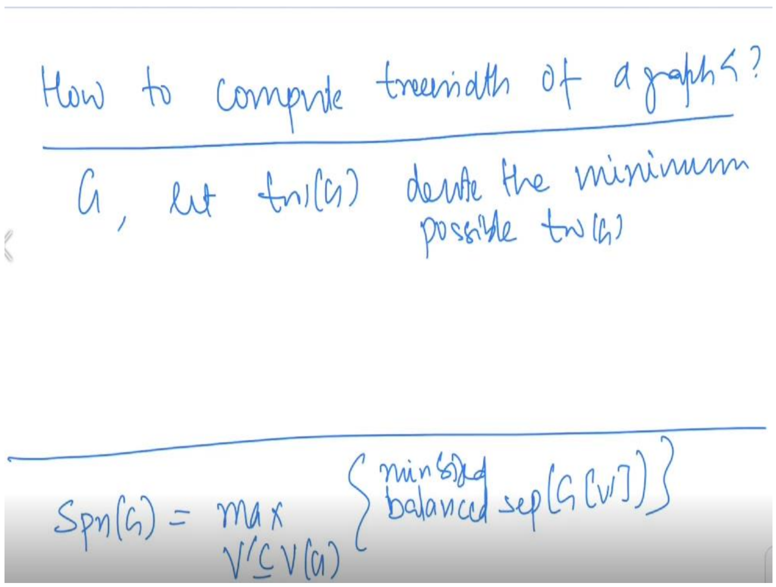
**Parameterized Algorithms**
**Prof. Neeldhara Misra**
**Prof. Saket Saurabh**
**The Institute of Mathematical Science**
**Indian Institute of Technology, Gandhinagar**

**Lecture - 27**
**FPT Approximation Algorithm for Computing Tree Decomposition-Part 01**

**(Refer Slide Time: 00:15)**




Welcome to the third lecture of week 6. Up until now we have seen that like how we can apply do dynamic programming over graphs of boundary tree width. But, in this lecture we will see how to actually compute this. So, this is about, how to compute tree width of a graph G, so this is what, this so given a graph G let t w G denote the minimum trade worth of it or the minimum possible. So, there are several question that could arise here.

**(Refer Slide Time: 02:00)**

TREEWIDTH

Input: G, k

parameter: k

Question: Is $tw(G) \leq k$ ?

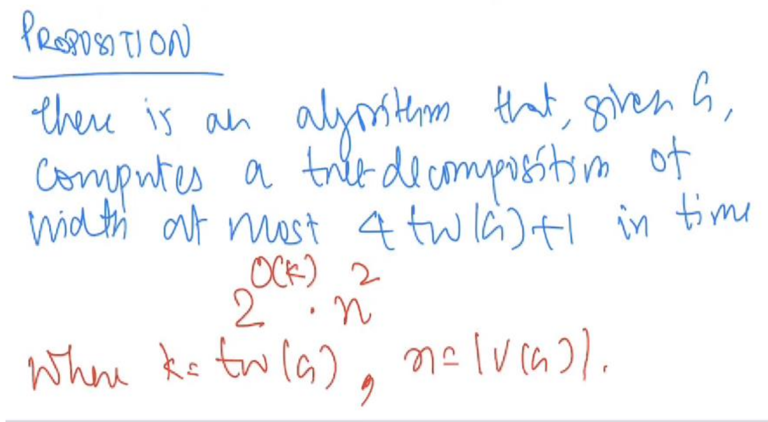
Bodlaender

$2^{O(k^3 \log k)} \cdot n$ [linear time]

So, let me so I have maybe. So, let us so what are the two questions we could ask. So, we could ask the decision version of the graph problem could be, the decision version of the question could be so input is G, k my parameter is k and the question is tree width of G at most k? Now, this problem was known to be FPT already from 1980s. But the first explicit algorithm was given by Bodlaender, in by Hans Bodlaender.

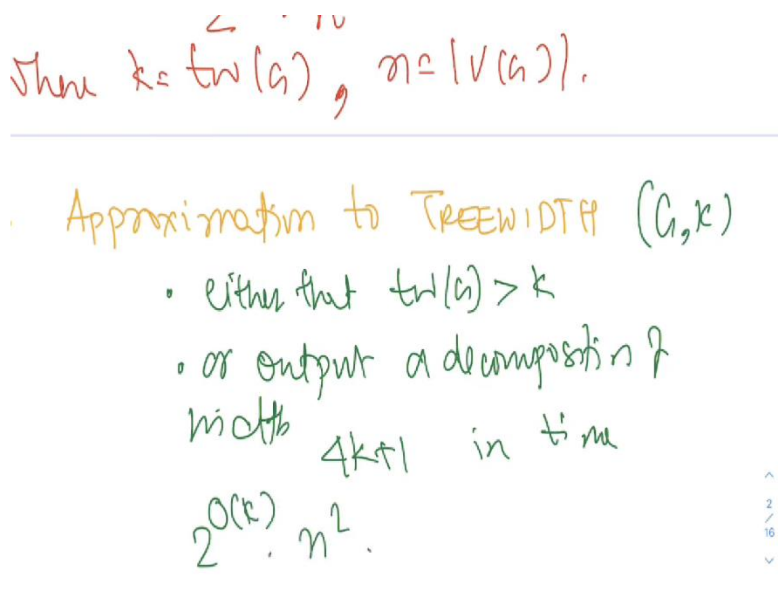
It was given first explicit algorithm, was given by Hans Bodlaender who showed that design an algorithm for tree width running in time if I k cube log k times n. So, that also, it was also, shows gave us the first linear time, because there is a linear dependence on the input size. But this algorithm is quite complicated and we will not be doing in this course, but rather we will be giving an approximation for tree width as a parameter. So, what algorithm we are going to present in this lecture today is the following.

