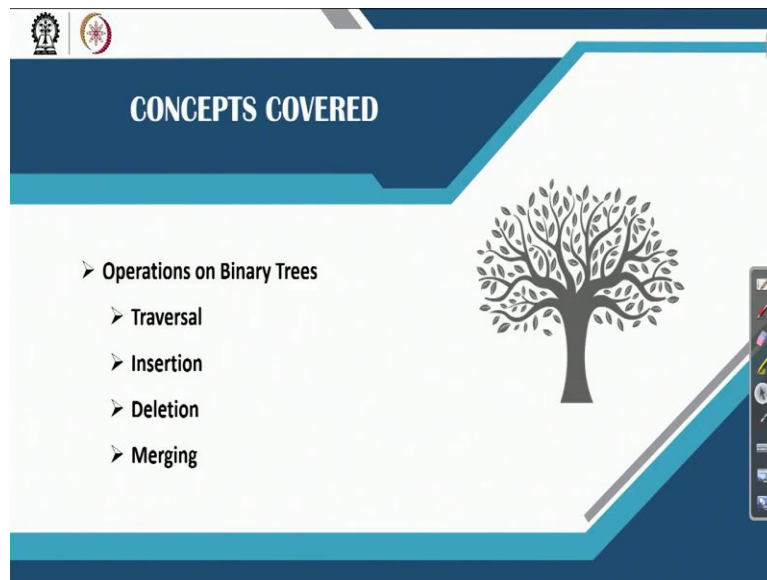


Data Structures and Algorithms Using Java
Professor Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 28

Operations on Binary Tree Data Structures

We have learned about binary tree data structure, which is a special of its kinds and distinguishable compared to other data structures, like array stack queue link list. This is because it gives a non-linearity so far the information maintenance is concerned. Now, so information how it is stored in a binary tree data structures that we have learned in the last video lectures. So, in this video lectures, we will try to learn about different operations on this data structure.

(Refer Slide Time: 01:08)



So, the most frequently used operations are listed here like traversal, insertion, deletion and merging. So, let us see how all those operations can be defined first. Then we will go for implementation of all these operations in Java programming language, which will be discussed in the next video lecture.

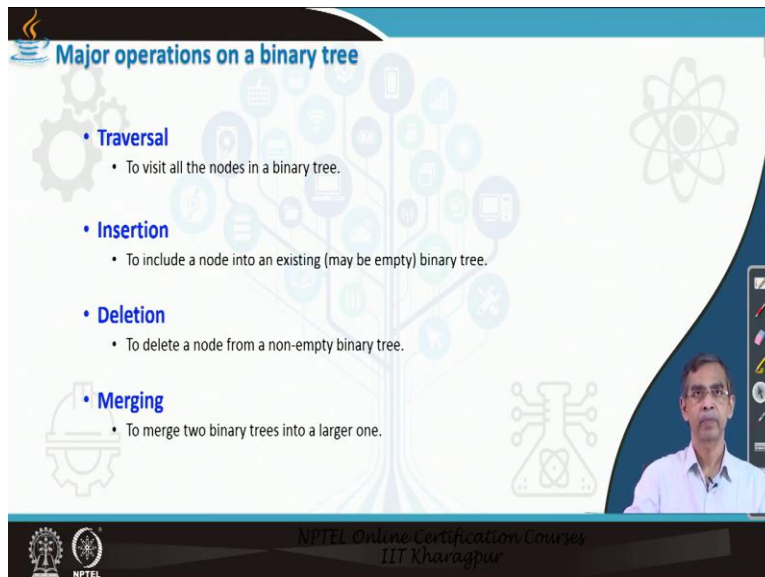
(Refer Slide Time: 01:29)



The slide features a central graphic of a tree where the branches are represented by various icons such as a gear, a lightbulb, a document, and a network symbol. To the left of the tree is an icon of a coffee cup with steam rising from it. The title "Operations on Trees" is written in a bold, blue font. The background is white with a blue border at the top and bottom. A small inset video of a man in a white shirt is visible in the bottom right corner. The footer contains the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

Now, so operations on trees means how we can manipulate these data structures, manipulation in terms of printing all the elements those are there in the structure or insert a new element into it deleting an element from it or merging two structure into one structure.

(Refer Slide Time: 01:51)



The slide lists four major operations on a binary tree, each with a brief description. The title "Major operations on a binary tree" is at the top left. The operations are: Traversal (To visit all the nodes in a binary tree.), Insertion (To include a node into an existing (may be empty) binary tree.), Deletion (To delete a node from a non-empty binary tree.), and Merging (To merge two binary trees into a larger one.). The background is white with a blue border at the top and bottom. A small inset video of a man in a white shirt is visible in the bottom right corner. The footer contains the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

- **Traversal**
 - To visit all the nodes in a binary tree.
- **Insertion**
 - To include a node into an existing (may be empty) binary tree.
- **Deletion**
 - To delete a node from a non-empty binary tree.
- **Merging**
 - To merge two binary trees into a larger one.

