Traversal in Binary Tree (InOrder, PostOrder and PreOrder Traversals)

codewithharry.com/videos/data-structures-and-algorithms-in-hindi-66

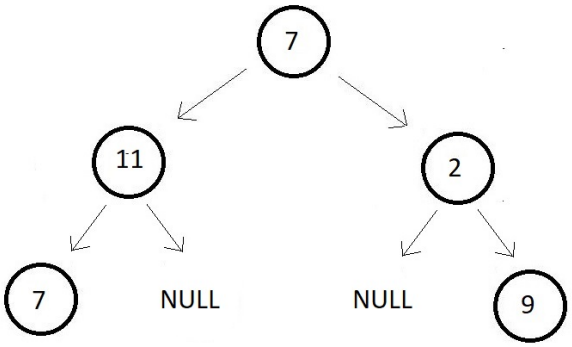
In the last tutorial, we learned to implement the linked representation of binary trees in C. We saw how easy it was for us to visualize the elements decorated as a real tree when implemented using linked representation. Today, we'll learn how to traverse in binary trees.

Traversal in a binary tree:

If you remember the previous lectures where we saw the traversal in linked lists, traversal meant visiting every node in the list at least once. Traversal has several applications, one might want to visit each node and print their values, another might want to add each of those elements. It is, therefore, crucial to study the techniques of traversal.

But things were easy when we used to traverse through the lists, or the arrays, since they were linear structures, and the directions you choose to traverse were limited, either front to rear or from rear to front. But today we'll accomplish the complexity of traversing through a non-linear data structure, binary tree.

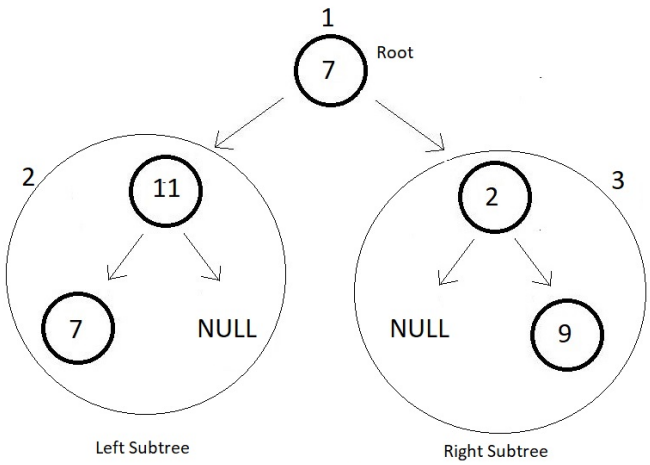
So basically, we have three different ways to traverse in a binary tree. They are InOrder, PostOrder, and PreOrder Traversals. Let's take a sample tree, and walk through it one by one, using each traversal technique for better understanding.



Let's start with the first one.

PreOrder Traversal in a Binary Tree:

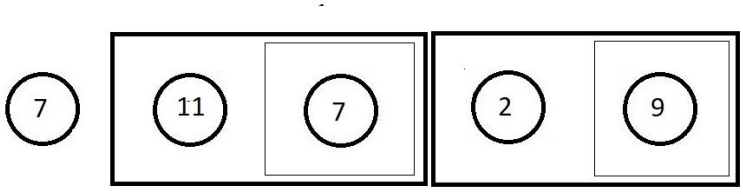
So in this traversal technique, the first node you start with is the root node. And thereafter you visit the left subtree and then the right subtree. Taking the above example, I'll mark your order of traversal as below. You first visit section 1, then 2, and then 3.



Now, this was to give you a general idea of the traverse in a binary tree using PreOrder Traversal. Each time you get a tree, you first visit its root node, and then move to its left subtree, and then to the right.

So, here you first visit the root node element 7 and then move to the left subtree. The left subtree in itself is a tree with root node 11. So, you visit that and move further to the left subtree of this subtree. There you see a single element 7, you visit that, and then move to the right subtree which is a NULL. So, you're finished with the left subtree of the main tree. Now, you move to the right subtree which has element 2 as its node. And then a NULL to its left and element 9 to its right.

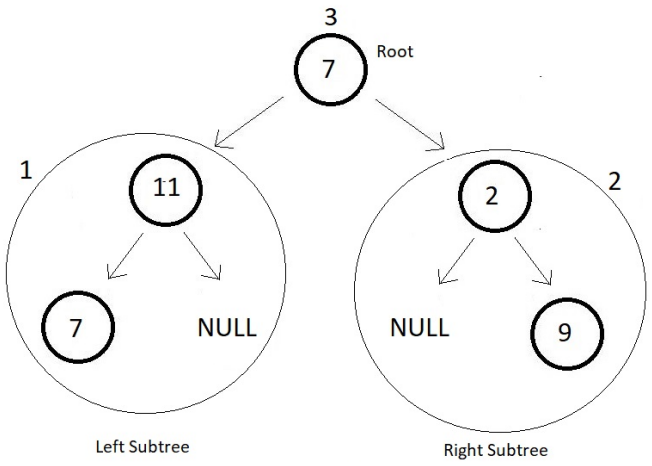
So basically, you recursively visit each subtree in the same order. And your final traversal order is:



Here, each block represents a different subtree.

