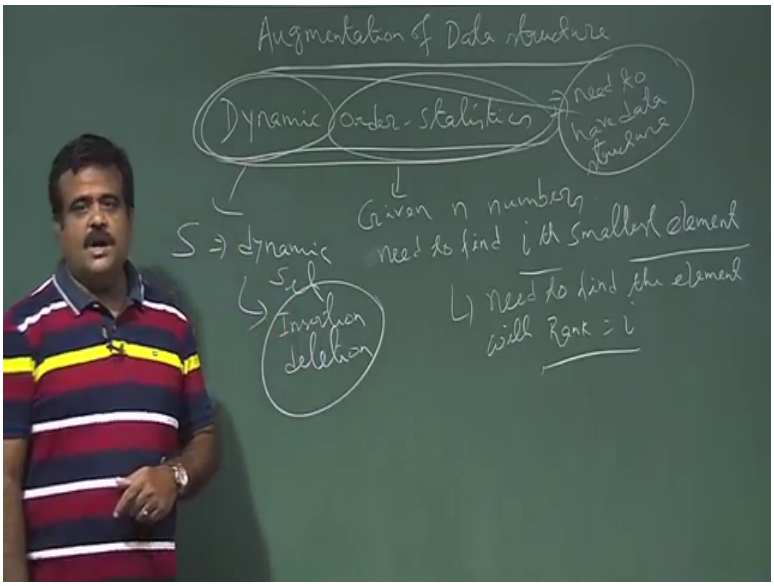
An Introduction to Algorithms
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture - 29
Augmentation of Data Structure

So we talk about augmentation of the data structure. So, the problem is suppose we have a problem, now we need a data structure for that problem. So, the augmentation augmenting means.

So, instead of being a brand completely new data structure from the sketch, what we can do, we can make use of the old data structure which you know and we can do some augmentation there. We can do some we can have some extra field of information for our problem. And then which will help us to solve our problem so that is called augmentation augmenting of data structure. So, we will we will we will take 2 example for this 2 problem. First one is dynamic order statistics ok.

(Refer Slide Time: 01:08)



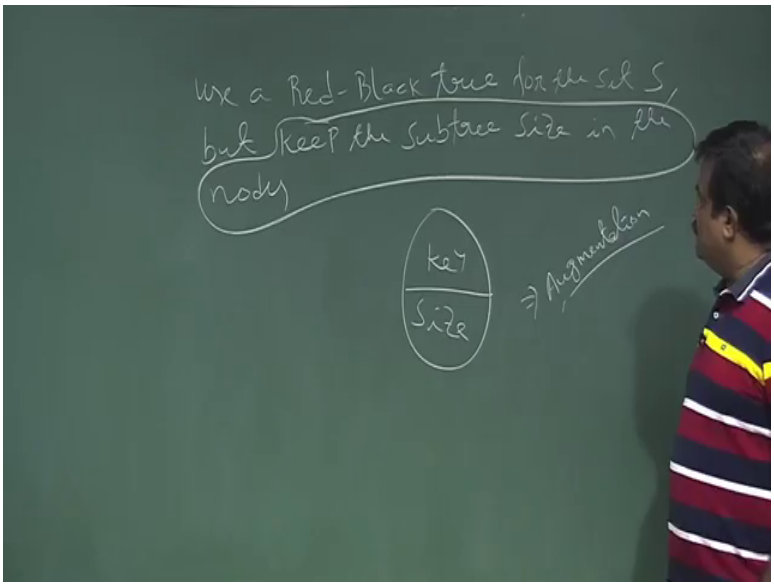
So, we have seen the order statistics problem, order statistics problem means, we have a we have given a we have given N numbers a array of N numbers, and we need to find out so, need to find out i-th smallest element. So, this is the problem of order statistic so that means, we need to find out the element we need to find out the numbers element whose rank is i.