So, these are the basically the major operations those are required in order to use it in any application.

(Refer Slide Time: 01:58)

The slide features a central graphic of a tree where the branches are represented by various technology-related icons like a smartphone, a laptop, and a Wi-Fi symbol. To the left of the tree is an icon of a steaming coffee cup. The title 'Tree Traversals' is written in blue text. A small video inset in the bottom right shows a man speaking. The footer contains the NPTEL logo and the text 'NPTEL Online Certification Courses IIT Kharagpur'.

Now, let us see first traversals operation. So, traversal operation is the most frequently used operation we can say, this basically as you know traversal means it is basically visiting all elements in the structure. So, all elements means here actually all elements are stored in the form of a node. So, basically we have to visit all the nodes of a binary tree.

(Refer Slide Time: 02:22)

The slide is titled 'Traversal operations' and contains a list of four bullet points. To the right of the text is a binary tree diagram with a root node '+', a left child '-', and a right child '-'. The left child '-' has children 'A' and 'B'. The right child '-' has children 'C' and 'J'. Node 'J' has children 'E' and 'F'. A red hand-drawn box highlights the entire tree structure. A small video inset in the bottom right shows a man speaking. The footer contains the NPTEL logo and the text 'NPTEL Online Certification Courses IIT Kharagpur'.

- Traversal operation is a frequently used operation on a binary tree.
- This operation is used to visit each node in the tree exactly once.
- A full traversal on a binary tree gives a linear ordering of the data in the tree.
 - For an example, if the binary tree contains an arithmetic expression, then its traversal may give us the expression in infix notation, postfix notation or prefix notation.

Now, so traversal operation actually can be done in many ways and as you know a tree structure can be access from specific node that is a specially designated known it is called the root node. So, if we can start from the root node, then we have to traverse by following a systematic

method, if you do not follow systematic method actually, we will not we may miss some nodes unvisited like, I am not visited. So, we have to do a systematic method to visit it.

Now, there are many ways this systematic city can be realized, but out of these the three major mechanisms are there major techniques rather are there they are called infix traversals and prefixed traversal and postfix traversals. Now, so for example in this binary tree if we follow a particular order because this binary tree as you see it actually represents and arithmetic expression, so as I said that infix traversal is a very one kind of traversal actually, if we follow I mean this infix traversals on these binary tree, then we will be able to see that this will give an arithmetic expression in infix notation.

We have also learned while we are discussing about stacks structure that postfix notation. So, postfix notation corresponding to another traversal namely called postfix traversal like. So, actually it is called the in-order, pre-order or post-order, in-order corresponding to increase post-order corresponding to postfix and pre-order corresponding to prefix, prefix is just opposition to the postfix notation. So, anyway so there are many way the binary tree traversals can be binary tree can be traverse. Now, let us see how these techniques are what we have mentioned about three different techniques, how these three techniques are.

(Refer Slide Time: 04:36)

The slide is titled "Different ways of traversing" and features a binary tree diagram with a root node 'R' and two leaf nodes 'T'. To the right of the tree, a list of six traversal sequences is presented in a table:

1. <u>R</u> T T	4. T T R
2. T <u>R</u> T	5. T R T
3. T T <u>R</u>	6. R T T

The slide also includes a list of bullet points: "Now a tree can be traversed in various ways.", "For a systematic traversal, it is better to visit each node (starting from the root) and its sub trees in the same fashion.", and "There are six such possible ways:". The NPTEL logo and "NPTEL Online Certification Courses IIT Kharagpur" are visible at the bottom.

Now, so here I have given an idea about how different ways rather, how different ways that systematically a binary tree can be traversed, that mean all nodes in a binary tree can be visited.

So, basic idea is that anyway you have to start from the root node, so this root node, then they are two sub trees may be empty whatever it may be. So, this is called the left sub tree and this is called the right sub tree suppose, so there are two sub trees and the root is there.

Now, what we can do is I have mentioned is that in which order the things can be visited, I can visit first root node, then we can visit this left sub tree and we can visit right sub tree, now same mechanism we can apply to this left sub tree that mean visiting left sub tree means we have to visit the main node the root node of the sub tree and then again left of left sub tree then right of left sub tree and it will continue until we reach to the end node that is called the leaf node.

The same thing is applicable to this also. Now, this is why the first way that it can be traversed is first root, then left sub tree, then right sub tree, alternatively we can do first left sub tree, then visit right root node then right sub tree, another another approach may be first left sub tree, then right sub tree, then the root node. Likewise 4, 5, 6, the different order actually. So, basically what are the different order?

These are the three different structures can be visited. So, three different structures all taking all permutations $(())(06:29)$ three factorial that means six different ways that can be visited. Now, so these are they are therefore altogether six different ways, it is a systematically of course, the ways the binary a binary tree can be visited. Now, let us proceed further, each visit if we follow how it will get the output?