**(Refer Slide Time: 03:50)**




So, that the proposition we are going to show today or the theorem we are going to prove so there is an algorithm that given G, given G and rather given G and in rather let us say given G computes a tree decomposition. Tree decomposition of width at most four times tree width of G + 1 in time n square where k equal to tree width of G and n equal to so, basically if you think of an approximation then it is like if I have given an integer k.

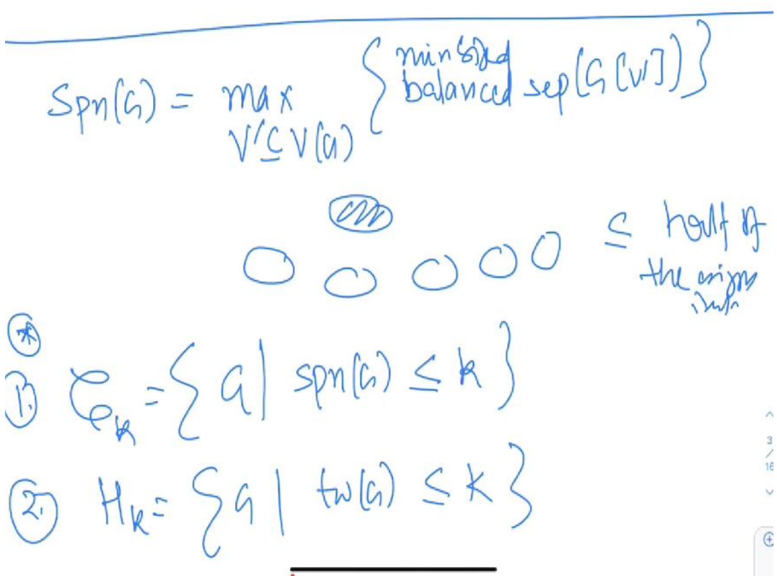
**(Refer Slide Time: 05:22)**




So, if you just think from the perspective of approximation then what it means actually is that we can convert this algorithm into as follows. So, what we can convert this algorithm into is as follows. So, basically tells that look I mean, if you could have run this algorithm then for. So, I will give you an algorithm for this nature. So, and using this algorithm actually we can, actually come up with the following approximation to tree width.

What is this? So, approximation to the problem tree width. So, this will output you, either that so you are given a G and k, then say either it will say the tree width of G is more than k or, rather strictly more than k or output or decomposition of width 4k + 1 in time. So, this is what we will do. Now so that is our whole idea.

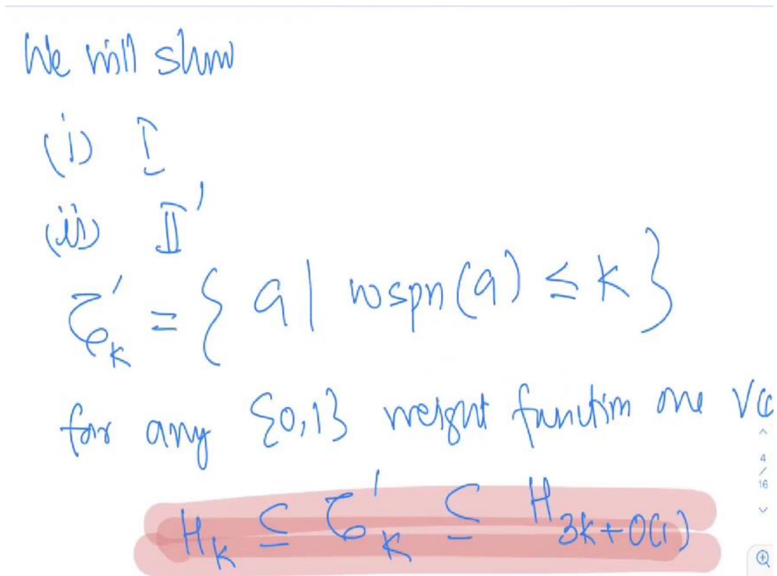
**(Refer Slide Time: 07:00)**




But like remember in the last lecture we talked about this notion of separation number. So, what was the separation number? So, separation number of a graph was like for a particular graph for like min size balanced everything but like we will try to do. And then we said that look what is the minimum? What is your, if I say what is the meaning of the separation number of graph G is at most k if I give you any induced sub graph of G.

