This is the third lecture on the topic of binary tree.

This lecture will explain how to print the entire binary tree as well as how to delete every node in a binary tree.

We want to visit every node in a binary tree. This is called to traverse the binary tree. Please notice the method explained here is applicable to any binary tree and not limited to binary search trees only.

To visit every node in a binary tree, we need to do three things: visit the node, visit the left subtree, and visit the right subtree.

Usually, the left subtree is visited before the right subtree is visited.

Thus, there are three possible orders of these steps.

If the node is visited before the left subtree and the right subtree, this is called pre order traversal.

If the left subtree is visited first, then the node, and the right subtree is visited the last, this is called in order traversal.

In the third scenario, the left subtree is visited first, then the right subtree, and the node is visited the last, this is called post order traversal.

Let’s go through an example.

First, let’s consider pre order.

To visit this tree, we write down the value of the root first. It is 27. Then, we need to visit the left subtree of 27 and the right subtree of 27.

To visit the left subtree, we follow the same rule. For the left subtree of 27, the root is 9 and it is visited first. Then, we want to visit the left subtree of 9 and the right subtree of 9.

To visit the left subtree of 9, we have to visit 3 first. 3 has no left child. The right child of 3 is 6.

The left subtree of 9 is 3, 6.

The right subtree of 9 is 15.

The left subtree of 27 is 9, 3, 6, 15.

Let’s consider the right subtree of 27. To visit the right subtree, we visit 38 first. Then, we visit the left subtree of 38, followed by the right subtree of 38.

To visit the left subtree of 38, we visit 32 first. Then, we visit the left subtree of 32, followed by the right subtree of 32.

The left subtree of 32 is 29. The right subtree of 32 is 36. Putting them together, the left subtree of 38 is 32, 29, 36.

The right subtree of 38 has only 46.

The right subtree of 27 is 38, 32, 29, 36, 46.

Now we put the left subtree of 27 and the right subtree of 27 together.

Pre-order will visit the nodes in this order: 27, 9, 3, 6, 15, 38, 32, 29, 36, 46.

Next, we consider in-order traversal.

For in-order traversal, we put 27 between the left subtree and the right subtree.

To traverse the left subtree of 27, we put 9 in the middle, between the left subtree of 9 and the right subtree of 9.

To traverse the left subtree of 9, we put 3 in the middle, between the left subtree of 3 and the right subtree of 3.

3 has no left child. 3’s right child is 6. Thus, the right subtree of 3 is 6.

The left subtree of 9 is 3, 6.

The right subtree of 9 is 15.

The left subtree of 27 is 3, 6, 9, 15.

Next, we consider the right subtree of 27.

We put 38 between the left subtree of 38 and the right subtree of 38.

To traverse the left subtree of 38, we put 32 in the middle. The left subtree of 32 is 29. The right subtree of 32 is 36.

Thus, the left subtree of 38 is 29, 32, 36.

The right subtree of 38 is 46.

Putting them together, the right subtree of 27 is 29, 32, 36, 38, 46.

Finally, the in-order traversal of this binary tree is 3, 6, 9, 15, 27, 29, 32, 36, 38, 46.

We can use pre-order, in-order, or post-order for any type of binary tree. These methods of traversal are not restricted to binary search tree.

If the binary tree is a binary search tree, then in-order traversal will output the keys in the ascending order.

This is because in a binary search tree, all keys in the left subtree are smaller than all keys in the right subtree.

Let’s see how post-order traversal works.

First, we put 27 at the end, after the left subtree of 27 and the right subtree of 27.

To traverse the left subtree of 27, we put 9 at the end, after the left subtree of 9 and the right subtree of 9.

The left subtree of 9 has 3 at the end, after the left subtree of 3 and the right subtree of 3.

3 has no left subtree. 3’s right subtree is 6.

Thus, the left subtree of 9 is 6, 3.

The right subtree of 9 is 15.

Putting them together, the left subtree of 27 is 6, 3, 15, 9.

Next, we consider the right subtree of 27.

First, we put 38 at the end, after the left subtree of 38 and the right subtree of 38.

The left subtree of 38 has 32 at the end, after the left subtree of 32 and the right subtree of 32.

The left subtree of 38 is 29, 36, 32.

The right subtree of 38 is 46.

Putting them together, the right subtree of 27 is 29, 36, 32, 46, 38.

Putting all of them together, post-order traversal of this tree generates

6, 3, 15, 9, 29, 36, 32, 46, 38, 27.

How can we write code for pre-order, in-order, and post-order?

This is how pre-order traversal can be implemented. This is a recursive function. At the top, the function checks whether the tree is empty or not. If it is empty, nothing can be done and the function returns.

Otherwise, pre-order prints the node’s value. Then, the function visits the left subtree, followed by the right subtree.

The function for in-order traversal is similar, except printing the node’s value is between visiting the left subtree and the right subtree.

Finally, the function for post-order traversal has line for printing at the end, after visiting the left subtree and the right subtree.

If we want to delete every node in a binary tree, what should we do? We can use the method of post-order traversal. We will delete the left subtree first, then, delete the right subtree, then delete the node.

It is important that this is the post-order traversal.

free tee N must be the last because after this line, tee N arrow left and tee N arrow right are not defined any more.