(Refer Slide Time: 06:52)

Example: Different ways of traversing

1. R T T	4. T T R
2. T R T	5. T R T
3. T T R	6. R T T

Visit 1:

- + T_l T_r
- + T_l T_r T_r
- + A T_l T_l T_r T_r
- + A B T_l T_l T_r T_r
- + A B * T_r T_r
- + A B * C T_l T_l T_r
- + A B * C / T_r T_r
- + A B * C / E T_l T_l T_r
- + A B * C / E F T_l T_r
- + A B * C / E F

NPTEL Online Certification Courses
IIT Kharagpur

(Refer Slide Time: 09:13)

Example: Different ways of traversing

1. R T T	4. T T R
2. T R T	5. T R T
3. T T R	6. R T T

Visit 1: $+ - A B * C / E F$

Visit 2: $A - B \mid C * E / F$

Visit 3: $AB \ C EF / **$

Visit 4: $FE / C * BA - \mid$

Visit 5: $F / E * C \mid B - A$

Visit 6: $** / FEC \ BA$

NPTEL Online Certification Courses
IIT Kharagpur

Example: Different ways of traversing

1. R T T	4. T T R
2. T R T	5. T R T
3. T T R	6. R T T

Visit 1: $+ - A B * C / E F$

Visit 2: $A - B \mid C * E / F$

Visit 3: $AB \ C EF / **$

Visit 4: $FE / C * BA - \mid$

Visit 5: $F / E * C \mid B - A$

Visit 6: $** / FEC \ BA$

NPTEL Online Certification Courses
IIT Kharagpur

Now, so I you can try with I did not go for the details of all the digits but I have listed whatever the other visits can give the output there, so this is the first visit we have already checked it. Now, here the second visit, second visit basically according to this one, we have to visit left sub tree first then visit root then right sub tree is there, so in this order, if we follow then you will see this is the expression that we can get it, sorry, sorry, so this is the expression that we can get it. So, this is according the second visit.

Now, again if we apply third traversals, I mean this one this one, so here you see first we visit left sub tree then we visit right sub tree and then we visit root and then again we repeat the same

thing again for each sub trees. So, if we follow this kind of order ordering in traversals then it will give the output this one. And so, this way if you continue others, so these are the different output that it will give it. Now, let us summarize all that traversals that we obtained according to the six different ways that the tree can be visited.

(Refer Slide Time: 10:43)

Example: Different ways of traversing

Visit 1: $+ \cdot A B * C / E F$

Visit 2: $A B + C * E / F$

Visit 3: $A B - C E F / * \downarrow$

Visit 4: $F E / C * B A +$

Visit 5: $F / E * C + B A$

Visit 6: $\downarrow * / F E C - B A$

Observation:

- Visit 1 and Visit 4 are mirror symmetric.
- Similarly, Visit 2 with Visit 5
- Visit 3 with Visit 6.

So, out of six possible traversals, only three are fundamental, they are categorized as given below:

1. $R \ T_1 \ T_1$	Pre-order traversal
2. $T_1 \ R \ T_1$	In-order traversal
3. $T_1 \ T_1 \ R$	Post-order traversal

NPTEL Online Certification Courses
IIT Kharagpur

Example: Different ways of traversing

Visit 1: $+ \cdot A B * C / E F$

Visit 2: $A B + C * E / F$

Visit 3: $A B - C E F / * \downarrow$

Visit 4: $F E / C * B A +$

Visit 5: $F / E * C + B A$

Visit 6: $\downarrow * / F E C - B A$

Observation:

- Visit 1 and Visit 4 are mirror symmetric.
- Similarly, Visit 2 with Visit 5
- Visit 3 with Visit 6.

So, out of six possible traversals, only three are fundamental, they are categorized as given below:

1. $R \ T_1 \ T_1$	Pre-order traversal
2. $T_1 \ R \ T_1$	In-order traversal
3. $T_1 \ T_1 \ R$	Post-order traversal

NPTEL Online Certification Courses
IIT Kharagpur

Now, here is basically the different summaries that we have listed here. So, these are the 6 different visits that we have already done and here you can see the visit 1 and visit 4 namely, visit 1 and visit 4 namely are basically same, but it is basically one is replica of others in the

sense that one is mirror reflection of the others, so it is basically if we take the reflection of this here, so basically the two things are same.

So, two things are same from either front to end or end to front in that sense likewise we can see the visit 2 and visit 5, if you check this, these two things are again mirror replica and finally this is the visit 3 and visit 6 are also mirror replica. So, what we can understand is that although there are 6 different ways that we can visit trees, but they are ultimately 3 are the actually because they can make sense and other 3 are redundant.

So, we can say that visit 1 then visit 3 and visit 6 or we can alternately 1, 2, 3, whatever you the way you can say let it be 1, 2, 3 like, so these are the 3 different what is called the traversals those are meaningful. Now, again let us come to this 3-traversal little bit closely, then we can see it. If we consider the first way of ordering then it basically leads to pre-order traversal and because this gives you an expression in the prefix notation.