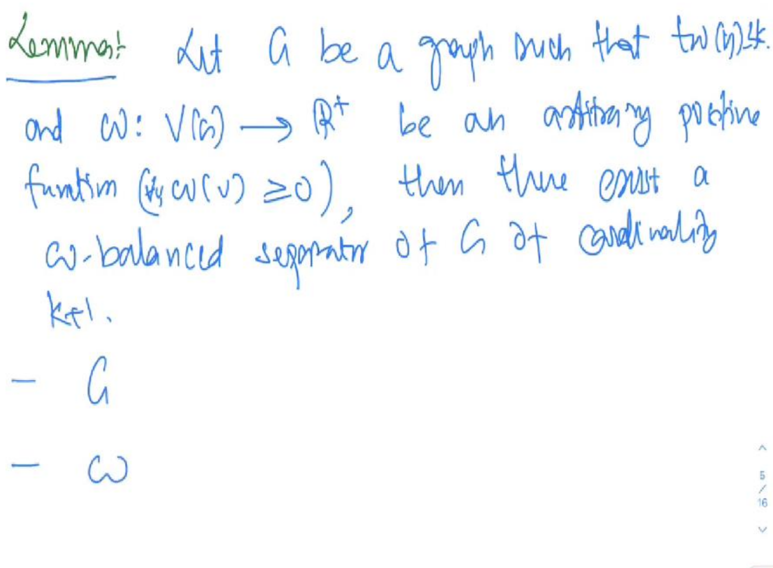
Then I can find a k size vertex, whose deletion will have property that every connected component has size at most half of the original graph. And, we wanted to relate between z k and H k and but we say that.

**(Refer Slide Time: 07:53)**




It will not happen that way but it will happen if we have some notion of weighted balance separator.

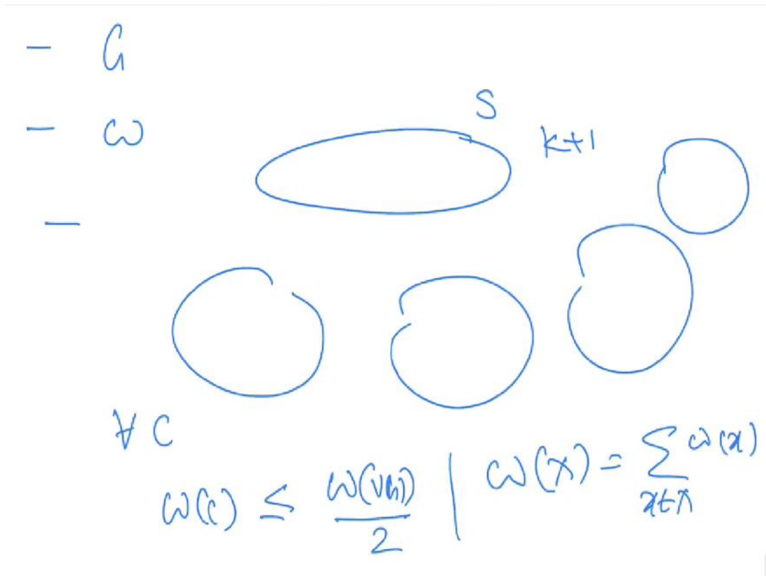
**(Refer Slide Time: 08:04)**




So, today's lecture let us try to do our first thing. So, this is a lemma that like would like to prove and that will show something interesting. Let G be a tree width let G be a graph let G be a graph such that tree width of G is at most k and W V and arbitrary weight function, arbitrary positive weight function, meaning weight of every for, every V, weight of V is greater than equal to 0. So, it is not negative arbitrary positive weight function.

Then there exist, let us say W balanced separator of G of cardinality k + 1. So, what is this tells us? It tells us that look you are giving me a graph G and there is a weight function W.

**(Refer Slide Time: 09:50)**

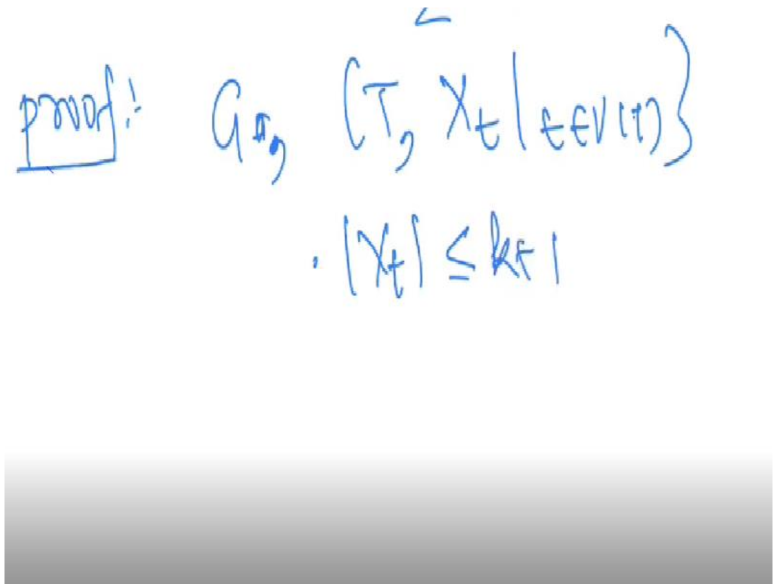



So, then I will find a W balance separator of G of cardinality k + 1, meaning I will find you a k + 1 sized set say S. So, that if you look at the connected component, what is the property of connected component? For every connected component weight of C is less than equal to weight of G divided by 2 where weight of G or weight of I should say weight of, and what is the weight of any set x? It is just the natural definition, weight of x is like x in x weight of x.