So, need to find the element with rank is equal to i. So, rank of an element means we have given the array or given N element if we so, rank is the position of that element in a sorted one sorted array. So, if you sort it the position of this element. So, this is the problem of order statistics. And we have seen some algorithm to solve this problem like select then the randomized version of the select. And then we look at the good finding the good that randomized version will take the linear time now, then we look at the look for a good pivot so that it will worst case it will be guaranteed linear time.

So, that is that we have seen now. So now, if this set is dynamic set so that is called dynamic order statistics. Suppose we have a set s, which is dynamic set dynamic. Set means so; it is changing any time the anybody can join. So, dynamic set means insertion and the deletion is allowed. So, these 2 operation make a set is dynamic. So, we are allowing anybody to leave from this set at any point of time that is deletion. And we are allowing the join a num join a element in this set. So, this set is dynamic. Now suppose we have a dynamic set and the problem is to find the i-th smallest element in that set. But now the set is dynamic.

So, for this problem we need to have a data structure. So, we need to have data structure for this dynamic order statistics, need to have data structure data structure to solve these problems so that we can do this problem in a faster way. So, for this we will just do the augmentation of the data structure, like we know we have a good data structure like red black tree which is balance. So, we will use the underline data structure as red back tree, and then we will do some augmentation there like we will we will keep some more information in the data structure. And that will solve our; that may help our problem. So that is called data structure augmentation.

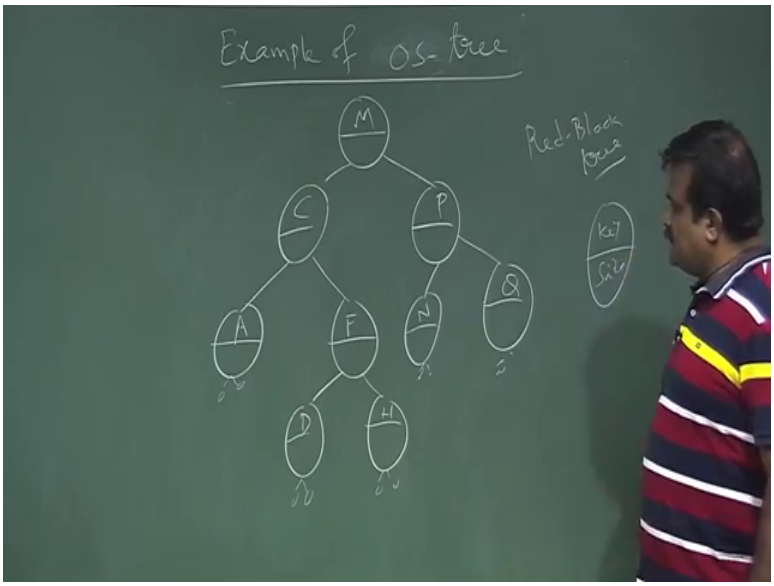
(Refer Slide Time: 05:08)



So, basically the idea is to so basically idea is to use the red black tree because it is a balance data structure, balance tree red black tree for a for the set s, because s set is dynamic. So, in the red black tree we can also perform the dynamic operation modifying operation like insertion deletion. And but we have to do some augmentation, but keep the subset size, keep the sub tree size in the node in the nodes. So, this is called this is the augmentation of the data structure.

So, we will just use the our red black tree data structure. So, we will we will make the red black tree. But we will we will keep this extra bit of information, extra information so that will help us for our solving our problem. So that means, each node is having this field this is the key and this is the size of the tree, size of the sub tree rooted at that node. And this is called augmentation this is called augmentation. So, so in red black tree we have key based on this key we perform this and with the key each node we are having this we are keeping this information also, for this will help us for our problem our red black our dynamic order statistics problem ok.