Now, this is on the other hand, it gives the regular expression and it is called the in-order or infix notation and this traversal may be termed as in-order traversal and this type of traversals if we apply to an arithmetic representation of a binary tree we need to give you the postfix notation and this traversal can be termed as post-order traversal.

So, in summary what we have learned we learned that if given a binary tree, it can be visited in three different ways producing the three different outputs actual, but whatever be the ways that you follow it will visit all the nodes without fail. And what are the three traversals? They are called pre-order, in-order and post-order traversals. Now, so this is basically the different ways that a binary tree can be traversed or can be visited.

(Refer Slide Time: 13:33)

Pre-order binary tree traversal

Order of traversal:

- Visit the root node R .
- Traverse the left sub-tree of R in preorder.
- Traverse the right sub-tree of R in preorder.

Algorithm Preorder (ROOT)

1.	ptr ← ROOT	// Start from the ROOT
2.	If (ptr ≠ NULL) then	// If it is not an empty node
3.	Visit(ptr)	// Visit the node
4.	Preorder(ptr → L)	// Traverse the left sub-tree of the node in preorder
5.	Preorder(ptr → R)	// Traverse the right sub-tree of the node in preorder
6.	EndIf	
7.	Stop	

NPTEL Online Certification Courses
IIT Kharagpur

Now, here I just try to write the algorithm for that, all those algorithm if you note they can be better defined recursively, because it is in a recursive nature, if you have to visit left sub tree that mean visiting the same fashion as the original tree to be visited and sub sub tree is the same thing the way the sub tree is visit like this one. So, these algorithm can be express in a English like language it, visit the root note R , this is I am talking about pre-order traversals.

Then traverse the left sub tree of R that is in the same order it is in pre-order and then right sub tree of R in pre-order. So, is R TL TR and this is the algorithm that you can follow in order to realize this what is called the procedure and programming can be done if we follow this concept it is there. So, this is about pre-order traversal, likewise we can extend this concept for other two traversals namely in-order and post-order.

(Refer Slide Time: 14:37)

Pre-order binary tree traversal : Example

Order of traversal:

- Visit the root node R .
- Traverse the left sub-tree of R in preorder.
- Traverse the right sub-tree of R in preorder.

Algorithm Preorder (ROOT)

1.	ptr ← ROOT	// Start from the ROOT
2.	If (ptr ≠ NULL) then	// If it is not an empty node
3.	Visit(ptr)	// Visit the node
4.	Preorder(ptr → L) // Traverse the left sub-tree of the node in preorder	
5.	Preorder(ptr → R) // Traverse the right sub-tree of the node in preorder	
6.	EndIf	
7.	Stop	

The diagram shows a binary tree with root A. A is the left child of B, and B is the left child of D. D is the left child of G. G is the left child of E. E is the left child of C. C is the left child of F. F is the right child of G. A sequence of nodes visited in preorder is shown as A, B, D, G, E, C, F.

Now, again here is an example if we apply this algorithm we will be able to see it is basically pre-order so visit root, so root is visited, then here we have to visit left sub tree so that means we have to visit this one, once we complete these visit then we will go to this one. Now, here coming to this one again to visit left sub tree means we have to visit B and then we have to come to visit this one.

So, it is B is visited and then come here there is no so come means we have D first, so D is visited and this part does not have any left sub tree, so next is we have to visit G, so D is visited and then finally G. And this completes the visit of this part then we have to finally visit E, after visiting this one then you have to visit E.

So, this way it basically completes the visiting all the nodes in the left sub tree of the root. Likewise if we continue next right it is a right sub tree and then right sub tree means we have to come here, so visit C first, then this part left sub tree it does not have any left sub tree so we have to stop it here, then come here then F and it completes.

So, this is the order that we can follow according to this technique actually. So, root is visited then left sub tree and then right sub tree. Now, when we visit left sub tree it is in the same order that we have to be visit it. So, this is basically the idea that you can have and then according you can see this. So, you can take any other binary tree and then apply this algorithm to practice

yourself and how it is coming and then you can check it, so that it will give it. Now, so this is the pre-order traversals, let us proceed further for other traversals.

(Refer Slide Time: 16:32)

In-order binary tree traversal: Example

Order of traversal:

- Traverse the left sub-tree of R in in-order.
- Visit the root node R .
- Traverse the right sub-tree of R in in-order.

Algorithm Inorder (ROOT)

```

1 ptr ← ROOT // Start from ROOT
2 If (ptr ≠ NULL) then // If it is not an empty node
3   Inorder(ptr → L) // Traverse the left sub-tree of the node in in-order
4   Visit(ptr) // Visit the node
5   Inorder(ptr → R) // Traverse the right sub-tree of the node in in-order
6 EndIf
7 Stop
  
```

The diagram shows a binary tree with root A. A has left child B and right child C. B has left child D and right child G. G has left child E and right child F. A sequence of nodes visited in in-order is shown at the bottom: D, G, B, A, C, E, F.