So, if you have a set x you just sum up the weights of each vertices. So, but basically this is why it is called weighted balance separator that and now notice that if you have got this at most half of the original graph kind of definition if we just would have put weights of every vertex to be 1. So, if the W is the weighted function which assigns every vertex 1, then would have had satisfy this property.

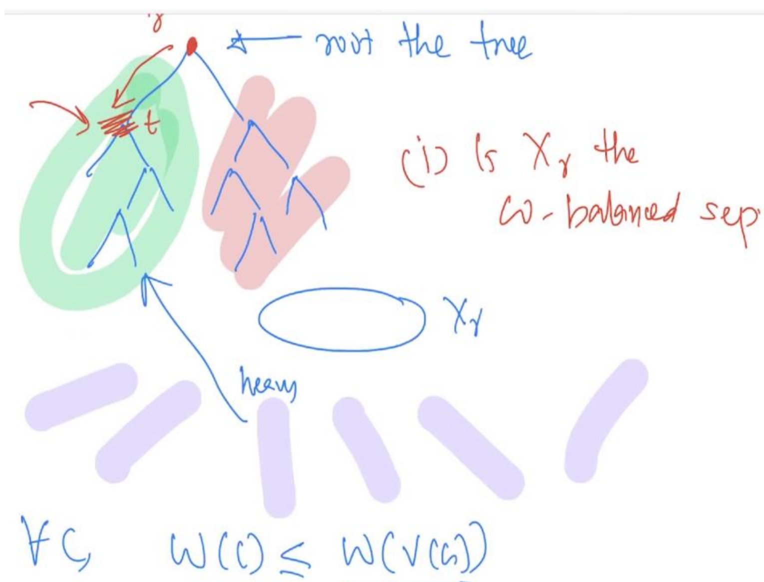
So, notice if a graph has tree width at most k then in fact it has it has a weighted balance separator for any positive weight function. So, that is interesting so not only it has so, not only what does it shows, that not only H k belongs to z k which was like graph with balance separator of size at most k for every connected component. But in fact it is like, I mean it is among the graph, like it is a graph with a amazing weight functions like any weight function separator that it belongs to.

**(Refer Slide Time: 11:36)**




So, how we going to prove this, the proof is very simple and it is if you recall correctly, we proved a balance separator for trees and it is going to be just exactly the same. So, now we have a graph G. So, we have a tree decomposition T, and some X t, t in v t. And what is the size of any X t, is at most k + 1. Because tree width of graph is at most k.

**(Refer Slide Time: 12:15)**



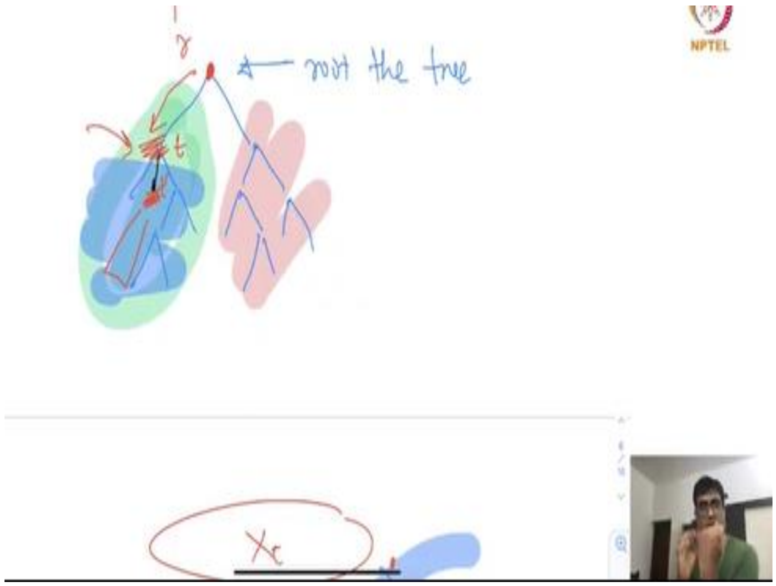

Now, what we are going to do you fix tree decomposition and you root the tree. And you ask yourself, so this is what we have to. So, this is my root and now I ask myself the following question. Is X r the weighted balanced separator? Let us ask ourselves this question, Is X r the weighted balance separator? Well, what you know that if you delete X r, what is the property? That the property is that vertices appearing here.

So, there will be a connected component like the connected component of my graphs are either contained inside this vertices here or this vertices here but they will not intersect all their intersection vertices are contained inside X r. So, now I ask myself, look at this connected components. Is this property does it property hold? I ask for every connected component, Is the weight of C at most weight of vertex set of G divided by 2? My answer is no.

