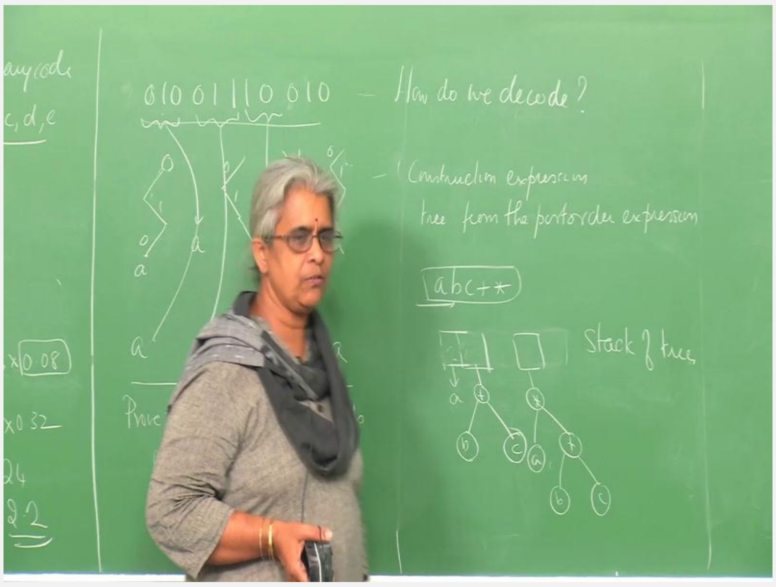
Programming, Data Structures and Algorithms
Prof. Hema Murthy
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Lecture – 49
Module – 09
Other applications: expression tree from
The post-order expression
Traversal of binary trees: preorder, postorder, inorder;
Illustration with expression tree and infix, postfix forms

There are many other applications like constructing the expression tree from the postorder expression. I leave you with an idea as how to do it.

(Refer Slide Time: 00:12)

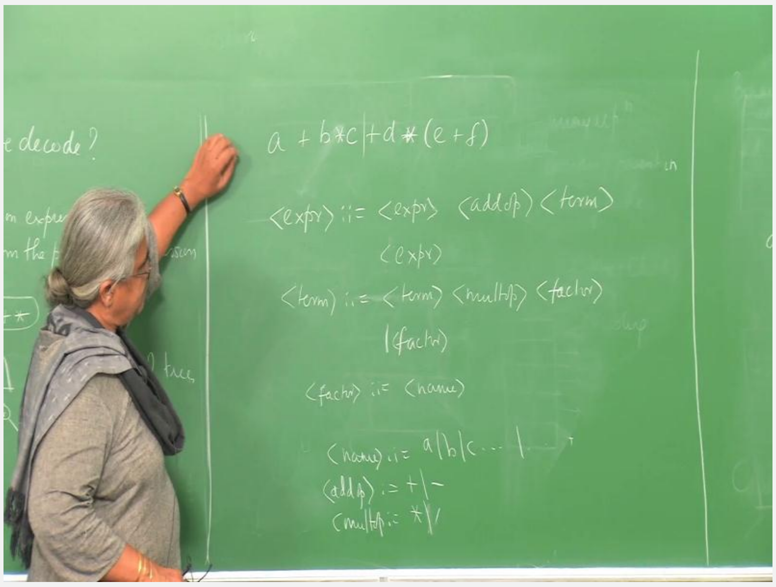


Suppose, I have a b c plus and star, what we do is, when we have symbols, first we have again a forest of trees pointing to a, this pointing to b, this pointing to as use, you traversae from the left to right of this expression. As you move from left to right of this expression what you do, you just create pointers to trees in a forest. Then, what you do is you come across plus, then you create a new node with plus and then, you take the top most two, it is like a stack, it is actually like a stack.

Then, you it is actually a stack of tress. What we do is now I put these from left to right, I put the trees a, b and c. And then, as soon as I come across the operator, what do I do, I pop the most recent two trees, then I create a new tree with the plus and I make the first tree the left child, the most recently popped out tree is the left child and the previously popped out tree be the right child.

Now, again I have one more symbol left, I come across the star. So, I pop out both the trees and I create put on to the stack, the star and the two pop trees become the, most recently popped tree becomes the right child and the most recently popped tree becomes the left child and the previously popped tree becomes the right child and as soon as, until you get a single tree which exists on the stack. So, this is the other application of binary trees and stacks. Both of them can be used quite efficiently to build what are called expression trees. So, this is a very, very useful application of binary trees.