Our next traversal is called in-order traversal, as the in-order traversal says that the root will be visited in between the left sub tree and right sub tree that is why it is call in-order. Now, so here we first visit the left sub tree, then visit root node and finally visit the right sub tree. And here is a simple algorithm that we can follow written recursively that this procedure can be implemented. Now, here let us have some examples, so that we can idea about it.

Here is a example. In this example as we see the same tree again. Now, what is the procedure? So, we have to visit this tree, so in order to visit these tree, we should visit this first, because it is a procedure that visit the left sub tree, then we will come back to visiting this one and then finally this one. Now, while we are visiting this part, again we have to visit in the same fashion, so visiting these part mean we have to visit this part first, then we will come here and then we will visit it one.

Now, again visiting this part means again we have to visit this part, but here is... no node, so we have to visit D. So, first node that we will be visited is D, then we have to come to G, so G is visited, now this part is completely visited then we will visit B, then B is visited, once B is visited then our next task is to visit right sub tree, so E visited. So, this way it completes the left sub tree visited. So, this is basically visit of the lift sub tree up to this part. Now, we have to visit

the next part, now visiting means so now this time left sub tree of R is in order visited next come to the A, so visit A once it is visited then we have to visit right sub tree.

Now, visiting rights sub tree again in the same fashion, first left sub tree it does not have any elements, so nothing, then we have to visit C, so visit C and finally come to the right sub tree it has only one element so F. So, these basically right sub tree of the root, left sub tree and this is the order that we can find it. So, this is basically the in-order traversal for a given binary tree and this is an example.

(Refer Slide Time: 18:53)

Post-order binary tree traversal

Order of traversal:

- Traverse the left sub-tree of R in postorder.
- Traverse the right sub-tree of R in postorder.
- Visit the root node R.

Algorithm Postorder (ROOT)

```

1  ptr ← ROOT // Start from ROOT
2  If (ptr ≠ NULL) then // If it is not an empty node
3      Postorder(ptr → L) // Traverse the left sub-tree of the node in postorder
4      Postorder(ptr → R) // Traverse the right sub-tree of the node in postorder
5      Visit(ptr) // Visit the node
6  EndIf
7  Stop
    
```

The diagram shows a binary tree with root A. A's left child is B, and A's right child is C. B's left child is D, and B's right child is E. D's left child is G. E's right child is F. The nodes are visited in the order G, D, E, B, C, F, A, as shown in the sequence at the bottom of the diagram.

Now, there is another traversals this is the last traversal approach it is called the post-order traversal, it is called the post in the centre root will be visited at the end that mean we have to visit left sub tree then we should complete the visit of all nodes in the right sub tree and then we will come back to the root to visit it.

And this is the algorithm that you can follow, so these are the step here left sub tree then the right sub tree and finally root node. And the algorithm recursively defined in order to implement this procedure is mentioned here. You can simply follow this one and write a code, so that will not be a big job. And now let us come to the example.

So, again this example that we can follow for the same tree. Now, here we have to lift sub tree, that means we visit this part, now here if we see if we follow the visiting of left sub tree in the same fashion, now here again to visit this one we have to visit left tree, now to visit the left sub

Now, let us proceed further. So, we have learned about the different tree traversals and now let us see the insertion operation. Insertion operation means given a binary tree we have to insert a new node into it.

(Refer Slide Time: 22:02)

Insertion operation

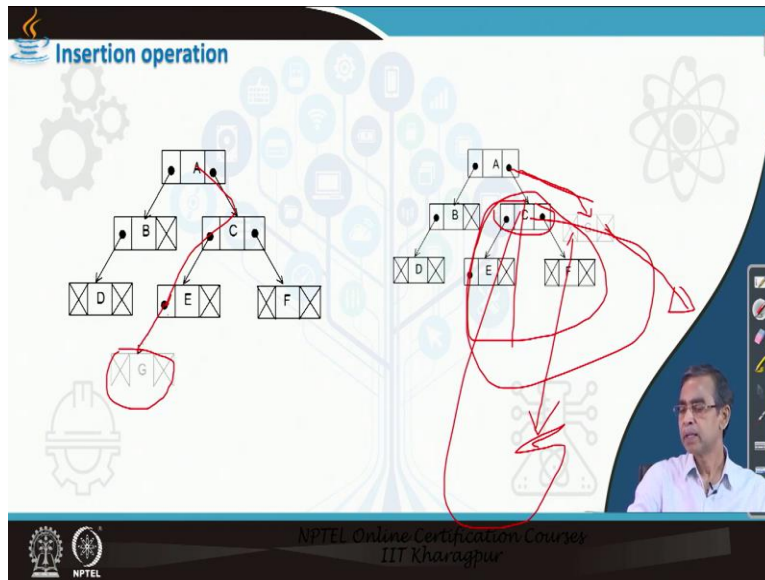
- Insertion procedure, in fact, is a two-step process:
 1. To search for the existence of a node in the given binary tree after which an insertion to be made, and
 2. To establish a link for the new node.