What is the meaning that my answer is no? Because it means there is a connected component here, which is either contained inside this green blood, or it is contained inside this purple blood. Such that the weight of this component is more than the weight of this. So, what will I do? I said fine, now in that case. So, if this was a weighted balance separated we have found this. Else, you move your token to suppose, this is heavy and heavy belongs to this side.

Then what will I do is that, I will focus my I will move my I will move, I will ask the part of the rooted graph the part of the comp like part of the, so if this is that component is inside this, then now I move my,this token here that piece and ask myself. Well, I will ask the same set of question to suppose this was like some other t.

**(Refer Slide Time: 15:00)**




Then now what will I ask second, Is X t, the desired weighted balance separator? So, now let us so now, what happens is the following, now you delete X t, the moment you delete X t what happens? So, now I delete X t and then all the components of this pieces are going to be here. And including something that comes from the because of that red vertex something. And all the components below X t from let us say here will be here into several components.

Now what could happen? If, X t does the job then we are happy, but notice what is happening at any point of time. At any point of time the way of reduction is that why did we move towards X t? Because, it had a connected component whose weight was strictly larger than the weight V of t. It means if I look at this even some of the all the connected components here, what is the property? Because this had a connected component whose weight was larger any connected component.

That we are going to get here, has a property that it is, weight is at most divided by 2. So, this is great. Now, what is what like so, either this X t does our job or look at one of its children. Like one of its children has a connected component and that connected component is like, is contained inside the subtree rooted like the, like if you look graph induced on the vertices in one of the subtree rooted at t that connected component is heavy.

So, then what will you do if suppose either this X t will do the job or maybe suppose we had a different then you will move the say t prime say is this X t prime does my job or not. So, notice that what is an invariant which are maintaining is that the heavy component is in the subtree rooted at the my current potential weighted vertex separator either that vertex does the job or I move below.

**(Refer Slide Time: 18:26)**

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

— We are moving away from the root r.

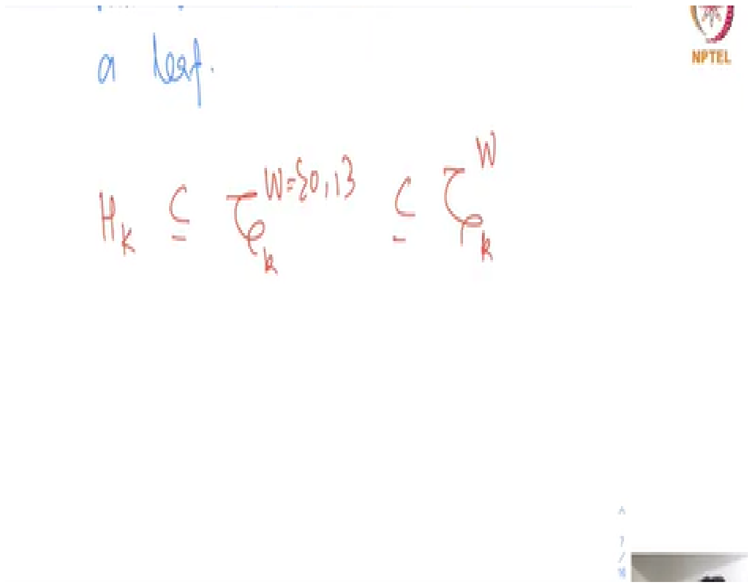
— Since my graph is finite either we will terminate or we will reach a leaf.

But at each time what is happening? At each time what is happening, that we are moving away from the root r. So, at since my graph is finite, either we will terminate or we will reach a leaf. Then, what is the property of that leaf? If you notice, if I reach some leaf here then we know that if I delete this, suppose this x l, what is the property? That the weight of this excel is very heavy, is like it is more than half of the whole vertex, here.

So, the moment if I delete it, I know that every connected component has weight at most the weight of the original graph divided by 2 because X l itself has weight larger and you are done. So, this is very identical to the proof of balance separator that we did for the for trees but now we are doing the weight function and now rather than one vertex being the separator it is a bag which is associated to the node of my rooted tree that acts as a balance separator.

So, now before we had 1 vertex because we were talking about tree, but now we are talking about graphs of bounded tribute or tree com tree with k. So, now the this back side X t is the size of the separator that we are talking about.