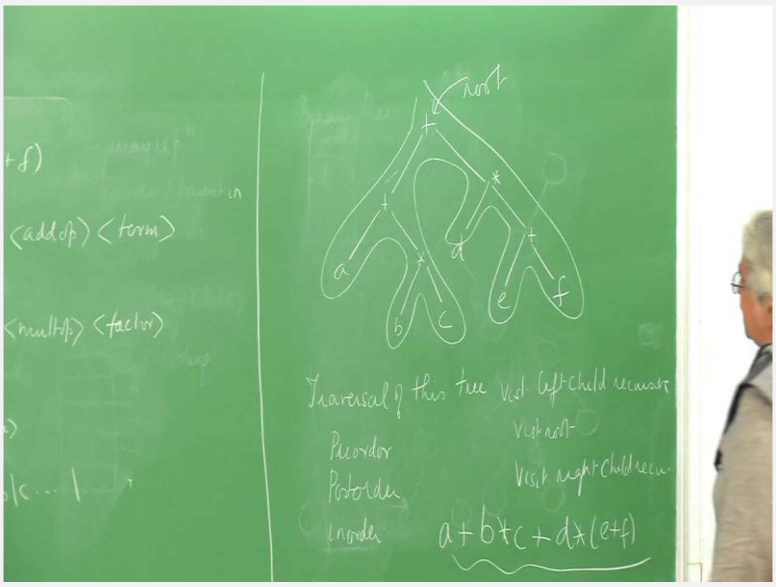
(Refer Slide Time: 03:07)



Now, another thing we always like to do when we build binary trees. Let us take this expression tree itself, let us look at an expression. Let us say we have something like this, then what we are doing the expression is evaluated,. So, it is evaluated from left. So, let us say I have this here plus star b c actually go over this from the right and let me. Let us say I form an expression tree, I am not known how many of you are familiar, we use this kind of a expression parsing to do this expression form or expression and term is form or a factor and let us say factor is the name as it is what it mean, let us say this is a, b, c and. So, on here.

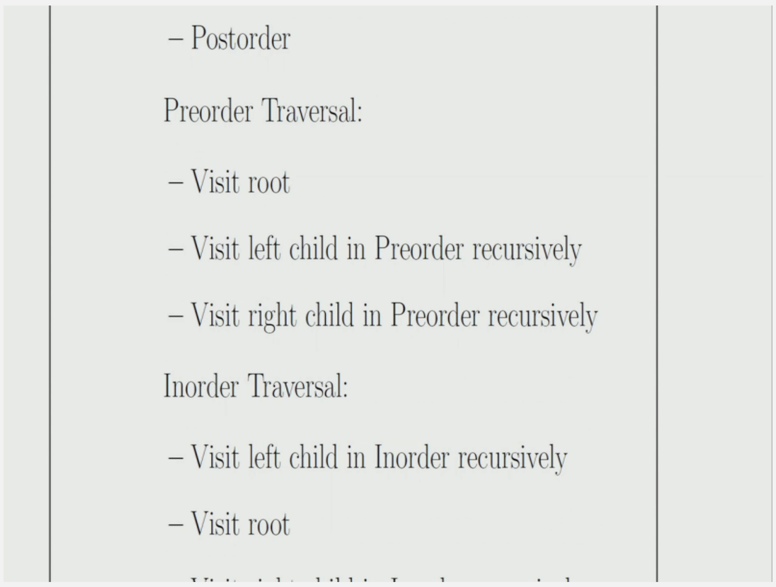
Suppose, I use this grammar to generate this expression, we have already done a course on compilers. Let us see, this is completely an unambiguous grammar, we add up here is plus or minus mult or star or slash, let us say. So, how do we parse this given expression? So, we start from the root over here, expression is defined as expression add operator terms, I start it over here.

(Refer Slide Time: 05:48)



Plus, then what do we have, then on the left hand side, then the other plus again add the plus, then I will get a here, I will get star b c and star b plus e f, this is how you will get a binary tree. What I have done is, all the exercise that I have done is to just show that the expressions, how many expressions that I have written like this, it can be written as the binary tree. Now, what we will do is we will talk about other things now. We will talk about traversal of this tree. Let us see what it gives us. There are three types of traversal preorder, postorder and inorder. Now, the definitions are given over here, how to traverse these trees.