(Refer Slide Time: 07:19)

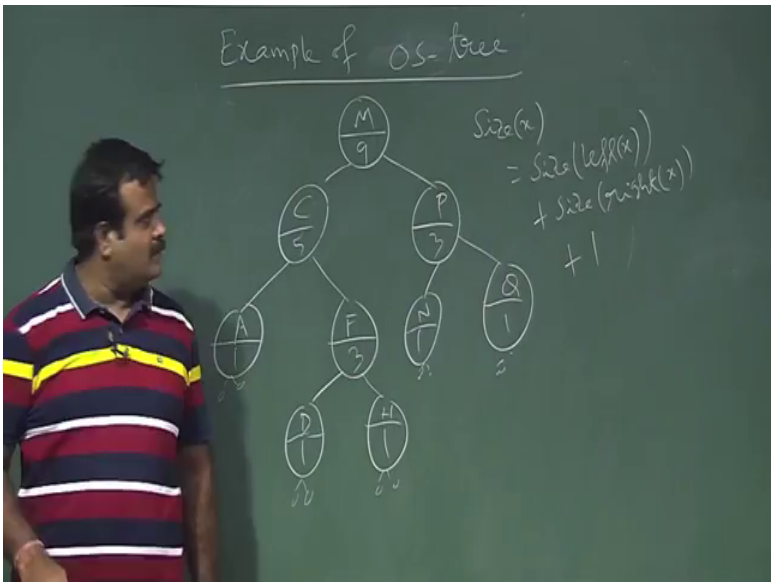


So, let us take an example. So, this is called order statistics tree we can say. So, this is the example of OS tree. This is basically red black tree, suppose these are the nodes. So, C P A F N Q D H.

So, then we have the leaf nodes to make this So, this is the this is our red black tree. And these are the notes these are the key. So, C is less than M. So, this M N, ABCD are in alphabetical order. So, the order they are coming in the alphabet here there are I mean, A is less than C in the alphabetical order. So that is why A is in the left side of the C. So, this is a, this is a red black tree we can make the color of it also. So now, we need to put this extra bit of information that is the So, this is the key value and this is the size. So, size of the tree rooted at this.

So now, So size is 1 for this node size of this nils are basically 0. And size of So, this is 1 and this is also this is the this is basically size of the tree rooted at this. So, size is basically 3. Size is 1, size is 1, 1 3 like this. Now what is the size of this size of this is basically size of the number of nodes rotated at this; so 3 plus 1 4 and then 5. So, this is now this is also this plus this 8 and plus 1 9. So, what is the formula we are using for size?

(Refer Slide Time: 09:53)

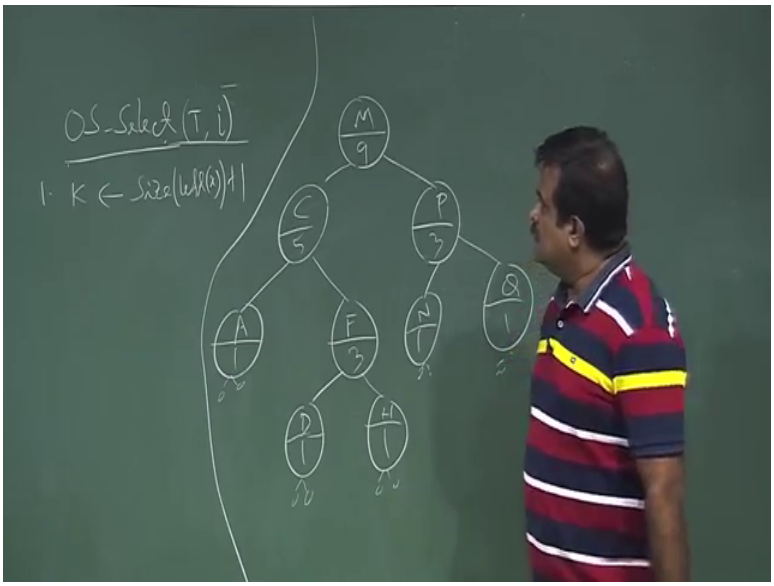


So, size of a node is basically size of the left plus size of the right plus 1. So, size of x is basically size of left of x plus size of right of x plus 1.