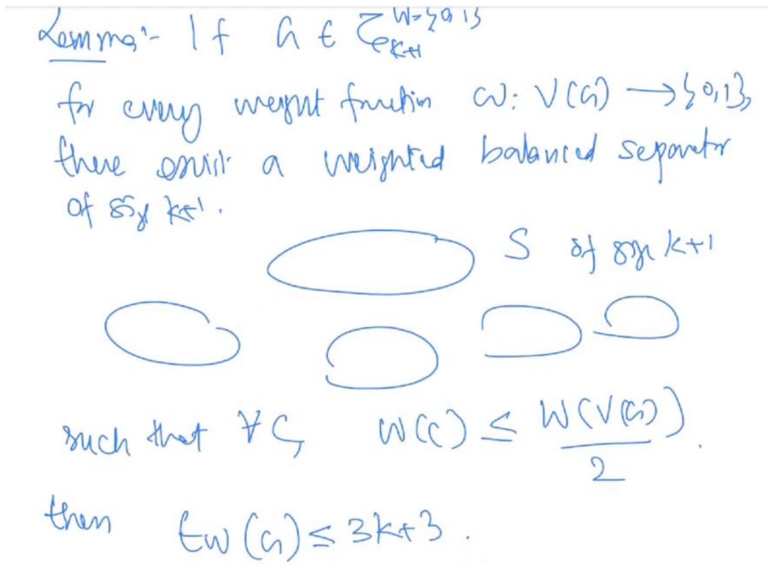
**(Refer Slide Time: 20:06)**




So, we have shown this, what is it implies? It implies, that we have proved, in some sense we have proved 1 that actually we have proved that H k if a graph has, so now, here is my H k and suppose this is like weight function but only 0, 1 and let us define arbitrary weight function. Arbitrary weight function, like what I mean by arbitrary weight function? Means like given a graph G and any weight function W you have a weighted balance separator with respect to that.

And with 0, 1 it is only those weight function, so I am taking graph G, but I restricted myself to only 0, 1 weight function. So, What we have shown now that H k actually be like, if you have a graph of tree width at most k, then actually it has a weighted balance separator with respect to zeta z. So, this is what we have shown that H k belongs to this family of graph. But now, what we are going to show to you so partial converse and that is what will constitute our algorithm.

**(Refer Slide Time: 21:23)**

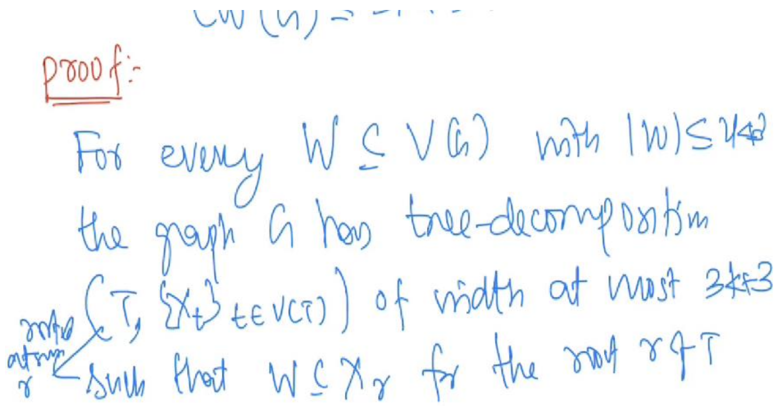



So, my lemma is if G belongs to, say I will write it explicitly, what is the meaning of this? Meaning, for every weight function W from V 0 to sum 0, 1 there exist a weighted balanced separator, there gives a weighted balance separator of size k + 1. Meaning, for every weight function W from V G to 0, 1. What is the meaning of this? So, I will be able to find some set S of size k + 1 rather let us say size k + 1.

Such that for all component C weight of C is at most weight of vertex set of G divided by 2, that is it. If G has this, then I am going to show to you first that the tree width of G is at most 3k + 3. So, let us try to understand, what does it say? It says that look I have a graph G such that what is the property of this graph that you should give me any 0,1 weight function, I will be able to find a balance separator of size k + 1 like weighted valence separator of size k + 1.

Then what can you say, then I say if that is the case then I will be able to show to you that the tree width of this graph is at most 3k + 3. So, what will be prove actually to you.

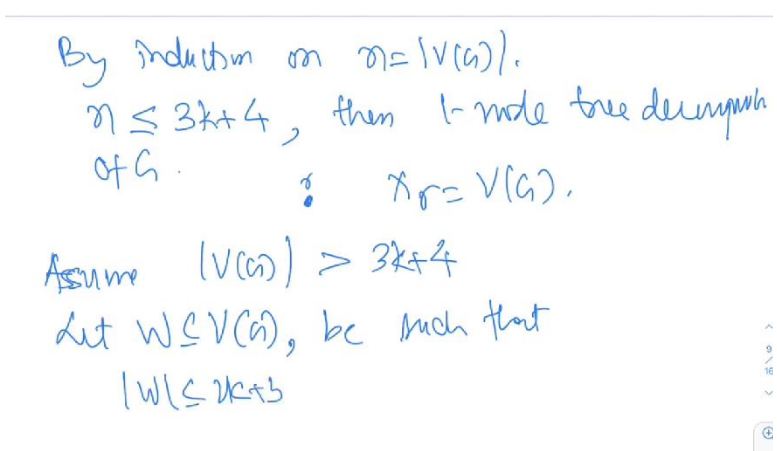
**(Refer Slide Time: 24:07)**




So, we I am going to prove this to you and in fact, I am going to prove slightly more general theorem, in fact what I am going to prove key for every W subset of V G with is at most 3 the graph G has tree decomposition the graph G has a tree decomposition T, X t of width at most 3k + 3 such that W is subset of X r, W is subset of X r for the root r of T. So, we will always assume that t is rooted at some r. So, what this tells us that?