(Refer Slide Time: 07:06)



– Postorder

Preorder Traversal:

– Visit root

– Visit left child in Preorder recursively

– Visit right child in Preorder recursively

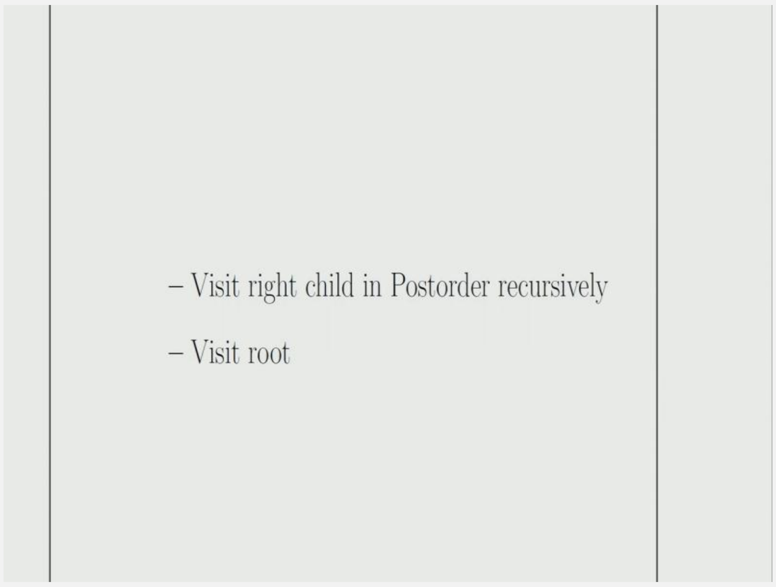
Inorder Traversal:

– Visit left child in Inorder recursively

– Visit root

So, preorder traversal says visit the root, then what you do, you visit the left child in preorder recursively, visit the right child in preorder recursively. It means what now, let us take the…Then, the inorder traversal says visit the left child in inorder recursively, visit the root this is the right child in inorder recursively. Then, postorder traversal says visit the left child in postorder.

(Refer Slide Time: 07:39)

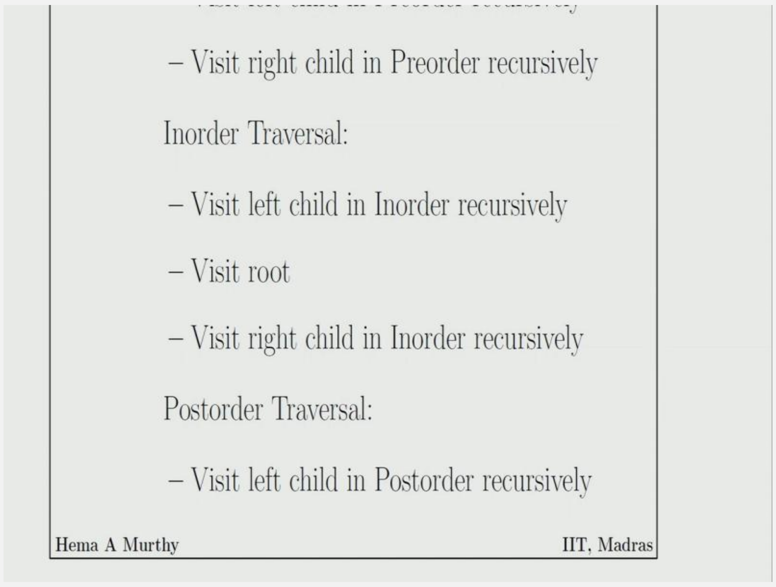


– Visit right child in Postorder recursively

– Visit root

And visit the right child in postorder recursively, visit the root, got it. Let us see what this gives,. So, everything has to be done recursively. What is the meaning of this?

(Refer Slide Time: 07:52)



– Visit right child in Preorder recursively

Inorder Traversal:

– Visit left child in Inorder recursively

– Visit root

– Visit right child in Inorder recursively

Postorder Traversal:

– Visit left child in Postorder recursively

Hema A Murthy IIT, Madras

Let us look at inorder traversal, visit the left child in inorder recursively. Let us say, I want to do this traversal of the tree, traversal means simply walking around to it. Thus the way the listed node is got changes. Now, what it says let us look at inorder traversal, it says first we look at inorder, because it will become clear to you what we are doing. Visit left child recursively, visit root and visit right child recursively is what here inorder traversals.

So, let us look at the tree what could we do. So,; that means, I am starting from the root here and I am going to recursively keep on going down to this until I reach the leaf node. Then, what do I do, I display the leaf node. So, I keep going down, because I have to visit the left most child. Then, what happens I go up, then I visit the root and get plus, then again I go to the right child and again after recursively do the same thing; that means, what preorder, inorder all these are recursive traversals.