NPTEL Online Certification Courses
IIT Kharagpur

So, there are two step actually insertion procedure first, first of all, I told you once that a binary tree usually contains non-duplicate elements in it, because duplicate elements there is no meaningful things are actually there, so we assume that all nodes are distinct there. Now, whether so we have to insert first we have to check whether I mean whether the node is already exists or not although it should not be but this is a customer rechecking that okay, we check that if the node is exist would then definitely we should not go for insertion.

Now, so first we have to search that if the node exist in the binary sub tree or not. If exist then we do not have to do anything, if it does not then what we will have to do is that where the search end there actually we have to insert the node the new node to be inserted. Now, the question is that how we can see that whether our searching end or not, searching will end where actually. So, this can be explained with an example.

(Refer Slide Time: 23:12)



And here is an example for example, say suppose we want to insert G and we found that G should be inserted as the left child of this node, now so what you can do is that we can start from here we can come here and we reached here and then this basically is an empty, so we just store the address of this node in this field, this means that it will establish a link from this field to this node actually, so this way new node will be inserted.

Now, let us come back to another approach saying we want to insert not here but we want to insert where C is present, so we want to insert G here where C located, so what you can do is that we can just push these things little bit one and then these can be made link that means this A the right part of this link will store the node G and then either these part or these part anyone part we can be at address to the C. So, this means that it will point this one and this will point this sub tree here.

So, this way the node can be inserted, here anywhere actually in the tree. But here again problem you can say that, no no, we will not insert as a left sub tree, we can insert the right sub tree, that also possible, we can insert we can make a link from this field to the node C which basically store the left right part of the sub tree of the node.

So, this way whatever be the technique that you can follow so insertion procedure can be like this. But as you know there is an ambiguity or little bit what is call not safe (())(25:03) precise

rule, but without any precise rule the algorithm cannot be implemented, so we should follow some standard method of inserting a node into a binary tree.

But while inserting a binary tree standard method imposing a standard method is really not so easy job, so we have to think something else that we will discuss how a precise rule can be formulated, so for the insertion is concerned so that an anyway so that it is uniformly insert into a same location all the time, that means there should not be any ambiguity. So, this is about the insertion procedure that we have learned about.

(Refer Slide Time: 25:47)



Now, next is deletion operation.

(Refer Slide Time: 25:50)

Deletion operation

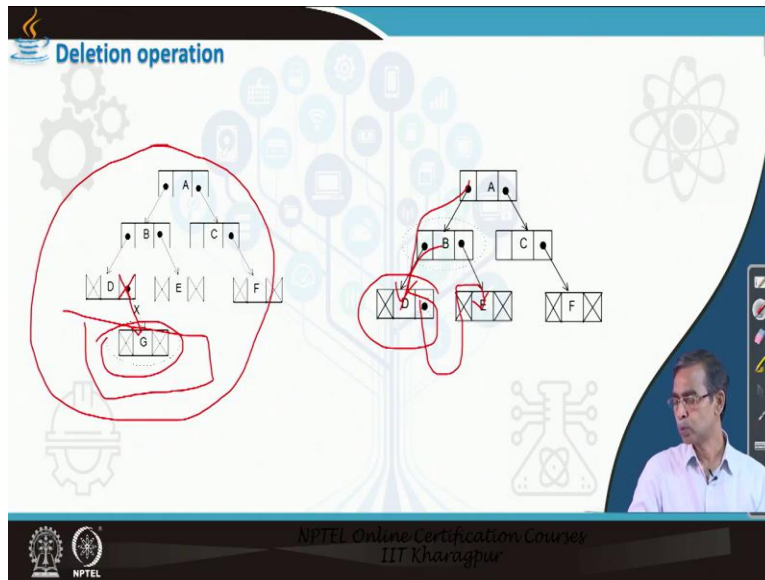
- Like insertion procedure, deletion is also, a two-step process:
 1. to search for the existence of a node in the given binary tree, which node to be deleted
 2. to adjust the links among parent and child nodes of the deleted node.

NPTEL Online Certification Courses
IIT Kharagpur

Now, deletion operation like insertion is a two-step procedure. So, first of all we have to check that whether the node exists or not, because the element that you want to delete that may not be available in the binary CST, if it is not there or if binary tree is empty whatever be the situation then in that case your deletion will fail, rather you do not have to do anything.

On the other hand, if you find the node that it is present there then the next step is basically manipulate the link that means you have to just remove this node and then for all other nodes should not be lost their link so you have to update the links. So, this is the procedure that you can follow.

(Refer Slide Time: 26:32)



Let us, have an example so that we can understand the arithmetic method of deletion. Now, suppose this is the input tree and we want to delete this node G, so simply what is the matter of deletion is that this is a leaf node and deleting a leaf node is easy, so they are basically the link which basically stores in his parents needs to be updated. So, that means that initially it will give the link to this node. Now, we have to make a null into this field, this means that it is deleted from the node.