Look not only this you give me any set W of size at most 2k + 3 in fact I can come up with a tree decomposition where the root bag will have the property that W is contained inside that root. So, I mean so this is and you will see that,why we will need this? Because we are going to prove our statement using induction. We are going to prove our statement using induction.

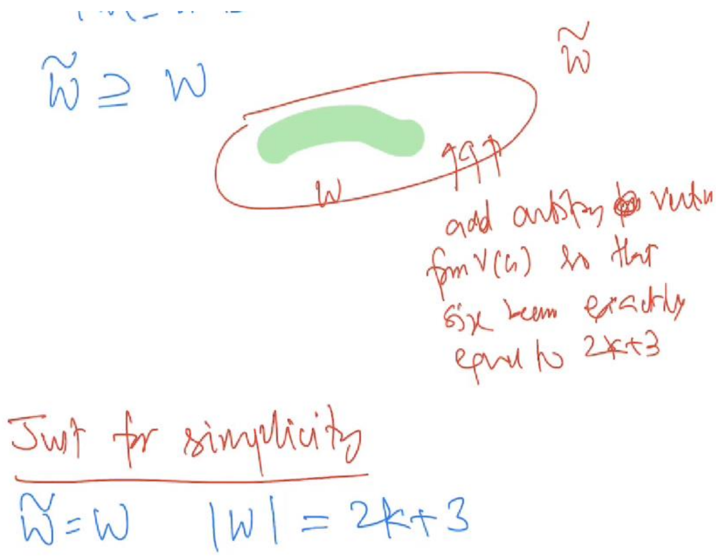
**(Refer Slide Time: 26:22)**



So, proof is by induction on number of vertices of my graph. So, what I have, what is my base case, I am going to ask if number of vertices is less than 3k + 4. Then if n is at most 3k + then we will return 1 node tree decomposition of G and right that is it, one node tree decomposition. So, basically, what will you do, you just assign a node it is a root everything? And, what is X r? X r is basically V of G, that is it. And you notice that, it satisfies all the property.

So, from now onwards we are going to assume, that the vertex set of this graph is strictly greater than 3k + 4. Now, let W subset of V G be such that you gave us. And what is the size of W, is 2k + 3.

**(Refer Slide Time: 27:47)**




So, what I am going to do that given this W I am going to construct W construct W tilde, I am going to construct W tilde, which will contain W. By what so suppose? This is my W. But the size of W is strictly less. So, what will I do to make W tilde I will add arbitrary vertices in the arbitrary vertices from V G. So, that size becomes exactly equal to 2k + 3. And, notice because we have that many vertices.

So, we can always if you gave me W whose size is much smaller than or strictly smaller than 2k + 3 then I will arbitrary add subset of vertex, some set of vertices from my graph and make a set W tilde and notice that if I can come up with a tree decomposition with W tilde being the root then definitely that other property that even W is contained inside this because W tilde contains

W. So, just to, just for simplicity. We will rename W, W tilde is nothing but W and W tilde has size exact W. And we just renamed it. Otherwise it has size exactly equal to 2k + 3.

**(Refer Slide Time: 29:49)**

$\tilde{W} = W \qquad |W| = 2k+3$

– G, has weighted balanced separator for any $\{0,1\}$ weight function.

$f : V(G) \longrightarrow \{0,1\}$

$f(v) = 0 \quad \text{if } v \notin W$

$f(v) = 1 \quad \text{if } v \in W$

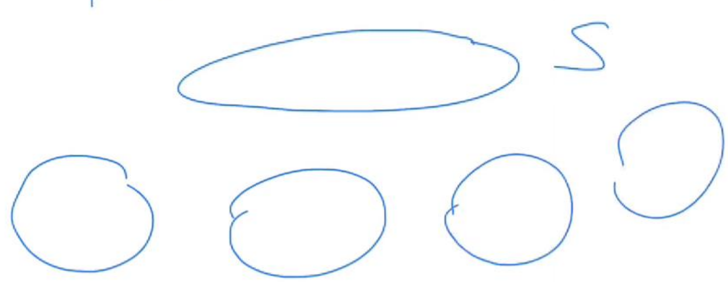
Now, what will happen? So, now what I am going to do? I am going to, I know that my graph G has weighted balance separator for every 0, 1 weight function. So, I fix a weight function. What is my weight function? So, my f of v G to 0, 1 is so, basically what I would like to do is that, I would like W to be separated very nicely. So, then what I am going to do is that I am going to assign f of V is 0 if V does not belong to W and f of v is 1 if v belongs to W.