PostOrder Traversal in a Binary Tree:

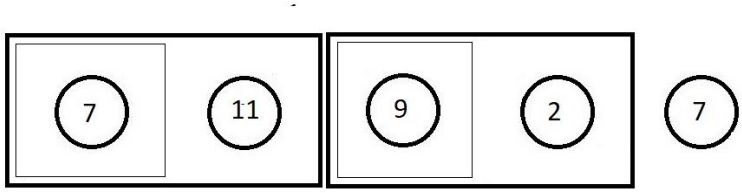
In this traversal technique, things are quite opposite to the PreOrder traversal. Here, you first visit the left subtree, and then the right subtree. So, the last node you'll visit is the root node. Taking the same above example, I'll mark your order of traversal as below. You first visit section 1, then 2, and then 3.



This was again a general idea of you traverse in a binary tree using PostOrder Traversal. Each time you get a tree, you first visit its left subtree, and then its right subtree, and then move to its root node.

I expect you to write the flow of the traversal in PostOrder yourself, and let me know if you could. We would anyway see them in detail.

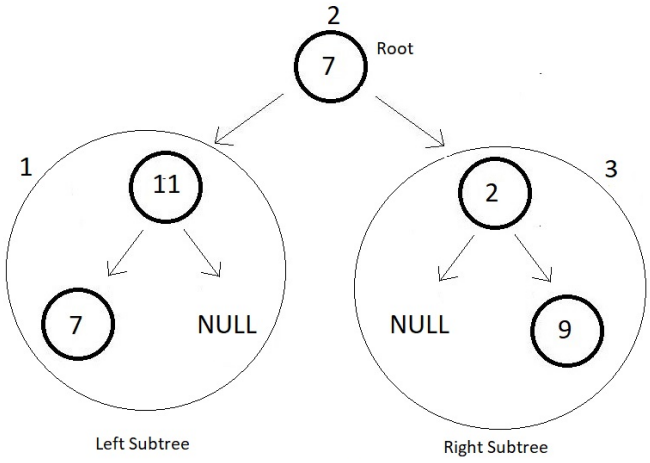
Hint: Your final traversal order should be.



Here, each block represents a different subtree.

InOrder Traversal in a Binary Tree:

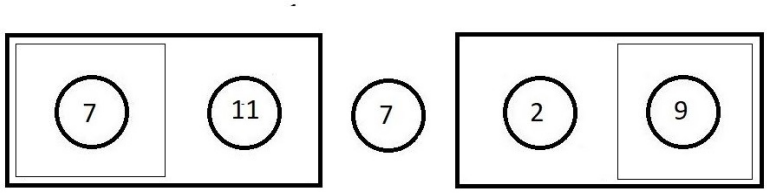
In this traversal technique, we simply start with the left subtree, that is you first visit the left subtree, and then go to the root node and then you'll visit the right subtree. Taking the same above example, I'll mark your order of traversal as below. You first visit section 1, then 2, and then 3.



This was a general idea of you traverse in a binary tree using InOrder Traversal. Each time you get a tree, you first visit its left subtree, and then its root node, and then finally its right subtree.

I expect you to write the flow of the traversal in InOrder yourself, and let me know if you could.

Hint: Your final traversal order should be:



Here, each block represents a different subtree.

So basically, the names PreOrder, PostOrder, and InOrder were given with respect to the position we keep our root node at while traversing in the Binary Tree. Since we first visit the root node in the first technique, that's why the name PreOrder. And we visit the root node at last in the PostOrder and in between in InOrder. Each of these techniques has one thing in common: the left subtree is traversed before the right subtree.

And these were the traversal techniques we had. We'll cover the programming part for each of these three in our coming lectures and in great detail. You shouldn't miss out on any of these. And do not forget the two techniques I left for you to explore.

I appreciate your support throughout. I hope you enjoyed the tutorial. If you genuinely appreciate my work, please let your friends know about this course too. If you haven't checked out the whole playlist yet, move on to codewithharry.com or

my YouTube channel to access it. See you all in the next tutorial where we'll learn to implement our first traversal technique in C, **PreOrder traversal**. Till then keep coding.