So, on the other hand if suppose we have to delete B here, now deleting B is not that same procedure, rather what we have to do is that we have to update many links in many nodes there actually. So, if we delete B then we have to maintain D E so many things are there. So, maybe that okay, once we delete B that means D can be made it in this way, so this basically linked that D when whatever the link because B is deleted so this information should be stored, so this information then I can store here.

Again, ambiguity is there, why it is D? Why it is not E? So, E can be replaced B and in that case again the link updation is possible. So, again the ambiguities are there. Now, again in order to resolve the ambiguity we have to decide a certain protocol or policy or a species rule and then that all those things ambiguities can be avoided.

(Refer Slide Time: 28:02)

The slide features a central title "Merging Operation" in blue text. To the left is an icon of a coffee cup with steam. The background is a light blue grid with various icons like gears, a tree, and a hard hat. A small video inset in the bottom right shows a man in a white shirt speaking. At the bottom, there are logos for NPTEL and IIT Kharagpur.

Now, let us come to the another operation merging operation. As I told you merging operation means we have to combine two binary tree into a single binary tree, sometimes it is very useful in many application that we have given two or more trees and we have to combine all trees into one, can making a larger trees like.

(Refer Slide Time: 28:30)

The slide is titled "Merging operation" and contains a bulleted list:

- Another fundamental operation that is possible is the merge operation.
- This operation is applicable for trees which are represented using linked structure.
 - There are two ways that this operation can be carried out.

A text box on the left explains: "Suppose, T_1 and T_2 are two binary trees. T_2 can be merged with T_1 if all the nodes from T_2 , one by one, is inserted into the binary tree T_1 (insertion may be as internal node when it has to maintain certain property or may be as external nodes)." To the right, two binary trees are shown: T_1 with root A and children B, C, D; and T_2 with root X and children Y, Z, W. A red box highlights the nodes of T_2 and a red arrow points from the text box towards them.

So, for this purpose the merge operation can be applied, so I can explain the merge operation how it looks like, now so here is basically consider this is the one tree T_1 and this is another tree T_2 , now merging can be done in many ways actually, now here the merging can be done for

example, I can merge all the nodes of this T as a because it has the two links, so we can make either anyone to this link, so this way it can be merged easily. So, we have to check that whose link part is still null, then we can make the root of the another tree as a link and this way it can be.

So, this is true for here or here or there anywhere because there are many link null link fields are there so we can do it. But again here if you see there are many way merging is possible, it means the operation is again ambiguous. Now, we have to make certain rule again precise rule that how the ambiguity can be resolved, so that it can give unit tree always whenever we perform merging for the same set of trees.

Another approach that can be followed is that, that approach is basically we can remove one note that mean delete one node starting from the root node and inserting the same node into another tree. So, this way that is also but it is again costly appear because we have to delete all nodes in a one tree and then insert that node into the same tree, so altogether there is a huge number of both deletion and insertion operation takes place, compare to the previous approach that we have given it is easy actually just simply managing one link that is all.

(Refer Slide Time: 30:30)

Merging operation

- Another fundamental operation that is possible is the merge operation.
- This operation is applicable for trees which are represented using linked structure.
 - There are two ways that this operation can be carried out.

Another way, when the entire tree T2 (or T1) is included as a sub-tree of T1 (or T2). For this, obviously we need that in either (or both) tree there must be at least one null sub-tree. We will consider in our subsequent discussion, this second case of merging.

The diagram shows two trees, T1 and T2, with nodes labeled M1 through M10. A red circle highlights a portion of the merged tree structure, showing how one tree is integrated into another.

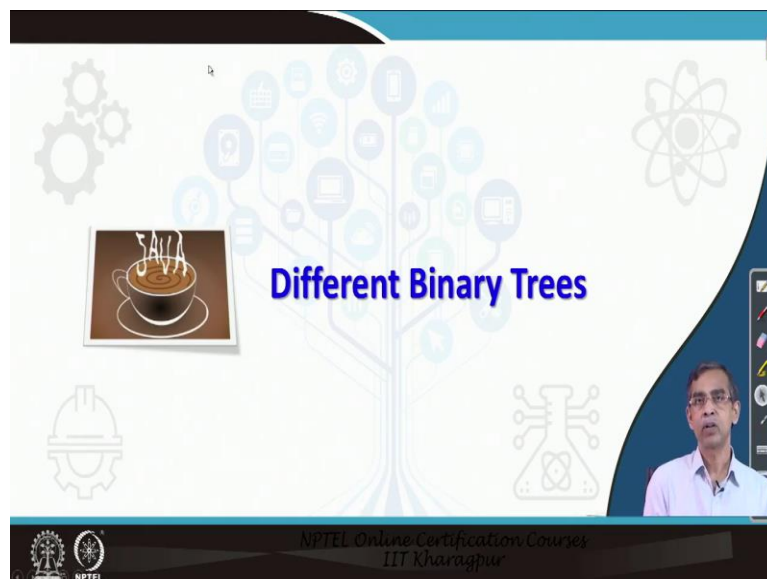
NPTEL Online Certification Courses
IIT Kharagpur

So, there is again trade-off of that which method we should follow and depending on these things the merging operation can be made either very expensive or it is less expensive depending on what protocol what procedure you follow. So, if suppose here root node does not have left sub

tree link left part, then we can easily combine it and then this way the merging can be done. So, here for example, so this is the $T_1 T_1 T_2$ and fortunately if we see that the root node does not have only right link part so I can make the link to the next node.