**(Refer Slide Time: 30:44)**

$f(v) = 1 \quad \text{if } v \in W$

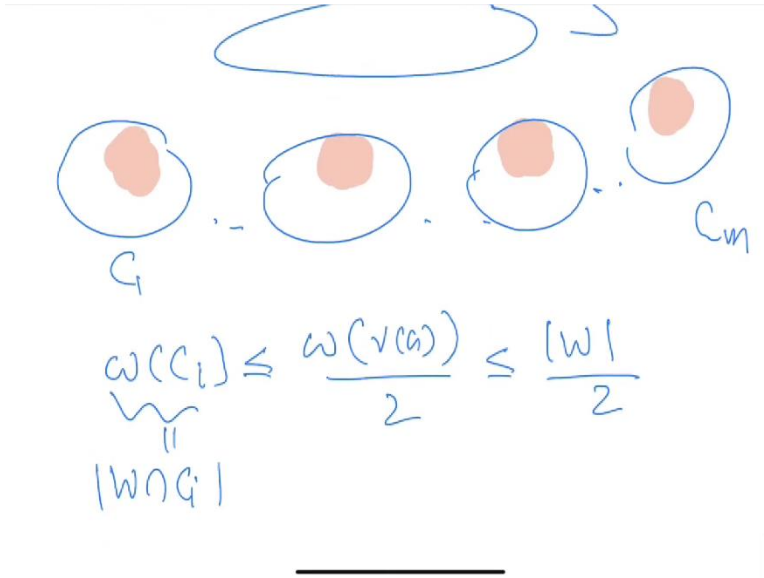
$w(V(G)) = |W|$

a separator $S$



So, notice what is the weight of vertex set of G is nothing, but equal to W. So, it means there exists a separator. There exists a separator S whatever property of the set S? Now, look at this property of set S, I look at every component. So, what is my component? Suppose we have component C 1 to C m M components.

**(Refer Slide Time: 31:25)**




The property is that if I look at weight of C i, it is at most weight of vertex set of G divided by 2 which is nothing. But, at most weight of, so we it is nothing but cardinality of W divided by 2. And what is weight of C I is, basically this is nothing but W intersection C i. So, this is what it is. Now to apply induction what we like to do is that to apply induction I would like to say, I will try to see that.

**(Refer Slide Time: 32:31)**

$G_i := G[C_i \cup S]$

• to apply induction

$\rightarrow |C_i \cup S| < |V(G)|$

$W_i := (W \cap C_i) \cup S$

$|W_i| \le 2k+3$

Call: $V_i := C_i \cup S$

$$|V_i \cap W| \le |W_i| \le |W \cap C_i| + |S|$$
$$\le \frac{|W|}{2} + |S|$$

Find I will make a graph G i, what is a graph G i? It is graph G intersect is C i union S. So, basically so what we have done here? Now is that we are taking this as one piece of the graph. So, I am going to decompose this, then I am going to decompose this in as a one piece and so on and so forth. So, this is what G i means. But now I would like to apply induction I will like to apply induction.

So, to apply induction first of all I need to show that C i union S is strictly less than V G. That is one thing and I have to define what my W i is, so, the tuple on which I am going to apply induction. Because, what I am proving inductively, inductively I am trying to prove that given a graph G and I said W of particular size, I can come up with a tree decomposition such that the root back contains the W i specified.

So, now I am going to call induction on this particular graph, so I need to show that this is this, and I am going to set W i as what I am going to say W i is nothing. But, W intersection C i union S this is what I am going to set. So, the intersection of W which I started with this blood and set S. So, I also need to show to apply induction that the W i is at most 2k + 3 to apply induction I need to show what is, what was that I started with, so we started with 2k + 3.

So, I need to show that this is at most 2k + 3. So, let us try to do each of these things. Now, let us try to show bound let us see, what is V i? Or rather, let us, I would like to upper bound W i intersection C i. So, what is this? W i intersection C i is, let us call V i C i union S. And now what I am interested in is that what is V i intersection? V i intersection, capital W. So, what is the V i intersection capital W?

So, if you notice what is V i intersection capital W is upper bounded by what like, where could W i,W is going to be either here or it is going to be contained inside S in this portion of the graph. So, which implies that, V i intersection W is nothing but, upper bounded by W i which is upper bounded by W intersection C i + cardinality of S. But, what is the W intersection C i? W intersection C i is upper bounded by cardinality of W by 2 + cardinality of S.

What was cardinality of W? cardinality of W was exactly equal to, if I recall correctly we made it exactly equal to 2k + 3.

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

$$= \frac{2k+3}{2} + k+1$$
$$= k + \frac{3}{2} + k+1$$
$$= 2k + 2.5$$
$$< 2k+3 = |W|$$

- there exist a vertex of original W that donot belong to $V_i$

$$|V_i| < |V(G)|$$

So, 2k + 3 divided by 2 and what is cardinality of S was exactly equal to 1. So, this is equal to k + 3 by 2 + k + 1 which is 2k + 3 by 2 + 1 is 2.5. So, what do this implies? This is exactly equal to strictly less than equal to 2k + 3 which was equal to W. So, what does this imply? It implies that there exists a vertex of W, what does this statement implies? It implies that there exists a vertex of original W that do not belong to the existing vertex out that do not belong to V i.

What does that imply? It implies that cardinality of V i is strictly less than cardinality of original graph. So, it is good for applying induction. So, we have shown that this is strictly.