:15)**



So, let us how can we  $B_j, X$ , so let us try to understand what happens? So, here is my a piece of graph for which all these information are there, this part of the graph. And now we have this same set  $X$  or no, sorry,  $C_j$  and here is some set  $X$  that is not going to be part of the solution. It means the remaining has to be part of the solution. So, what I am saying that what is the max weight possible independent set in  $G_j - Y$ ?

So, look at the maximum weight independent set that could consist of what? So, notice that could consist of some set here and some set here. I do not know which set it is. So, what generally we do is that we guess.

**(Refer Slide Time: 50:30)**



So, we will guess in  $C_j - X$ , let us call that  $Y$ , so we guessed that must go to the solution. So, if this is the  $Y$  which goes to the solution, what do we know? It is neighbors cannot go to the solution and this is where we use the fact that these graphs are constructed very well. In the sense where are the neighbours of  $Y$ ? Neighbours have  $Y$  are either there in  $C_{j-1}$  or here this is where the neighbours of these vertices in  $Y$  are. So, if you pick up  $Y$  in your solution you are forgetting which one? You are forgetting all the neighbours of  $Y$ . So, what is that?

**(Refer Slide Time: 51:39)**

The image shows a digital whiteboard with handwritten text in blue ink. At the top, there is a header bar with some icons and the text 'T 000000'. The main content is as follows:

$$B[j, X] = \max_{\substack{Y \subseteq G \setminus X \\ Y \text{ is an independent set}}} \left\{ \begin{array}{l} W(Y) + \\ B[j-1, N(Y) \cap G \setminus X] \end{array} \right.$$

Below this, it says:

Choices of  $Y$ :  $2^{k-|X|}$

Time to fill table takes  $2^{k-|X|}$ .

So, now I can write down this recurrence. So, the recurrence is notice  $B_j X = \max Y$  subset of  $C_j - X$ , that is a first,  $Y$  is an independent set otherwise I mean you can never complete it one independent set. And the moment you have what is this. So, it is like a weight of  $Y$  so now you know that you are not going to pick anything all are it is neighbours. So, and you are also not going to pick anything else from the column  $C_j$ , so everything else is from the induced graph  $C_{j-1}$  and from which you are not going to pick up the  $N$  of  $Y$ .

So, basically you are forbidding  $N$  of  $Y$ , so basically this is nothing but where it is stored? It is stored in  $B_{j-1}$ ,  $N$  of  $Y$  intersection  $C_{j-1}$ , that is it. So, how much time will it take to fill this table? Notice, so choices of  $Y$  is how much? To fill this table, what is the choices of  $Y$ ? So, the choice of  $Y$  is like  $2^{\text{power } k - \text{cardinality of } X}$  and once we have this, it is like so we have so many choices and then you have some  $2^{\text{power } k - X}$  numbers. And from them you choose the



maximum weight. So, to fill this table, it is going to take time to fill table takes  $2^{\text{power } k - \text{mod } X}$ .

**(Refer Slide Time: 53:41)**

$$\sum_{i=0}^k \binom{k}{i} 2^{k-i}$$

$$= 3^k \cdot n^{O(1)}$$

$3^k \cdot n^{O(1)}$  is FPT on subclasses of  $k$  and  $n$  grids

So, how much overall time does it take to fill? So, it is summation, so choices of  $X$  which is like  $k$  choose  $i$ ,  $i$  going from 0 to  $K$ . And for each of them I am going to do this, so which is nothing but 2 to the power  $k - i$ . So, by binomial this is going to be some 3 power  $k$  and  $n$  to the power  $O$  of 1. So, actually you can solve all these things in keep our  $k$  running time which is FPT. Now notice that in all these things, we had this notion of graphs.

So, small graph, then the next we are going to graph but the interaction between like the new set of introduce vertices with the previously constructed graph is limited. So, this is the limit which will help us to construct the definition and the sequence of the graphs you are going to form. And that sequence of the graph is what will allow us to give a decomposition. As well as to come up with a dynamic programming algorithm the one which we have constructed. So, now what we have shown now? So, all we have shown that max independent set is FPT on subclasses of  $k$  times  $n$  grids.

**(Refer Slide Time: 55:26)**

choosing  $x$

$$= 3^k \cdot n^{O(1)}$$

$3^k \cdot n^{O(1)}$  is FPT on subchain from kids.

---

Is MIS FPT on  $\mathbb{Z}_k$ ?

→ we still don't know!

But Is max independence set FPT on? We still do not know; we will take up the thread in the next lecture, starting from here.