So, this is the So, size of left of x is 5; that means, the left sub tree size of the left sub tree is 5 for this node size of the right sub tree is 3 and the node itself. So, this is basically 5 plus 3 8 and plus 1. So, this is the formula for making the size. So, we can easily get the size; now suppose we maintain this size, now how it will help us to find the i-th smallest element? So that is called so, we want to perform OS select.

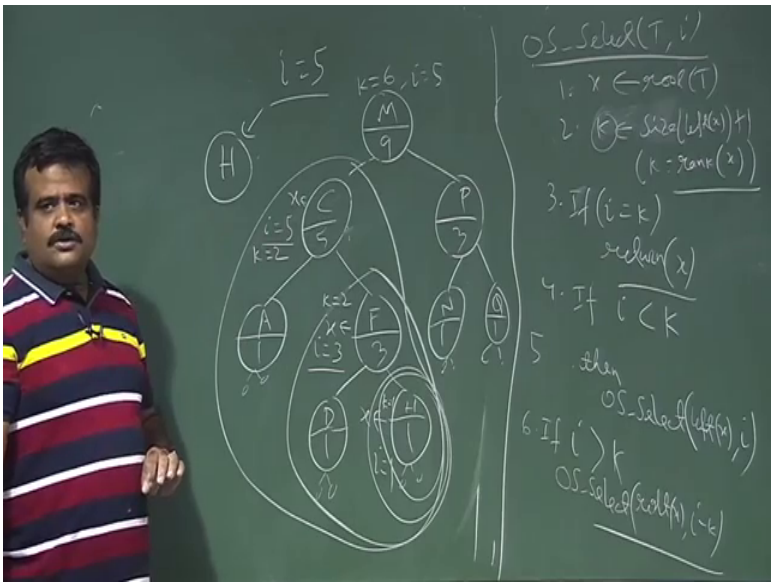
OS select, so we want to get the i-th smallest element in this. So now So, for that we just So, suppose this is the red black tree and this is the x. So now, we want to find the i-th smallest element that is called OS select i So, order statistics, but here the dynamic on the dynamic set; so i-th smallest element.

(Refer Slide Time: 11:42)



So, what we do So, basically we start with this if you write k is equal to size of left of x plus 1, just if we do k is equal to size of left of x plus 1, can you tell me what is the meaning of this k? So, basically before that let us take the root as a.

(Refer Slide Time: 12:16)



So, x is the oh maybe we can write there we have more space. So, OS select t comma i. So, we just put x is equal to root of t and then we take k is equal to size of left of x left of x plus 1, size of left of x plus 1. So, what is this k means? So, k is basically denoting the rank of x ok.

Why k is rank of x? Because see I mean this is the red black tree, now red black tree is a binary search tree. So, if it is a binary search tree then the in order traversal will give us the sorted array I mean the sort that is the BST sort. If you recall the BST sort in the BST sort what we are doing we are forming the tree, we are inserting the we are forming the tree and in the BST what we are doing we are just performing the in order tree walk. So, if we perform the in order tree walk that will give us a sorted array, I mean in order tree walk basically it is the left subtree then root then the right subtree. So that means, So that means, the left size of the left. So, after the left subtree the root will be printed. So that means the rank of that node.

So, will come after the printing all the all left sub tree node. So that is the size of the left subtree. So that means, position of ax is basically in the sorted array is basically size of all nodes I mean size of the left subtree plus 1. So, this is this means k is the k is basically rank of x. Basically, this is coming from in order tree walk. Now we got k, now if So, suppose we are looking for suppose, we are looking for I mean k-th smallest element. So, if i is equal to k then we are happy we return x, otherwise if you are not So happy I mean if not So, lucky I mean if we are just get the k which you are looking for k-th smallest element, that is the root itself. Otherwise what we do? If i is less than k else if, i is less than k,

So that means, we know this root is the say k-th smallest element and we have. So, this is the 7 th smallest element say this is 5. So, this is the 6 smallest element, suppose we are looking for third smallest element, then i is less than k. So, what we have to do? We have to look at the left part of the tree. So, then if i is less than k then we just look at, then we call OS select with left of x comma i, otherwise if i is greater than k, if i is greater than k then we have to.