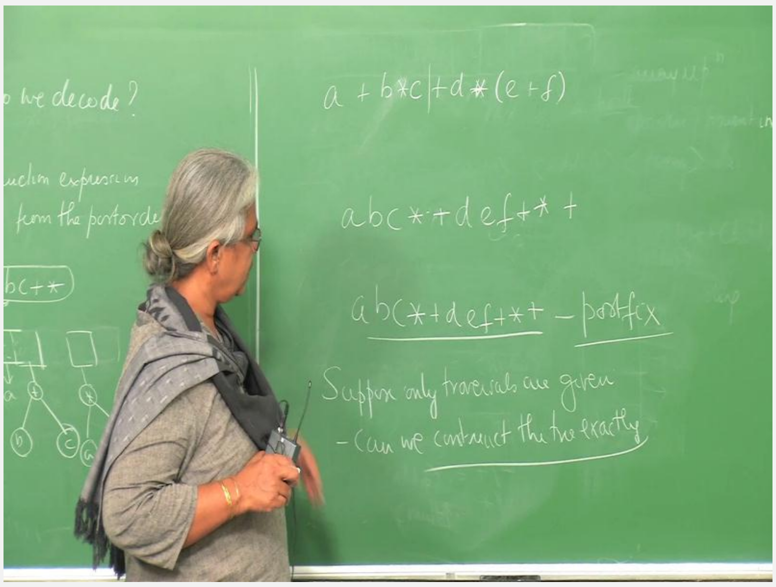
So, I go to the right child, then again I will do left child recursively root, then then right child and then go back. Then, what happens here,. So, I come down this plus go down here, this becomes b star c and then what happens, this is node as we already been visited,. So, I go back up to the root of the tree, b may plus, then again I come to the right node star, again I have to do the recursively left child. So, I do d, I come to the leaf node star, then I have to add e plus f, the bracket has to come, because it is one level lower.

So, inorder traversal essentially gives me the expression back. So, what is it tell me now, may I look at the inorder traversal of the tree, I get the infix expression of the given tree. This one extra information that we have used here, we use the factor star is done after plus, because it is one level lower in the tree,. So, I get the infix expression. So, let us look at what postorder expression, postorder traversal of the tree will give you.

What does postorder traversal say, ((Refer Time: 10:55)) postorder traversal says visit left child postorder in postorder recursively, then ((Refer Time: 11:03)) visit right child in postorder recursively, visit root and the same thing keeps on getting repeated again and again. So, I take the left child recursively visit in postorder, let me just explain this over here. Actually, what is mean to us, what does it gives you,. So, I am moved here, visit the left child, I cannot came down, there are no more leaf nodes,. So, I display a.

Then, I have to go to the right child, go back left child, right child and then root is what it says, go to the right child. Then, right child again I have to recursively go down, do postorder, left child, right child, then root.

(Refer Slide Time: 11:45)



we decode?
a + b*c + d*(e+f)
abc*+def+*+
abc*+def+*+ – postfix
Suppose only traversals are given
– Can we construct the tree exactly

So, what will happen, I get a b c star, then, what do I get, now all of them have been visited, I go back up,. So, left child, right child, then root get plus here and then, what happens. So, the left child of this root tree is completed now,. So, I go to the right tree over here. Then, what do I get now, what do I get here now, I have get d, then again this is the left child, this is the leaf node, there is nothing else to see, therefore, I display the leaf node.

Then, I go to its right child, again I have to look at postorder over there. Therefore, I get e f plus plus e left child postorder, right child postorder, root then I will put the plus over here that is how it comes here and then finally, this star and then finally, the plus, the last plus sign, that we have. So, I am made a mistake here, this plus should not be here. See, a b c star, then I have to do the right child, d e f plus star plus. So, what did we get now, we got a b c star plus correct, d e f plus star plus. How many are there? 1, 2, 3 correct.

So, this is what we get as the expression. What is the expression now, this is nothing,, but the postfix expression which all of us have already been done. When we talked about stack, stacks we talked about generating the postfix expression from an infix expression. So, what is interesting here is once it is represented as a binary tree, if I do inorder traversal I get the infix expression, when I do postorder traversal I get the postfix expression and obviously, if I do preorder traversal I will get the prefix expression.

So, this is also a very big application of binary trees. Now, once this expression is given like this, I can use this stack of trees to evaluate this particular expression. So, I hope you are with me here and you have understood what I have said. So, what is it we are doing notice that neither we have looking at this various traversals to make it simpler is preorder traversal can be thought of as, you display the node the first time you visit it.

In inorder traversal, you display the node the second time you will visit it. Remember, I have drawn this path over here which goes around the tree and in postorder traversal you display the node, the last time you visit it. So, these are simple traversals schemes. So, now the next question that we would like to ask is suppose you are given only the traversals, traversals are given, can we construct the tree exactly?

Let us look at only with respect to binary trees and see if I am given the, what is that we have two traversals now, inorder traversal and we have the postorder traversal. We can also do the preorder I encourage you to do the preorder traversal of the tree. Now, given the inorder traversal can I construct the tree uniquely without any other information and given the postorder traversal, can I construct the tree uniquely, let us see how you will do this in the next lecture.