And this is the resultant T that you can get it. But again, you have to be fortunate that this root node does not have either left or right link there. So, there is a many situation are there and then it is the task of the computer scientist to decide very nice approach there, so that all those problems can be address later. And we will see how this problem can be address if so far the T data structures is concerned.

(Refer Slide Time: 31:34)



Actually, to address all those problems regarding insertion deletion regarding merging even insertion deletion whatever we have mentioned here that in order to resolve these things scientist, they propose different form of the binary term, each form will follow certain specific properties.

(Refer Slide Time: 32:01)

Different binary trees

There are several types of binary trees possible each with its own properties. Few important and frequently used trees are listed as below.

1. Expression tree
2. Binary search tree
3. Heap tree
4. Threaded binary tree
5. Huffman tree
6. Height balanced tree (also known as AVL tree)
7. Red black tree
8. Splay tree
9. Decision tree

NPTEL Online Certification Courses
IIT Kharagpur

Different binary trees

1. Expression tree
2. Binary search tree
3. Heap tree
4. Threaded binary tree
5. Huffman tree
6. Height balanced tree (also known as AVL tree)
7. Red black tree
8. Splay tree
9. Decision tree

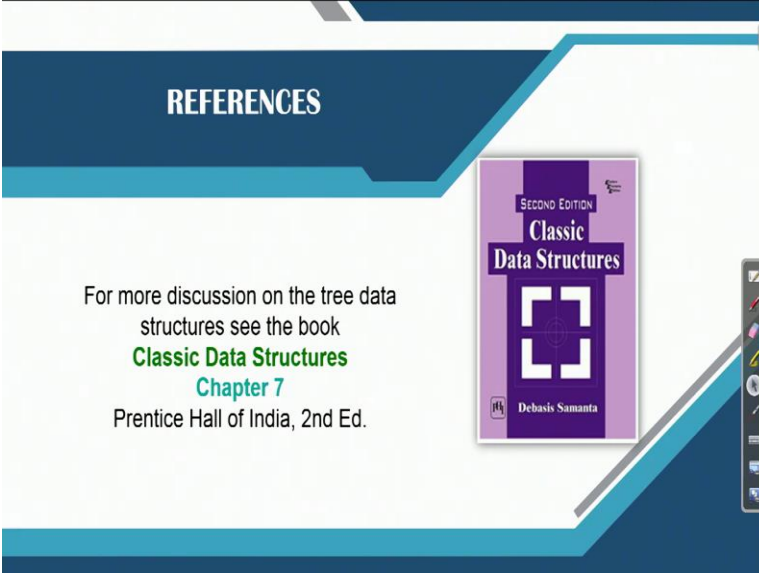
NPTEL Online Certification Courses
IIT Kharagpur

Now, based on this certain specific properties, there are numerous types of trees there, I have listed only nine different type of trees, but in this lecture it is not possible to discuss all the binary trees that I have listed here. But there are few binary trees which are very important and most frequently used and it is a very interesting also, I have listed four such binary trees like binary search tree then heap tree and then Huffman tree and another is called the height balanced tree.

Height balanced tree is also called AVL tree, each tree has their own what is called the unique characteristics their advantages some tree have their own merits demerits, there is a pros and cons whatever it is there. So, in our next video lectures one by one we will try to understand

about four different tree and then the operations that we have discussed all these operation in the context of specific trees and finally there programming.

(Refer Slide Time: 32:59)



The slide features a dark blue header with the word "REFERENCES" in white. Below the header, on the right side, is a book cover for "Classic Data Structures, Second Edition" by Debasis Samanta, published by Prentice Hall of India. The cover is purple and white with a stylized tree logo. To the left of the book cover, the following text is displayed: "For more discussion on the tree data structures see the book **Classic Data Structures** Chapter 7 Prentice Hall of India, 2nd Ed." The slide also includes a vertical toolbar on the right edge with various icons for navigation and editing.

Now, if you want to understand about other binary tree other type of binary tree, then I will recommend you to follow this book chapter 7, where all such trees are discussed in details with their operations algorithm there characteristics and applications. So, we have learned about the operations on binary tree as a whole or in general, but in our next lecture few lectures we will try to understand the operations on binary tree again with reference to a specific type of binary tree. Thank you very much.