Look at the right part of the tree then we have to call OS select again right of x comma, but we have already seen the k-th smallest; so i minus k-th smallest in the right part of the tree. So that is the OS select now let us take an example, suppose we have for this case, suppose we all looking for say fifth smallest element in this example. Suppose we are looking for fifth smallest element we have a Q over here, we are having Q over here. So, this is the complete tree I mean, now suppose this is our set this is our set s and we have this structure we have the this data structure which is basically a red black tree, but we did some augmentation there by adding this extra field that is the size we are keeping

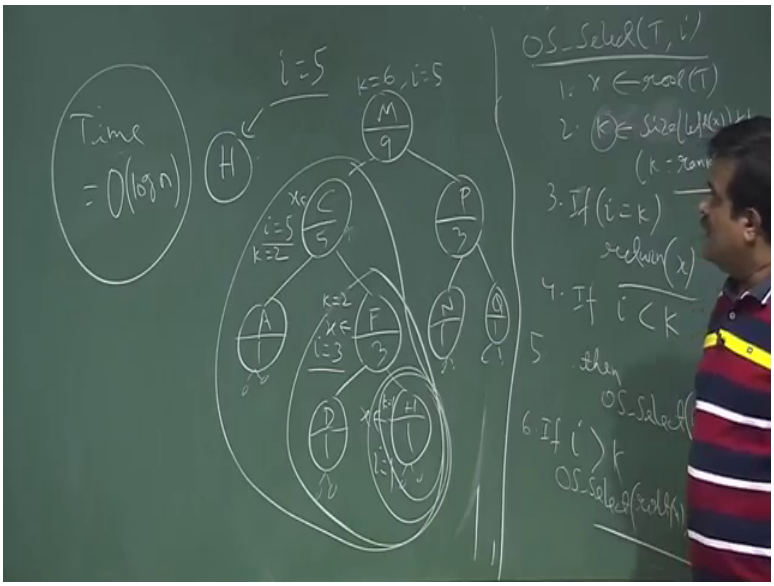
the size of the tree. Now suppose we are looking for fifth smallest element. So, we just execute this code OS select.

So, we come here this is x. So, what is the k value k is basically six, but we are looking for fifth smallest element. So k which is not the case over here So that means, we have to go further. Now i is less than six. So, we have to go for the left part of the tree, and this is our x now. And here also you are looking for fifth smallest element with this, now what is the rank of this rank of this is basically k is basically left subtree is 1 plus 1 2, rank of this is 2.

So now, i is greater than this So, you have to go for the right part of the tree and this is our x. But now we have already seen the second smallest. So, we are looking of fifth smallest now our i is basically this i minus k. So, third smallest element in this subtree we are looking for. Now what is the k over here; what is the rank of this node? Rank of this node is again 2. So now, i is greater than so, we have to go for right part of the tree with x is this, but with i values is basically 1. So, this is our So, what is the k? K is basically 1. So, we are getting. So, H is basically fifth smallest element is basically H. So, H we return. So, this code will return basically H.

So, this is the OS select. So, so this set is dynamic; that means, any we can insert a node over here, suppose we can in insert a node over here we can delete a node over here. And So, what is the running time of this code? Running time is basically order of log N because, we are just going to the height of this tree, when you are searching we are just going to the height of this tree.

(Refer Slide Time: 19:35)



So, that height is this is the balance tree, so that is why time for OS select is log N. So that is the advantage of having the balance data structure balance tree. So, this is the balance tree and So, the height is H and we are going we are executing this we are going at most the height comparison. So that is why it is order of n. So now, the question is how we can perform this insertion and deletion operation, and now the.

So, how to perform insert and delete operation here? So, this is the dynamic order of statistics. So, so this is Q. So, suppose we want to insert a node say which is basically k, suppose we want to insert a node k we want to insert k.

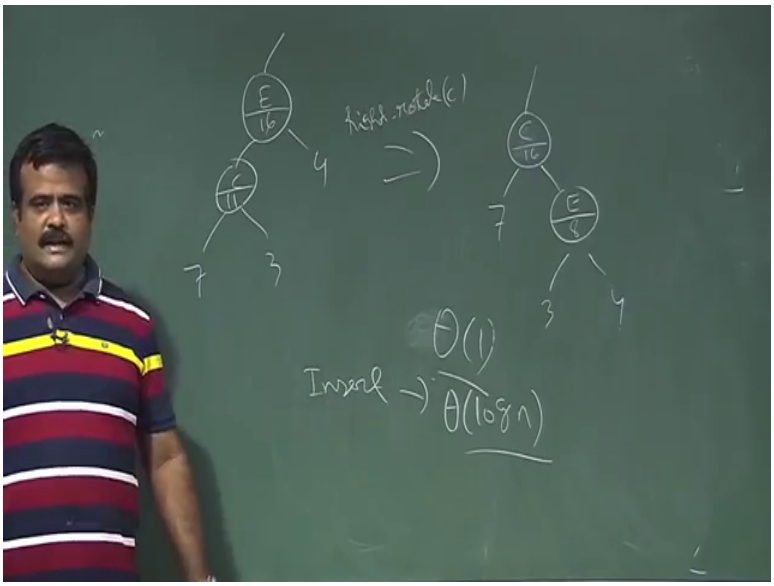
(Refer Slide Time: 20:45)

So, what we do? We just k is alphabetically less than M we will go to the this side k is greater than this we will go to the this side k is greater than this will come here k is greater than this will insert k over here.

Now once you insert k we need to change this field like. So, this is now became 1, and this will become 2 now, and this will become now 4, and this will become 6, and this will become 10. Now all the path it will be changing. So, this operation, and not only that then again we have to make this tree as a red black tree. So, for that we need to do the red black tree insert operation. So, in that case we need to maybe perform the rotation operation. So, we have to see in the rotation operation how we can handle the rotation operation, whether it is easy to handle basically we need to handle this extra bit extra information that augmenting things. So, if we can handle this extra information in a in a constant time or in the same time with the red black tree rotation, then we are in then there is no issue ok.

So now, we will talk about how to handle this rotation operation for a this for this extra information.

(Refer Slide Time: 22:19)

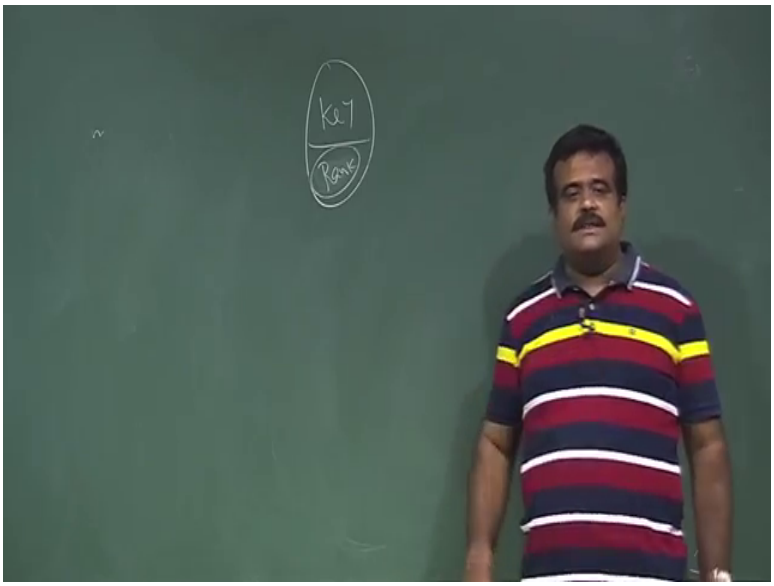


So, suppose we have, we have a tree like, this E then we have C, and we have a tree hanging over here with 4, 4 7 3. Now we want to C up E down. So, this is basically right rotate, right rotate on C. So, C will be up and E will be down sorry, C will be up E will be down and the left part. So, this 7 will be hanging here.

And 3 will be hanging here, 4 will be hanging here. Now how to make these changes? Now this is basically this plus this 11, and this is basically 15 and this 16. Now here also we can just change this. So, this is 3 plus. So, this is 8 and this is also 16. So, this is also similarly I mean 2 to maintain this extra bit of information, we are not really using we are just using constant time. Because just we are change because these are not changing these are alpha beta gamma, alpha beta gamma. So, these are not changing we are just we are just changing these bits. So, these bits is or can be maintained in constant time. So, if this is maintain in constant time then the insertion, so insertion or deletion can be done in logarithm time.

So, this is most important. So, this dynamic set we can preserve this set again this red black tree by logarithm time that is important. So now, the question is instead of storing the subset size. So, our augmentation is we have the key value; here we are storing the subset size.

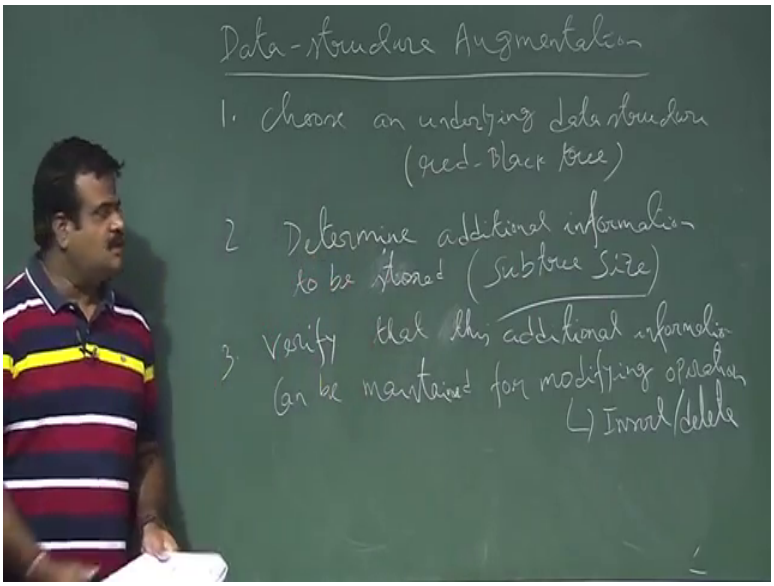
(Refer Slide Time: 24:32)



Now, if we just store the rank itself, if we do this type of augmentation, then what is the issue? If we can store the rank then we can just look at the this looks like easy to search the i-th smallest element basically, we need to search the element whose rank is i, but problem with showing the rank is, now suppose we insert a node which is a minimum then the all other node which (Refer Time: 25:06) have then we need to change all the elements, the rank will change because we have inserted minimum or maximum element, then the rank of each node will change sorry,

So in that case it will take it will not take height time it will take the order of N times.

(Refer Slide Time: 25:38)



So, the keeping rank is not a good idea, sorry. So, let us just write the methodology for this data structure augmentation. In the data structure augmentation basically we have seen the we have to choose a order statistics which is I sorry, we have to choose a data structure. And in that data structure we have to do some augmentation. So, basically we have to choose a underline data structure, data structure. So, here we have taking the red black tree in our example.

And then we have to decide based on our problem we have to decide the augmentation things; that mean, we have to determine the additional information, determine the additional information to be stored, additional information to be stored, for our case for our OS select we stored the subset size, sub tree size. We did not store the rank of a note, because it will not help us. Then we have to verify the modifying operation, verify how we can maintain this additional information, verify that this additional information.

This additional information can be maintained for modifying operation, maintain in the same timing. Modifying operation means insert delete insert a node delete a node. So, this is basically the methodology behind the our data structure augmentation, basically we take a underline data structure. Then we put some extra bit of information based on our problems. So, in the next class we will do another problem where we have to use the data structure augmentation, which is called interval trees. So, we will discuss in the next class.

Thank you.